

IndraMotion MLC 14VRS

First Steps

Commissioning Manual
R911341517

Edition 02



Title IndraMotion
MLC 14VRS
First Steps

Type of Documentation Commissioning Manual

Document Typecode DOK-MLC***-F*STEP**V14-CO02-EN-P

Internal File Reference RS-4f210dc5b559587e0a6846a500394072-2-en-US-3

Change Record

Edition	Release Date	Note
Edition 01	2014-05	First edition
Edition 02	2015-09	Extensions and improvements

Copyright © Bosch Rexroth AG 2015

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 test, Solution Integration PLC, Safety and Control Hardware, GeKue (KaWa/MePe)

Table of Contents

	Page
1 About this documentation.....	5
1.1 Validity of the documentation.....	5
1.2 Documentation structure.....	6
1.3 Use of the safety instructions.....	6
1.3.1 Structure of the safety instructions.....	6
1.3.2 Explaining signal words and safety alert symbol.....	6
1.3.3 Symbols used.....	8
1.3.4 Signal graphic explanation on the device.....	8
1.4 Required and supplementing documentation XLC/MLC.....	8
1.5 Names and abbreviations.....	14
1.6 Customer feedback.....	14
2 First steps - IndraLogic.....	15
2.1 Project goal "First steps - IndraLogic".....	15
2.2 Technical requirements.....	15
2.3 Creating a new IndraWorks project.....	16
2.3.1 Resetting the control (CMLx5).....	16
2.3.2 Starting IndraWorks.....	16
2.3.3 Creating a new project.....	16
2.4 Adding a new device to a project.....	17
2.5 Specifying basic settings for the device.....	18
2.5.1 Device settings.....	18
2.5.2 Communication settings.....	18
2.5.3 PLC characteristics.....	21
2.5.4 Selecting function package.....	22
2.5.5 Function module settings.....	23
2.6 Creating a PLC program.....	25
2.6.1 Control of a traffic light system.....	25
2.6.2 Creating programs and function blocks.....	25
2.6.3 Editing programs and function blocks - Part 1.....	30
2.6.4 Task configuration.....	48
2.6.5 Declaring global variables.....	49
2.6.6 Testing "SEQUENCE" in simulation mode.....	50
2.6.7 Editing programs and function blocks – Part 2.....	52
2.6.8 Testing the application on the control.....	60
2.7 Creating visualization for the PLC program.....	61
2.7.1 Visualization in IndraLogic.....	61
2.7.2 Creating a new visualization.....	62
2.7.3 Editing visualization.....	62
2.7.4 Testing visualization.....	73
2.8 Restoring a project.....	74

Table of Contents

	Page
3 First steps - Motion	77
3.1 Project goal "First steps - Motion"	77
3.2 Technical requirements	77
3.3 Creating the Motion subproject	78
3.3.1 Resetting the control (CMLx5)	78
3.3.2 Starting IndraWorks	78
3.3.3 Creating a new project	78
3.3.4 Adding a new device to a project	80
3.3.5 Specifying basic settings for the device	80
3.3.6 Adding Sercos III drives (real axes) to the project	86
3.3.7 Adding a virtual axis to the project	87
3.3.8 Downloading the Motion subproject	88
3.4 Programming the motion sequence in function block language (FBD)	91
3.4.1 General information	91
3.4.2 Programming the virtual axis "vAxis1", programming language FBD	92
3.4.3 Downloading the PLC subproject "virtual axis" "vAxis1" (FBD)	99
3.4.4 Recording the motion of the axis "vAxis1" (trace), FBD	102
3.4.5 Programming the real axis Drive1 (FBD)	107
3.4.6 Downloading the complete PLC project (FBD)	111
3.4.7 Workflow "Virtual axis → Electronic gear → Real axis "	111
3.4.8 Recording the motion of both axes (trace), FBD	112
3.5 Programming the motion sequence in Structured Text (ST)	113
3.5.1 General information	113
3.5.2 Programming the virtual axis "vAxis1", programming language ST	115
3.5.3 Downloading PLC subproject "virtual axis" "vAxis1" (ST)	122
3.5.4 Recording the motion of the axis "vAxis1" (trace), ST	125
3.5.5 Programming the real axis "Drive1" (ST)	131
3.5.6 Downloading the complete PLC project (ST)	136
3.5.7 Workflow "Virtual axis → Electronic gear → Real axis "	136
3.5.8 Recording the motion of both axes (trace), ST	137
3.6 Restoring the project	138
4 First steps - FlexProfile	141
4.1 Project goal "First steps - FlexProfile"	141
4.2 Technical requirements	142
4.3 Creating the Motion subproject	142
4.4 Creating a FlexProfile	144
4.4.1 Creating a FlexProfile, general information	144
4.4.2 Creating the FlexProfile via IndraWorks seadogs	144
4.4.3 Creating the FlexProfile via PLC	146
4.5 Completing the PLC subproject, executing a FlexProfile motion step	150
4.6 Downloading and testing the PLC subproject	153
4.7 Workflow "Virtual axis → FlexProfile 0 → Real axis "	154
4.8 Recording the motion of both axes (trace)	155
4.9 Restoring the project	156

Table of Contents

	Page
5 First steps - Robot Control V2.....	159
5.1 Project goal "First steps - Robot Control V2".....	159
5.2 Required hardware and software.....	159
5.3 Procedure when creating an RoCo project	159
5.4 Details on the "RoCoV2TestV14" project	170
6 Service and support.....	175
Index.....	177

1 About this documentation

1.1 Validity of the documentation

Overview on target groups and product phases

In the following illustration, the framed activities, product phases and target groups refer to the present documentation.

Example: In the product phase "Mounting (assembly/installation)", the "mechanic/electrician" can execute the activity "unpack, mount and install" using this documentation.

This documentation is intended for users (first-time users) to familiarize with the basics of the IndraWorks Engineering system using an IndraMotion MLC control.

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.

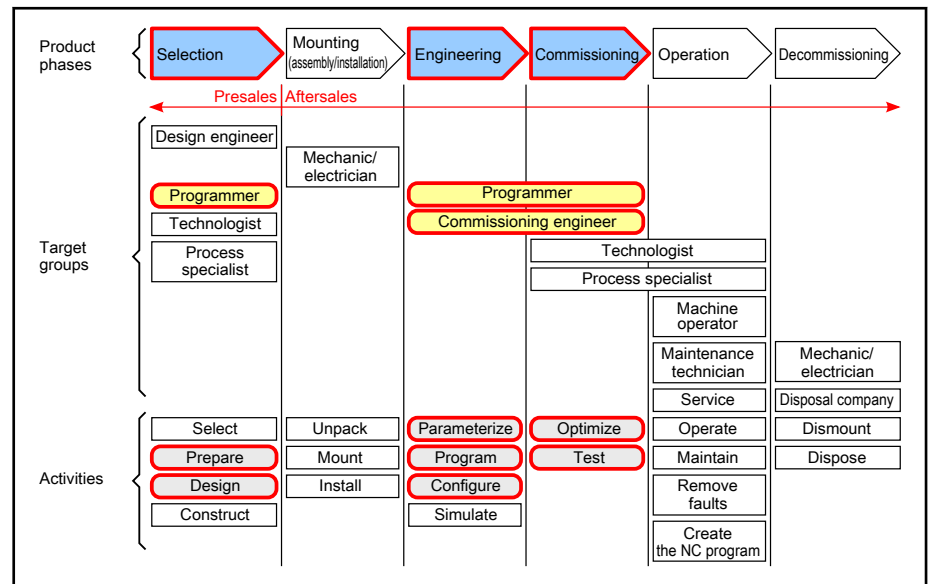


Fig. 1-1: Assigning this documentation to the target groups, product phases and target group activities

Purpose

This documentation includes the following descriptions about IndraMotion MLC 14VRS:

- First steps - IndraLogic
Programming basics with IndraLogic and visualization
 - First steps - Motion
Basics of the control of a drive from a PLC program (PLCopen), recording motion of axes via trace
 - First steps - FlexProfile
Creation and application of MotionProfiles
 - First steps - Robot Control V2
From the creation of a Robot Control project up to its execution
- The online help of the system is based on this documentation.

About this documentation



For a description of the "First steps with a SafeLogic control", refer to:

Rexroth IndraWorks

SafeLogic 14VRS

First Steps

Part number [R911341520](#)

1.2 Documentation structure

For the programming basics with IndraLogic and visualization, refer to [chapter 2 "First steps - IndraLogic" on page 15](#). The chapter can also be understood without control in the simulation mode (soft PLC).

For the basics of the control of a drive from a PLC program (PLCopen) and of recording the motion of axes via trace, refer to [chapter 3 "First steps - Motion" on page 77](#). This chapter can also be understood with a real control without drive (with virtual axes).

For information on the creation and the use of MotionProfiles, refer to [chapter 4 "First steps - FlexProfile" on page 141](#). This chapter is based on [chapter 3 "First steps - Motion" on page 77](#).

For information from the creation of a Robot Control V2 project up to its execution, refer to [chapter 5 "First steps - Robot Control V2" on page 159](#).

For information on the customer service helpdesk, refer to [chapter 6 "Service and support" on page 175](#).

1.3 Use of the safety instructions

1.3.1 Structure of the safety instructions

The safety instructions are structured as follows:

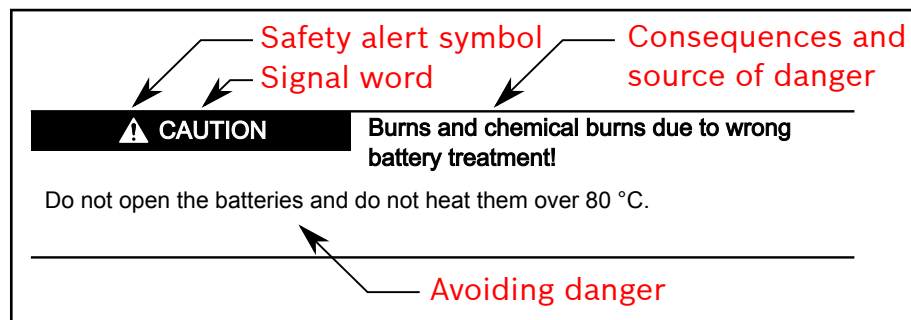


Fig. 1-2: Structure of the safety instructions

1.3.2 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 used to draw attention to the safety instruction and also provides information on the severity of the hazard.

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.

About this documentation

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

In the event of non-compliance with this safety instruction, minor or moderate injury can occur.

NOTICE

In the event of non-compliance with this safety instruction, material damage can occur.

About this documentation

1.3.3 Symbols used

Hints are represented as follows:



This is an information.

Tips are represented as follows:



This is a tip.

1.3.4 Signal graphic explanation on the device



Prior to the installation and commissioning of the device, refer to the device documentation.

1.4 Required and supplementing documentation XLC/MLC

Documentation titles with type codes and part numbers

System overview
Rexroth IndraLogic XLC 14VRS System Overview DOK-XLC***-SYSTEM**V14-PRRS-EN-P, R911341483 This documentation provides an overview on the possible hardware/software components of the automation system IndraLogic XLC of the mentioned version. The documentation facilitates assembling a system.
Rexroth IndraMotion MLC 14VRS System Overview DOK-MLC***-SYSTEM**V14-PRRS-EN-P, R911341504 This documentation provides an overview of the hardware/software components of the automation system IndraMotion MLC in the mentioned version. The documentation facilitates assembling a system.

Tab. 1-1: XCL/MLC documentation overview - System overview

First steps
Rexroth IndraMotion MLC 14VRS First Steps DOK-MLC***-F*STEP**V14-CORS-EN-P, R911341517 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.
Rexroth IndraLogic XLC 14VRS First Steps DOK-XLC***-F*STEP**V14-CORS-EN-P, R911341489 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.
Rexroth IndraMotion MLC 14VRS Project Conversion DOK-XLCMLC-PROCONV*V14-APRS-EN-P, R911341494 This documentation describes the project conversion of the IndraMotion MLC.

Tab. 1-2: XCL/MLC documentation overview - First steps

Engineering
Rexroth IndraWorks 14VRS Software Installation DOK-IWORKS-SOFTINS*V14-CORS-EN-P, R911344286 This documentation describes the IndraWorks installation.
Rexroth IndraWorks 14VRS Engineering DOK-IWORKS-ENGINEE*V14-APRS-EN-P, R911343566 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.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Function Description DOK-XLCMLC-FUNC***V14-APRS-DE-P, R911341700 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraMotion MLC.

Tab. 1-3: XCL/MLC documentation overview - Engineering

Diagnostics and service
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics DOK-XLCMLC-DIAG***V14-RERS-EN-P, R911341481 This documentation includes all control diagnostics implemented in the control systems IndraLogic XLC and IndraMotion MLC.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Parameters DOK-XLCMLC-PARAM***V14-RERS-EN-P, R911341479 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.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Commissioning DOK-XLCMLC-STARTUP*V14-CORS-EN-P, R911341502 This documentation describes the steps to commission and service the IndraMotion MLC and IndraLogic XLC systems. It includes checklists for frequent tasks and a detailed description of the steps.
Rexroth IndraWorks IndraMotion Service Tool DOK-IWORKS-IMST*****-APRS-DE-P, R911341383 This documentation describes the IndraMotion Service Tool (IMST). IMST is a web-based diagnostic tool used to access a control system via an Ethernet high-speed connection. The IMST allows OEMs, end users and service engineers to access and remotely diagnose a system. The PC has to use at least Internet Explorer 8, Firefox 3.5 or a higher version. The following control variants are supported: • IndraMotion MLC L25/L45/L65/L75/XM2/VPx • IndraLogic XLC L25/L45/L65/L75

Tab. 1-4: XCL/MLC documentation overview - Diagnostics and service

About this documentation

Drive controllers
Rexroth IndraDrive Firmware for Drive Controllers MPH-, MPB-, MPD-, MPC-08 DOK-INDRV*-MP*-08VRS**-APxx-EN-P, R911332643 This documentation describes all functional properties of the IndraDrive firmware in the variants MPH-08, MPB-08, MPD-08 and MPC-08.
Rexroth IndraDrive Drive Controllers MPx-02 to MPx-08 DOK-INDRV*-GEN-**VRS**-PAxx-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.
Rexroth IndraDrive MPx-02 to MPx-08 and HMV DOK-INDRV*-GEN-**VRS**-WAxx-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 during troubleshooting.
Rexroth IndraDrive Control Units CSB02, CSE02, CSH02, CDB02 DOK-INDRV*-CXX02*****-PRRS-EN-P, R911338962
Rexroth IndraDrive MPx-18 Functions DOK-INDRV*-MP*-18VRS**-APRS-EN-P, R911338673 This documentation describes all functional properties of the IndraDrive firmware in the variants MPB-18, MPM-18, MPC-18 and MPE-18.

xx

Corresponding edition

Tab. 1-5:

XCL/MLC documentation overview - Drive controllers

PLC
Rexroth IndraWorks 14VRS IndraLogic 2G PLC Programming System DOK-IWORKS-IL2GPRO*V14-APRS-EN-P, R911343571 This documentation describes the PLC programming tool IndraLogic 2G and its usage. It includes the basic use, first steps, visualization, menu items and editors.
Rexroth IndraWorks 14VRS Basic Libraries IndraLogic 2G DOK-IL*2G*-BASLIB**V14-LIRS-EN-P, R911343920 This documentation describes the system-comprehensive PLC libraries.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries DOK-XLCMLC-FUNLIB**V14-LIRS-EN-P, R911341491 This documentation describes the function blocks, functions and data types of the RIL_CommonTypes, ML_Base and ML_PLCopen libraries for the IndraLogic XLC/IndraMotion MLC. It also includes the error reactions of function blocks.

Tab. 1-6:

XCL/MLC documentation overview - PLC

Field buses
Rexroth IndraWorks 14VRS Field Buses DOK-IWORKS-FB*****V14-APRS-EN-P, R911341485 This documentation describes the field bus and local periphery connections supported by the IndraLogic XLC, IndraMotion MLC and IndraMotion MTX systems. The focus of this documentation is on the configuration, parameterization, commissioning and the diagnostics of the different periphery connections. This documentation is the basis for the online help.
Rexroth IndraWorks 14VRS Field Buses Libraries DOK-IWORKS-FB*LIB**V14-LIRS-EN-P, R911343575 This documentation describes the field bus libraries: RIL_ProfibusDP_02, RIL_ProfibusDP Slave, RIL_ProfinetIO, RIL_ProfinetIODevice, RIL_EtherNetIPAdapter, RIL_MappingList, RIL_SERCOSIII, RIL_Inline including their diagnostics and error reactions of the function blocks.
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.
Sercos System Manual for I/O Devices DOK-CONTRL-ILS3*****-APxx-EN-P, R911333512 This documentation describes the configuration, parameterization, commissioning and diagnostics of I/O devices with a Sercos interface.

xx Corresponding edition
Tab. 1-7: XCL/MLC documentation overview - Field buses

HMI
Rexroth IndraWorks 14VRS HMI DOK-IWORKS-HMI*****V14-APRS-EN-P, R911343569 This documentation describes the functions, configuration and operation of the user interfaces IndraWorks HMI Engineering and IndraWorks HMI Operation.
Rexroth IndraWorks 14VRS WinStudio DOK-IWORKS-WINSTUD*V14-APRS-EN-P, R911341585 This "User Manual and Technical Reference Book" facilitates working with the "Rexroth WinStudio" software for optimal results. This document provides technical information and step-by-step instructions to create web-enabled HMI/SCADA programs.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS HMI Connection DOK-XLCMLC-HMI*****V14-APRS-EN-P, R911341497 This documentation describes the visualization systems supported by the IndraLogic XLC and IndraMotion MLC and their connection.

Tab. 1-8: XCL/MLC documentation overview - HMI

About this documentation

Technology
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Generic Application Template DOK-XLCMLC-TF*GAT**V14-APRS-EN-P, R911341487 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.
Rexroth IndraMotion MLC 14VRS Technology Libraries DOK-MLC***-TF*LIB**V14-LIRS-EN-P, R911341511 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.
Rexroth IndraWorks Energy Efficiency Management 14VRS DOK-IWORKS-4EE*****V14-APRS-EN-P, R911339229 This documentation describes the HMI connection of the energy management system and the function blocks, functions and data types of the "RIL_4EE " library. The documentation also includes the error reactions of the function blocks.
Rexroth IndraMotion MLC 14VRS RegisterControl (Library) DOK-MLC***-REGI*CO*V14-LIRS-EN-P, R911341509 This documentation describes the inputs and outputs of the individual function blocks and provides notes on their use.
Rexroth IndraMotion MLC 14VRS RegisterControl (Application Description) DOK-MLC***-REGI*CO*V14-APRS-EN-P, R911341507 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.
Rexroth IndraMotion MLC 14VRS Winder Function Application DOK-MLC***-TF*WIND*V14-APRS-EN-P, R911341513 This application-related system documentation describes the application of the winder technology functions.
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.
Rexroth IndraMotion MLC 14VRS Robot Control V2 DOK-MLC***-ROCO****V14-RERS-EN-P, R911341588 This documentation provides information about the Robot Control V2. The focus is on PLCopen programming. The program structure, variables, functions, motion statements and the required system parameters are described. This documentation also includes the description of the "ML_Robot" library. Robot Control V2 is supported by a kinematic interface. The components of this interface are described in the "ML_KinTech" library.

Tab. 1-9: XCL/MLC documentation overview - Technology

Open Core Engineering - OCE
Rexroth IndraLogic XLC IndraMotion MLC 14VRS First Steps MLPI DOK-XLCMLC-MLPI****V14-CORS-EN-P, R911341033 This document describes the installation and the commissioning of the MLPI interface.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Automation Interface DOK-XLCMLC-AUT*INT*V14-APRS-EN-P, R911341499 This documentation describes the script-based access to IndraWorks project data via the interface of the Automation Interface.

Tab. 1-10: XCL/MLC documentation overview - Open Core Engineering - OCE

SafeLogic
Rexroth IndraWorks SafeLogic 14VRS First Steps DOK-IWORKS-SL*STEP*V14-CORS-EN-P, R911341520 This documentation describes the initial commissioning of the Safety function module and its external Safety periphery. Taking the example of a project, all required steps to commission a simple Safety application are executed.
Rexroth IndraMotion MLC IndraLogic XLC 14VRS SafeLogic System Overview DOK-XLCMLC-SL**SYS*V14-PRRS-EN-P, R911341696 This documentation describes the project planning, configuration, creation and commissioning of safety-related devices.
Rexroth IndraWorks 14VRS SafeLogic Project Configuration DOK-IWORKS-SL**PRJ*V14-APRS-EN-P, R911341694 This documentation describes the configuration of Safety applications.
Rexroth IndraControl SafeLogic Function Module DOK-CTRL-SL**FM*****-ITRS-DE-P, R911336576 This documentation is provided with the hardware and describes the commissioning, the installation, the decommissioning and disposal of the hardware.

Tab. 1-11: XCL/MLC documentation overview - SafeLogic

Hardware
Rexroth IndraControl L45/L65/L85 Control DOK-CTRL-ICL45L65L85-PRxx-EN-P, R911332116 This documentation describes the IndraControl L45/L65/L85 controls.
Rexroth IndraControl L25 DOK-CTRL-IC*L25*****-PRxx-EN-P, R911328474 This documentation describes the IndraControl L25 controls.
Rexroth IndraControl Lxx 14VRS Function Modules DOK-CTRL-FM*LXX**V14-APRS-EN-P, R911341583 This documentation describes all function modules of the Lxx controls including engineering and diagnostics.

xx Corresponding edition

Tab. 1-12: XCL/MLC documentation overview - Hardware

Hydraulics
Rexroth IndraMotion MLC 14VRS Hydraulic Functions DOK-MLC***-TF*HMOT*V14-LIRS-EN-P, R911341907 This documentation describes the function blocks, functions and data types of the MH_TechHydrBase and MH_TechHydr-Motion libraries. The documentation also includes the error reactions of the function blocks.
Rexroth IndraMotion MLC 14VRS Sequential Programming DOK-XLCMLC-SEQPROG*V14-LIRS-EN-P, R911341909 This documentation describes the sequential programming options, also when using GAT, and the MH_HydrControlProg library.

Tab. 1-13: XCL/MLC documentation overview - Hydraulics

About this documentation

1.5 Names and abbreviations

Term	Explanation
IndraMotion MLC	Compact Motion Logic systems with Motion, Robot and Logic Control functionalities
IndraWorks Engineering Framework	Project planning and commissioning tool of Bosch Rexroth
IndraDrive	Drive controller
Wizard	Wizard guiding through several dialogs for an ergonomic data input
Function block	Provides application-specific functionalities
IEC	International Electrotechnical Commission
Sercos	Sercos (Serial real-time communication system) interface is a world-wide standardized interface for the communication between controls and drives
XML	Extensible Markup Language (XML). It is a markup language to represent hierarchical structured data as text data

Tab. 1-14: Names and abbreviations used

1.6 Customer feedback

Customer requests, comments or suggestions for improvement are of great importance to us. Please email your feedback on the documentations to Feedback.Documentation@boschrexroth.de. Directly insert comments in the electronic PDF document and send the PDF file to Bosch Rexroth.

2 First steps - IndraLogic

2.1 Project goal "First steps - IndraLogic"

The draft and implementation of a simple traffic light control is used to introduce to the PLC programming and commissioning of the control. The PLC programming languages Structured Text (ST) and Sequential Function Chart (AS), Function Block Diagram (FBD) and, if required, Continuous Function Chart (CFC) are used.

A simple commissioning visualization completes the project.

This section describes a few typical operating sequences in IndraWorks. Proceed with these descriptions of the operating procedures to become familiar with IndraWorks and to see how the individual components interact.

Project restoration:

Irrespective of the hardware used, for example IndraMotion MLC L25, L45, L65, L75, XM2 or VPx, the project can be restored on your computer using your control.

Therefore, use the export file "Application_FirstSteps_IndraLogic_Basics.iwx" stored in your installation directory under "Drive:\All Programs\Rexroth\IndraWorks\Projects". For more detailed information, refer to [chapter 2.8 "Restoring a project" on page 74](#).



By default, IndraWorks is installed in "C:\Program Files\Rexroth". The IndraWorks installation directory can differ from the given installation path.

Subtasks

- Create a project with a control
- Edit the PLC program
- Edit the PLC visualization
- Simulate the PLC program
- Load the PLC program to the control

2.2 Technical requirements

- Software:
 - PC with installed IndraWorks Engineering for IndraMotion MLC 14VRS
- Control:
 - IndraMotion MLC L25
 - IndraMotion MLC L45
 - IndraMotion MLC L65
 - IndraMotion MLC L75
 - IndraMotion MLC XM2
 - IndraMotion MLC VPx

Meeting these prerequisites is described in:

- IndraMotion MLC 14VRS Functional Description, chapter "Commissioning and operation"

A control, e.g. of type "**IndraMotion MLC L65**" has to be available with the current firmware 14VRS. Additionally, the following requirements have to be met:

First steps - IndraLogic

- The IP address is known
- The control is connected to the programming device (PC) via Ethernet
- A valid firmware is loaded to the control
- If a Sercos III bus is available, it has to be bridged

2.3 Creating a new IndraWorks project

2.3.1 Resetting the control (CMLx5)

From previous use, the control can contain old program components, parameterizations and PLC data that are removed in this step by clearing the memory.

After the control is switched on, press the outer buttons (<Esc> and <Enter>) below the display and when **BOOT 1.02** is displayed, keep them pressed until **BOOTSTOP** is displayed.

Subsequently, press the two inner keys. **CLEANUP?** is displayed.

Confirm with <Enter>. **CLEANUP** is displayed. If the clean-up was completed successfully, **SUCCESS** and subsequently **REBOOT** are displayed.

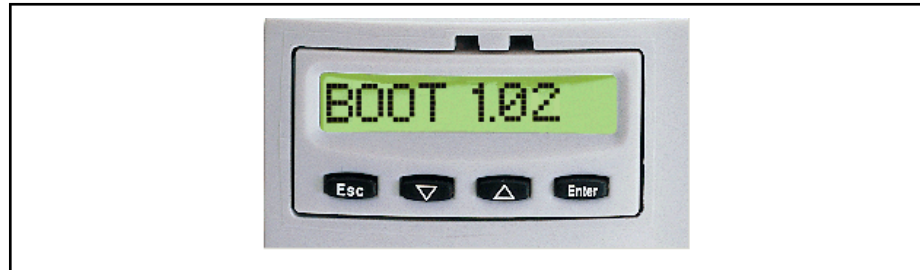


Fig. 2-1: Rebooting the control

The control without Sercos interface or bridged Sercos interface runs up to **BB RDY**. The memory is cleared and any possible Motion component present and the PLC component are ready for operation.

2.3.2 Starting IndraWorks

Start IndraWorks via **Start ► All Programs ► Rexroth ► IndraWorks ► Engineering**.

The IndraWorks engineering environment is started. IndraWorks can also be

started via the desktop icon .

2.3.3 Creating a new project

To create a new project, click on "Create an empty project" in the workspace.

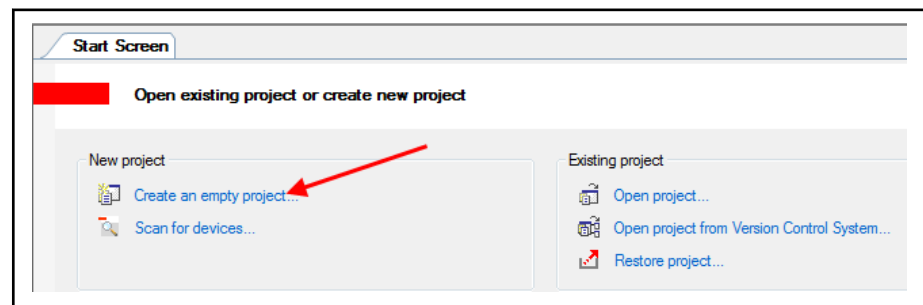


Fig. 2-2: Creating empty project

A new project can also be created via **File ► New ► Project...**

The wizard to specify the project name and the storage location settings for a new IndraWorks project starts (refer to the following figure).

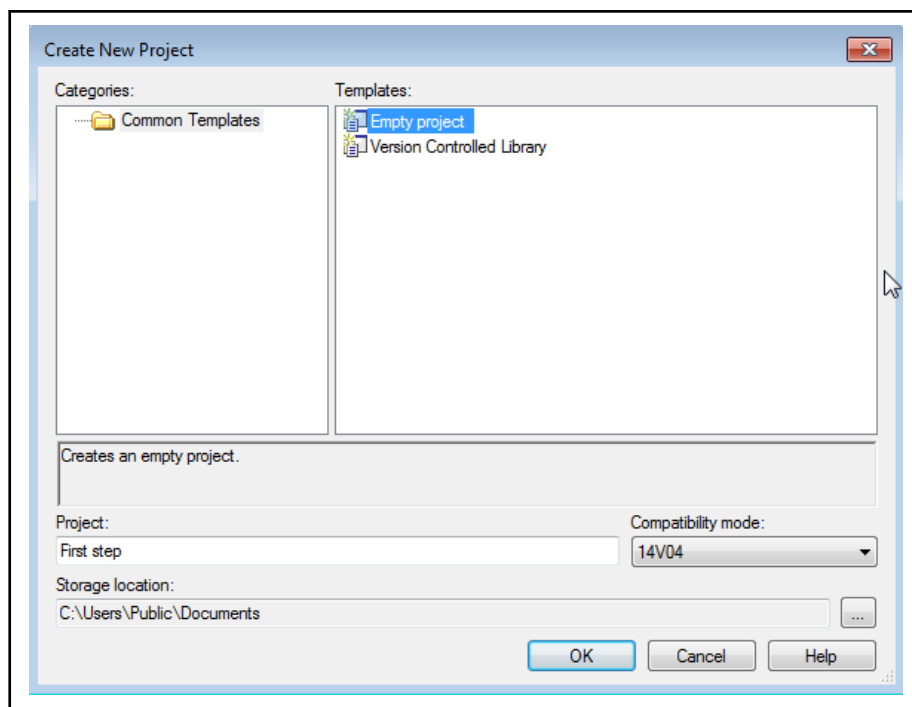


Fig. 2-3: Settings to save a new IndraWorks Project

- Project name** Specify a name for the project. Suggestion: "First steps".
 - Storage location** The field contains a predefined storage location in the directory "C:\Documents and Settings\username\MyDocuments".
A different directory can also be specified.
 - Compatibility mode** The IndraWorks interface scope corresponds to 14V04.
 - Project language** When selecting the project language, the language of the user texts is determined.
- The newly created project is displayed in the Project Explorer with the previously defined project name:



Fig. 2-4: "First steps" project

2.4 Adding a new device to a project

Drag the device (e.g. folder "IndraMotion MLC", "IndraControl L65") from the "Drive and Control" library and drop it onto the newly created project in the Project Explorer.

First steps - IndraLogic



- (1) Device: IndraMotion MLC L65
- (2) Project: First steps

Fig. 2-5: Adding a device

Subsequently, a wizard starts automatically to make the necessary basic settings as described below.

2.5 Specifying basic settings for the device

2.5.1 Device settings

Enter the general information on the device to be added.

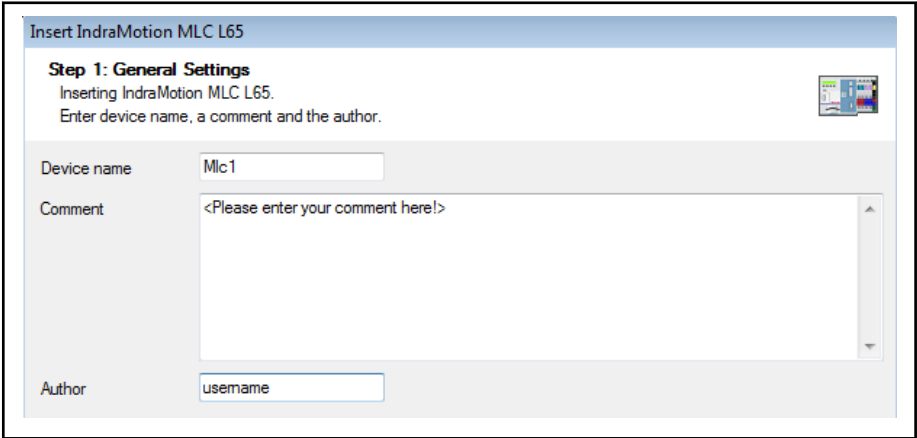


Fig. 2-6: Device settings

- Device name** Enter any device name, e.g. "Mlc1" in "Device name". By default, the field includes the name from the library. The device is subsequently displayed in the Project Explorer under the specified name.
- Comment** Add a comment that describes the device in detail. This field is optional.
- Author** Enter the name of the project author. This field is optional.



These entries can also be changed later on.

To confirm the entries, press **Next >>**.

2.5.2 Communication settings

In the "Configuration" dialog, define the settings for the firmware and communication between IndraWorks and the device (in this case: IndraMotion MLC L65).

Fig. 2-7: Creating a control (shown for the "IndraMotion MLC L65") - Hardware/communication

Device type The control selection predetermines the device type. If there are several different variants for one device type, select the device variant from the drop-down list.

Firmware version Select the firmware version here. This setting can generally remain.

Firmware release The firmware release in a firmware version is set here. This setting should also remain.

IP address  The Ethernet communication entries are not evaluated in the [chapter 2.6.6 "Testing SEQUENCE in simulation mode" on page 50](#). If the control is operated in the simulation mode, a connection test is not required.

Enter the target address under "IP address", i.e. the IP address of the control to which the application is to be loaded.

PLC gateway In "Gateway IP address", define the address of the computer on which the gateway server is running. By default, "localhost" is provided as computer name (address), "localhost" means that the locally installed gateway is addressed. Do not change this setting.

PLC communication In 14VRS, only TCP is still supported at the Engineering interface.

Due to compatibility reasons, UDP is provided for projects of 13VRS.

Connection test Click on **Execute** to check whether the communication between computer and target control is successful. The test result is logged in the "Connection test" field.

If the PLC communication test was successful, the IP address is shown.

First steps - IndraLogic

Any possible problems are signaled by flashing red or yellow icons in the window.

The following problems can occur:

- Output: "Communication to the control <Control name> with the IP address <IP address> failed!"
Causes:
 - The control is not switched on
 - The network cable is not connected
 - The IP address of the control and the IP address set do not match
Solution: Change the IP address in the dialog or the IP address of the control (IndraMotion MLC 14VRS Functional Description, chapter "Commissioning and operation", subchapter "Setting the IP address")
 - The network mask of the PC and/or control is incorrectly set
Solution: Change the IP address in the dialog or the IP address of the control (IndraMotion MLC 14VRS Functional Description, chapter "Commissioning and operation", subchapter "Setting the IP address")
- Output: "The configured firmware and the firmware in the control do not match."
Causes:
 - The control firmware is not up-to-date
Solution: Updating firmware (IndraMotion MLC 14VRS Functional Description, chapter "Control", subchapter "Firmware management")
 - The selected control does not correspond to the real control type

Click on **Next >>** to go to the next page.

2.5.3 PLC characteristics

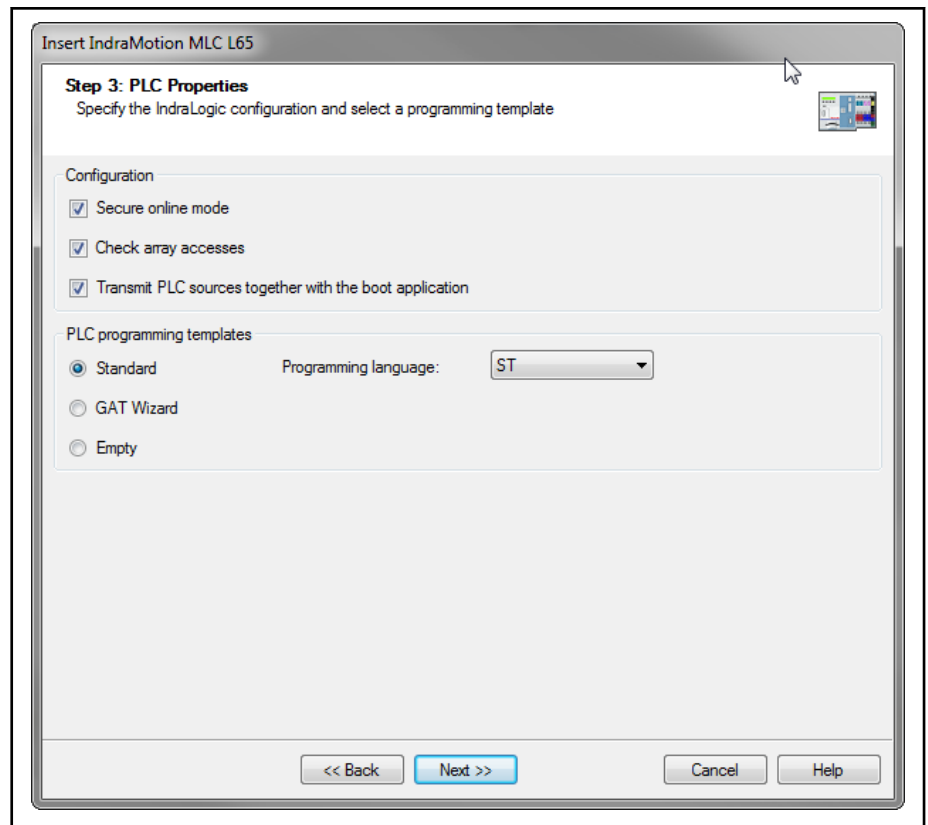


Fig. 2-8: Creating a control - PLC properties

Secure online mode

If the option "Safe online mode" is selected, a prompt is displayed asking for confirmation before any security-relevant actions are executed.

Checking array access

If this switch is enabled, the check function "CheckBounds" from the template database is automatically inserted below the PLC application node.

If this switch is disabled again in case of an already existing control, the check function "CheckBounds" is only automatically removed if the following comment block is still present in the implementation section of the check function:

```
(*****
This function was generated by the 'check array access'
option of the "Properties" dialog of the controller. If
you delete this comment, this function is not removed by
disabling the 'check array access' option.
*****)
```

To ensure that customized adaptations are not deleted in the implementation section of the check function when the switch is disabled, remove the comment block.

First steps - IndraLogic



As the automatically inserted check function is not removed anymore when deleting the comment block, the "CheckBounds" function has to be deleted manually from the application to disable the monitoring of the array limits.

A disabled switch does not mean that no CheckBounds functionality is active.

If the check function is inserted or deleted, login via online change is not possible. A download is required.

Transferring PLC sources together with the boot application

The checkbox is only selected if the PLC application type "Standard" or "GAT Wizard" is enabled and if there is no user-defined CheckBounds function.

When transferring the boot application, PLC program sources (POUs, tasks, etc.) are transferred to the control in addition to the compilation. Use this function to additionally display a detailed comparison for the PLC elements during online comparison.

Programming templates

Select between the following variants for the PLC application to be created:

Default: The empty default programs "MotionProg" and "PlcProg" are automatically created in the selected programming language. For other PLC objects, the programming language is selected when creating the object.

PLC programming language: Set the preferred PLC programming language. Select "ST" (Structured Text).

GAT wizard: Example projects or programming templates can be selected in an additional dialog. GAT is a programming frame ready for operation with predefined locations for the application code. Professional and well-structured application programs with prepared elements are automatically created using the dialog-based code generation of the GAT Wizard.

Empty: Creates an empty, non-executable PLC application without programs and tasks.

Click on **Next >>** to go to the next page.

2.5.4 Selecting function package



Due to the acquired firmware, the function packages are predefined and can thus not be selected for the controls based on the IndraControl L25/L45/L65/L75. For the controls based on IndraControl XM and VPx, select the function packages according to the purchased licenses!

1. If a function package is deselected, all function packages dependent on that function package are also deselected.
2. If a function package, which is subject to license, is selected, a license icon is automatically generated. This icon displays license information via tooltip or mouse click.
3. Depending on the activated function packages, the nodes below the created control are created in the IndraWorks project tree.
4. Even after creating a control, the function packages can be added or deleted via the corresponding "Properties" dialog.
5. When converting the device to a control type with a different functional scope, a conversion preview informing on the upcoming changes is still displayed.



If the function package "Motion" is removed for example, all configured axes are also deleted.

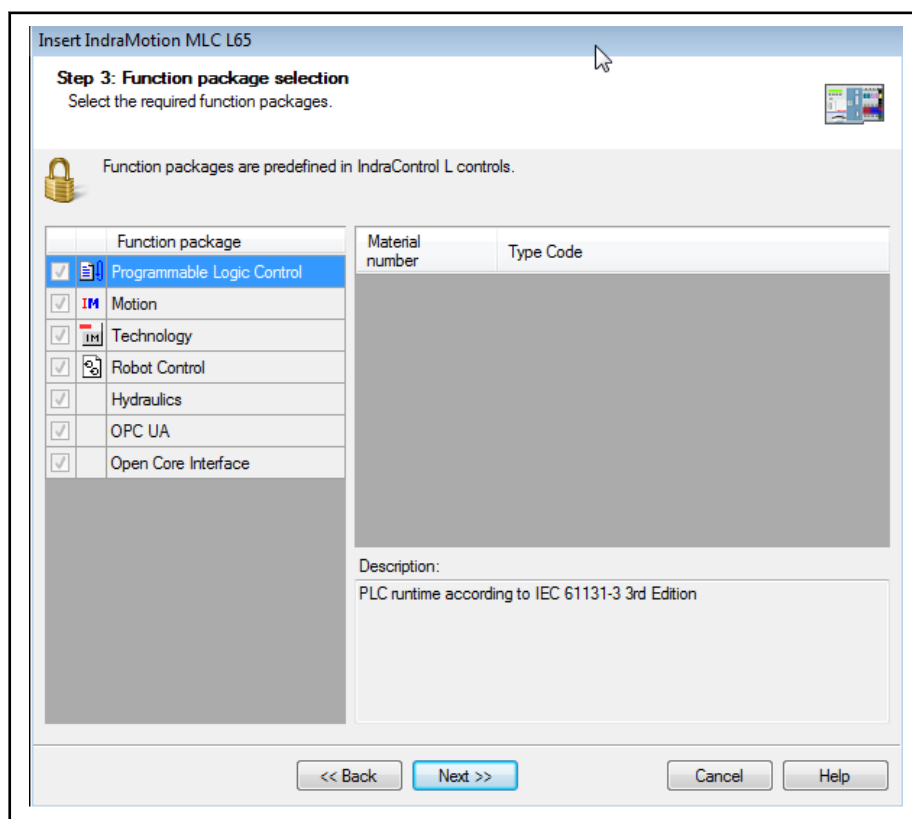


Fig. 2-9: Overview on the acquired function packages

Programmable Logic Control

The function package "Programmable Logic Control" includes the PLC runtime system based on the IEC 61131-3 3rd edition. The function package is enabled by default and cannot be deselected.

Motion

Firmware-based control of single-axis and synchronized multiple-axis motions.

Technology

PLC libraries for technology functions.

Robot Control

General kinematics and transformations to control robots.

Hydraulics

Central control of hydraulic axes.

OPC-UA

OPC UA server, OPC unified architecture, standardized variable exchange between control and visualization.

Open Core Interface

Interface for Rexroth Open Core Engineering.

The programming interface called Motion Logic programming interface (MLPI/OCI) allows to access functionalities encapsulated in the firmware of the IndraMotion MLC system. Thus, it is easy and fast to develop Motion Control applications in the programming languages C/C++/C#/VBA/JAVA using the engineering environment "Visual Studio" (see [Engineering network](#)).

2.5.5 Function module settings



Depending on the device type, the "Interfaces" dialog can differ. In this project, the dialog for the "IndraMotion MLC L65" device is exemplarily described.

First steps - IndraLogic

Enter the device and function module settings.

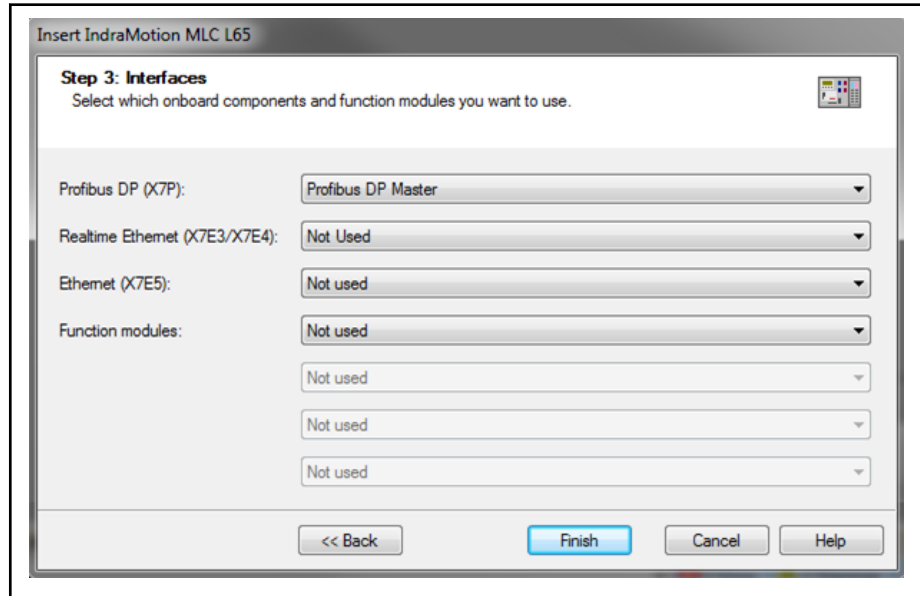


Fig. 2-10: Function module settings

Profibus DP (X7P) Current Profibus configuration that can be selected when adding a device. "Profibus DP master", "Profibus DP slave" or "Not used" can be selected.

Real-time Ethernet (X7E3/X7E4) Current Profinet or Ethernet/IP configuration that can be selected when adding a device. "Not used" or a Profinet or Ethernet/IP interface can be selected.

Ethernet (X7E5) As this interface is needed as Engineering interface (connection directly to your PC or via the operating network), the Ethernet/IP Adapter may not be enabled.

Function modules Up to four function modules can be connected to an "IndraMotion MLC L65/L75 / IndraLogic XLC L65/L75".

The function modules can be selected from the drop-down list if required.

To confirm the entries, press **Finish**.

The newly created device is displayed in the Project Explorer.

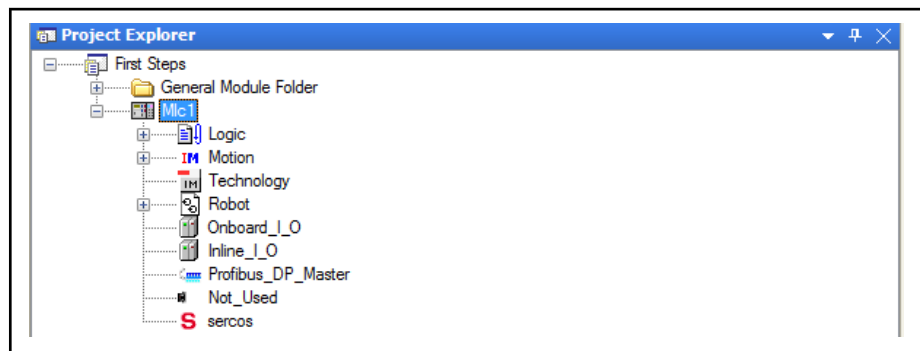


Fig. 2-11: Project Explorer: The "First steps" project with an IndraMotion MLC L65 control (with Sercos III bus)



The Project Explorer mirrors the components selected in the dialogs "Configuration" and "Interfaces".

2.6 Creating a PLC program

2.6.1 Control of a traffic light system

The example program "Mini traffic signal" is described in the following. The program is intended to control two traffic lights at crossroads. Both traffic lights alternate in their red and green phases. To prevent accidents, a yellow switching phase or yellow-and-red switching phases are intended between the red and green phases.

This example shows the following:

1. How time-dependent programs are displayed with the standard IEC 61131-3 language constructs.
2. How to edit the various languages of the standard using IndraLogic.
3. How to connect the program parts with each other.

2.6.2 Creating programs and function blocks

Adding "PlcProg" program in "CFC" or "ST"

The POU (Program Organization Units) "MlcProg" and "PlcProg" are preset using the language "Structured Text (ST)" and are available immediately after creating a control.

The POU "MlcProg" is not required in our project.



The following can be selected for the "PlcProg" program:

- When creating in "Structured Text (ST)", omit the rest of this chapter and go to [chapter "Adding SEQUENCE program in SFC" on page 28](#)
- When creating in "Continuous Function Chart (CFC)", continue in the text

Proceed as follows:

1. In the Project Explorer, highlight the "PlcProg" POU.
2. Right-click to open the context menu and select the menu item **Delete**.



The highlighted POU "PlcProg" can also be deleted with <Del>.

Add a new POU to your project by dragging the POU from the "PLC objects/ General objects" library dropping it to the "Application" node. Alternatively, use the command **Add ► POU** in the context menu of the application node.

First steps - IndraLogic

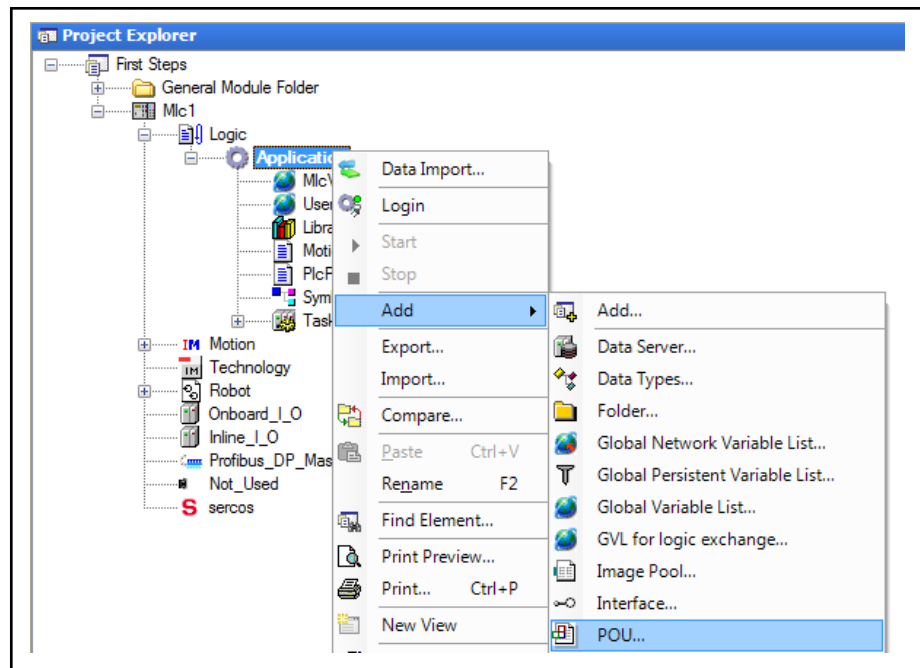


Fig. 2-12: Adding a POU from the library

The "Add object" dialog opens. Make the following settings in this dialog:

1. For the POU, enter the name "PlcProg" into the input field "Name:".
2. In the selection area, select "Program" under "Type:". Then, go to "Implementation language:" (programming language) and select "Continuous function chart (CFC)".

Press **Finish** to confirm the entries.

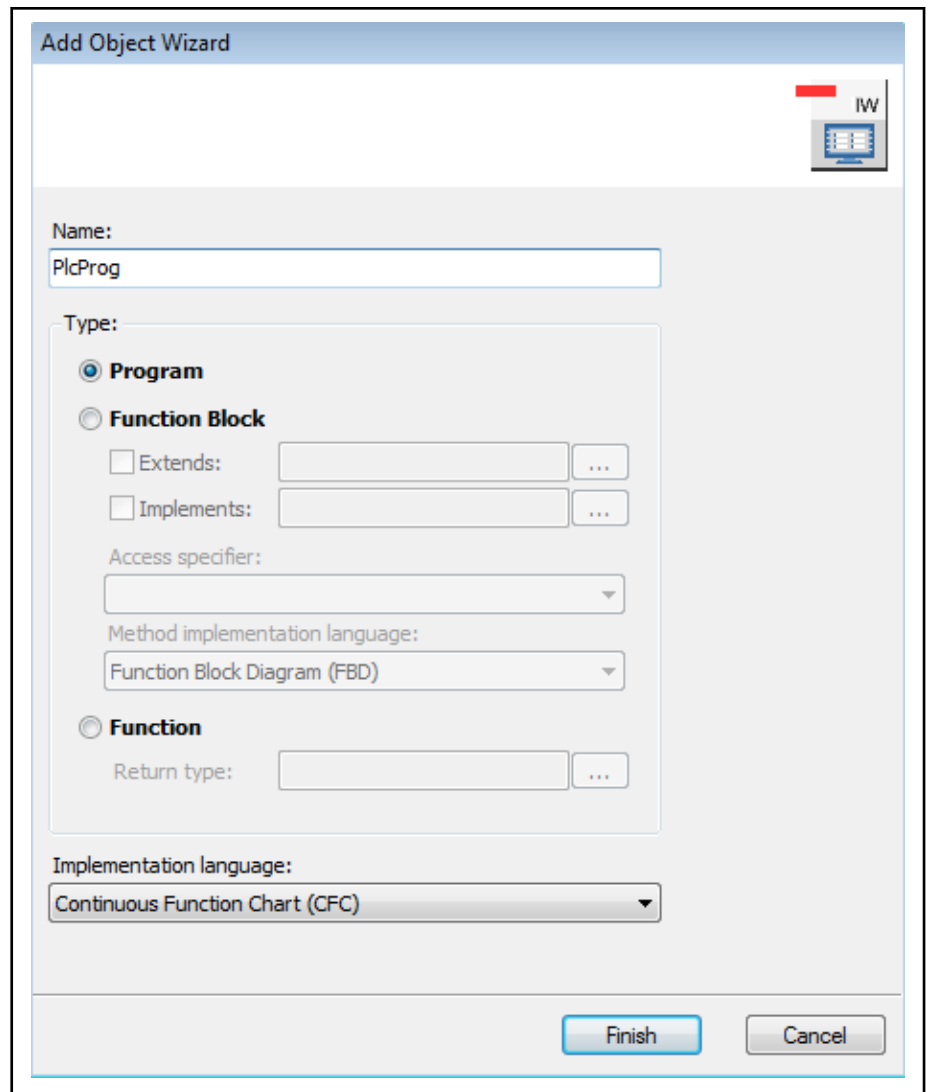


Fig. 2-13: Configuring POU "PlcProg"

The POU object "PlcProg", language "CFC", is now available in the project.

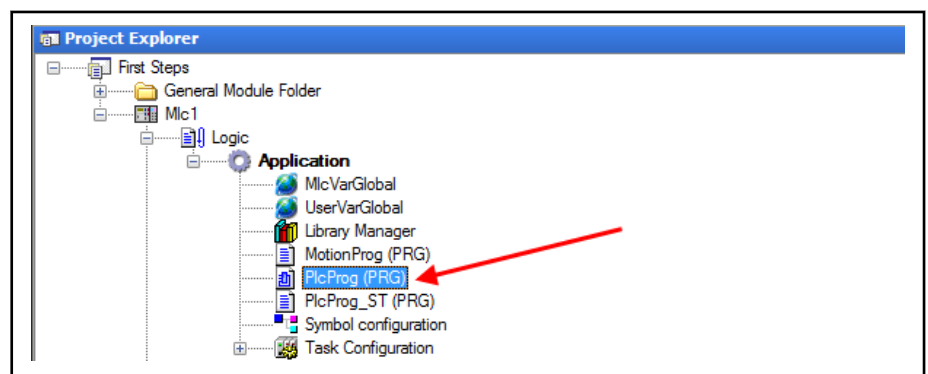


Fig. 2-14: POU "PlcProg" added to application

"PlcProg" task

In "PlcProg", an incoming start signal is coupled to the traffic light phase sequence and the outputs to control each color of the traffic signal are controlled.

First steps - IndraLogic

Adding "SEQUENCE" program in "SFC"

Highlight the "Application" and right-click to open the context menu of the node. Select **Add ► POU** from the context menu.

The "Add object" dialog opens. Make the following settings in this dialog:

1. For the POU, enter the name "SEQUENCE" into the input field "Name:".
2. In the selection area, select "Program" under "Type:". Then, go to "Implementation language:" (programming language) and select "Sequential function chart (SFC)".
3. Press **Finish** to confirm the entries.

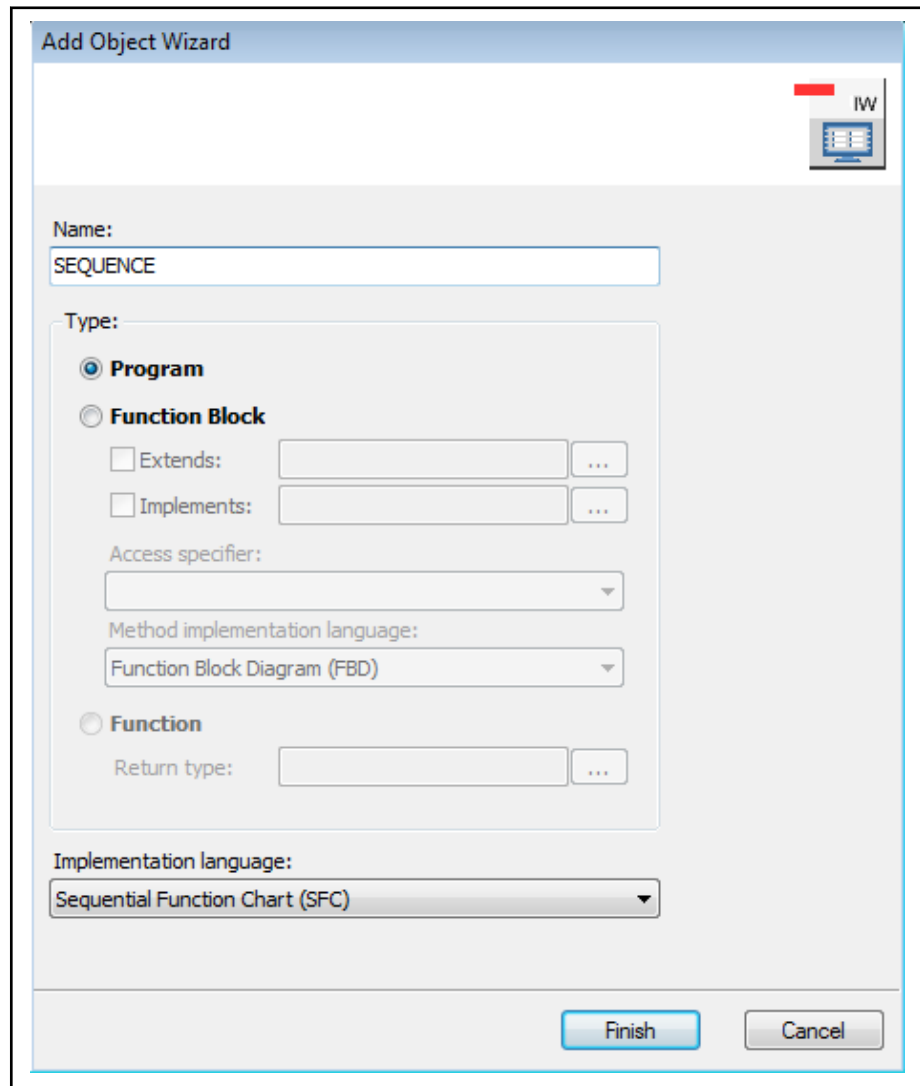


Fig. 2-15: Adding "SEQUENCE" program

The "SEQUENCE" object program is now available in the project.

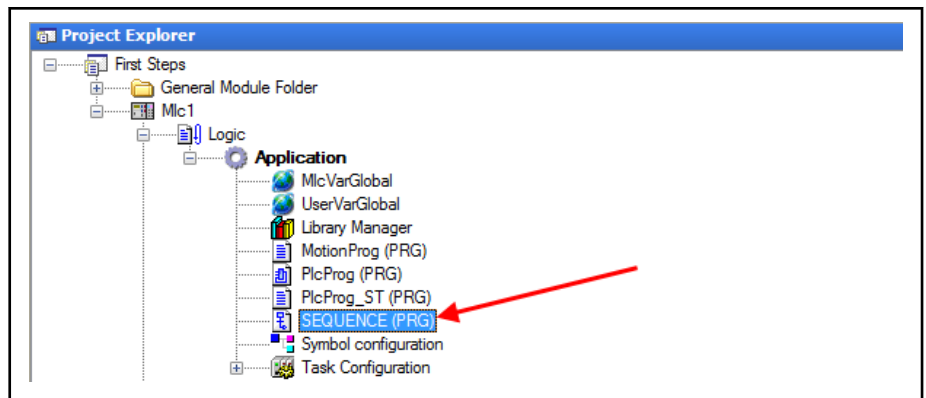


Fig. 2-16: "SEQUENCE" as part of the application

"SEQUENCE" task

The sequence of the control phases of the traffic light is programmed in the "SEQUENCE" program. The "WAIT" function block is called or values are saved for the individual traffic light phases to the "TRAFFICSIGNAL" function block for example to ensure that the correct light is on at the correct point in time for the correct duration.

Adding "TRAFFIC SIGNAL" function block in "FBD"

Highlight the "Application" and right-click to open the context menu of the node. Select **Add ► POU** from the context menu.

The "Add object" dialog opens. Make the following settings in this dialog:

1. For the POU, enter the name "TRAFFICSIGNAL" into the input field "Name:".
2. In the selection area, select "Function block" under "Type:". Then, go to "Implementation language:." (programming language) and select "Function block diagram (FBD)".
3. Press **Finish** to confirm the entries.

The POU "TRAFFICSIGNAL" of type "function block" is displayed in the project tree below the application.

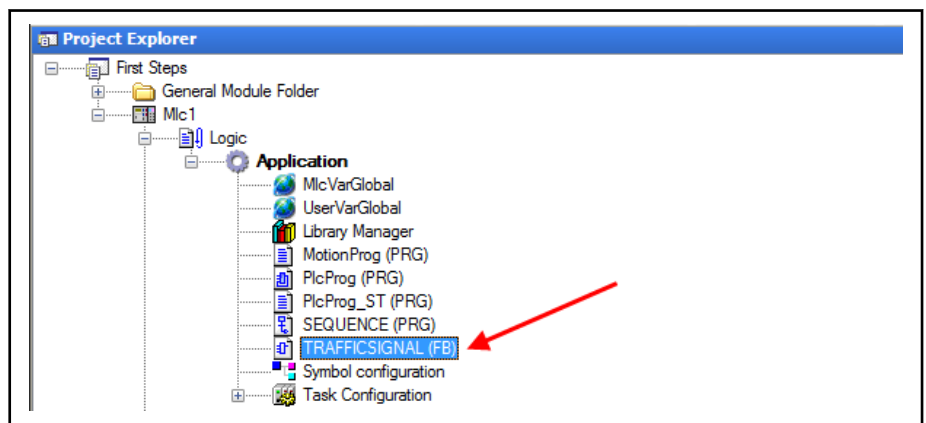


Fig. 2-17: "TRAFFICSIGNAL" function block as part of the application

"TRAFFICSIGNAL" task

The individual traffic light phases are assigned to the traffic light lamps in the "TRAFFICSIGNAL" function block. The red signal light is on during the red or the yellow-red traffic light phase. The yellow lamp is on during the yellow or the yellow-red traffic light phase. The green light is on during the green traffic light phase. Also refer to the following table.

First steps - IndraLogic

Traffic light phase	Red light on	Yellow light on	Green light on
Red	X		
Yellow-red	X	X	
Yellow		X	
Green			X

Tab. 2-1: Traffic light phase assignment

Adding "WAIT" function block in "ST"

The last function block to add is the POU called "WAIT".

Highlight the "Application" and right-click to open the context menu of the node. Select **Add ► POU** from the context menu.

The "Add object" dialog opens. Enter the following settings in the "Add object" dialog:

1. For the POU, enter the name "WAIT" into the input field "Name:".
2. In the selection area, select "Function block" under "Type:". Then, go to "Implementation language:" (programming language) and select "Structured Text (ST)".
3. Press **Finish** to confirm the entries.

The POU "WAIT" POU of type "function block" is now part of your application.

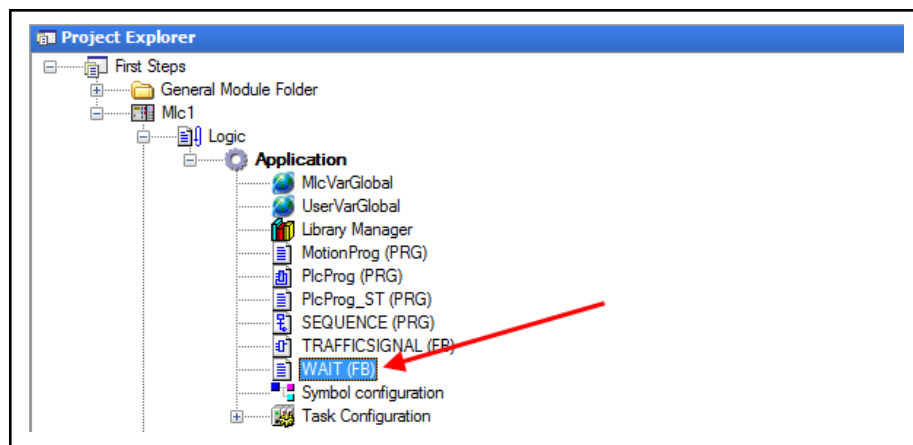


Fig. 2-18: "WAIT" function block as part of the application

"WAIT" task

In the "WAIT" function block, a simple timer stage is programmed using a "TP" function block from the IndraWorks Standard.library. The length of each traffic light phase can be specified using this timer. The input variable for the function block is the duration of the traffic light phase in milliseconds and it returns the Boolean value "TRUE" as output variable after timeout.

2.6.3 Editing programs and function blocks - Part 1

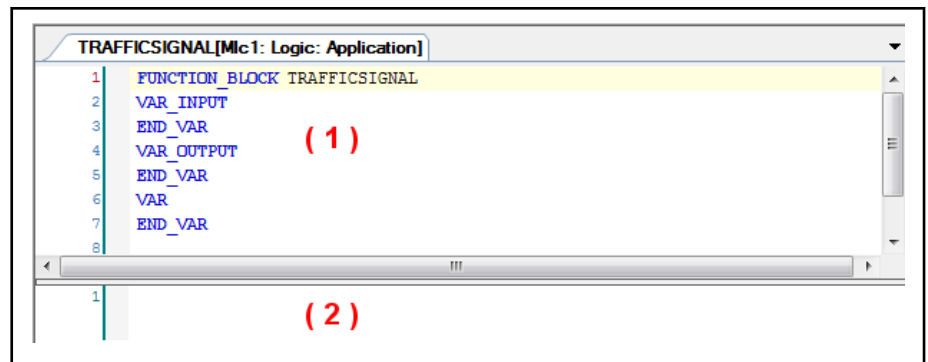
Editing "TRAFFIC SIGNAL" function block in "FBD"

Opening the POU

Select the "TRAFFICSIGNAL" function block using the mouse. Right-click to open the context menu. Select **Open** from the context menu of the "TRAFFICSIGNAL" function block. The editor function block opens (refer to the following figure).



Alternatively, double-click to open the editor of a POU.



- (1) Declaration section
 (2) Implementation section or body

Fig. 2-19: Editor of the "TRAFFICSIGNAL" function block open

POU editor

The editor is divided into two sections:

1. Declaration section
2. Implementation section

The input and output variables of the function block and local variables are created in the declaration section.

The user program is edited in the implementation section.

Declaring variables

There are two options to declare variables in a POU:

1. Create the variables directly in the declaration section.
2. Start with the programming without declaration and supplement data determined by the system in the input window (recommended).

The input window opens automatically if the entered variable is unknown to the system.



1. Create instances of function blocks, of structures (STRUCT) and of data arrays (ARRAY) directly in the declaration section.
2. Variables can be created more easily during programming using the input window.
 Ensure the correct assignment to the validity range and the correct data type.
3. If an "already declared" variable is provided again, it is generally a typing error.

Creating user program

The function block "TRAFFICSIGNAL" is provided with a "STATUS" variable of type "INTEGER" to process the program. The output variables "GREEN", "YELLOW" and "RED" are set to "TRUE" for the corresponding traffic light phases. The "TRAFFICSIGNAL" function block is called later in "PlcProg".

Editing first network

The user program is edited in the implementation section. Click in the editor next to "1" to select the first network in the implementation section. Right-click to open the context menu and select **Insert box**. Alternatively, select **FBD/LD/IL ► Insert box** from the menu.

First steps - IndraLogic

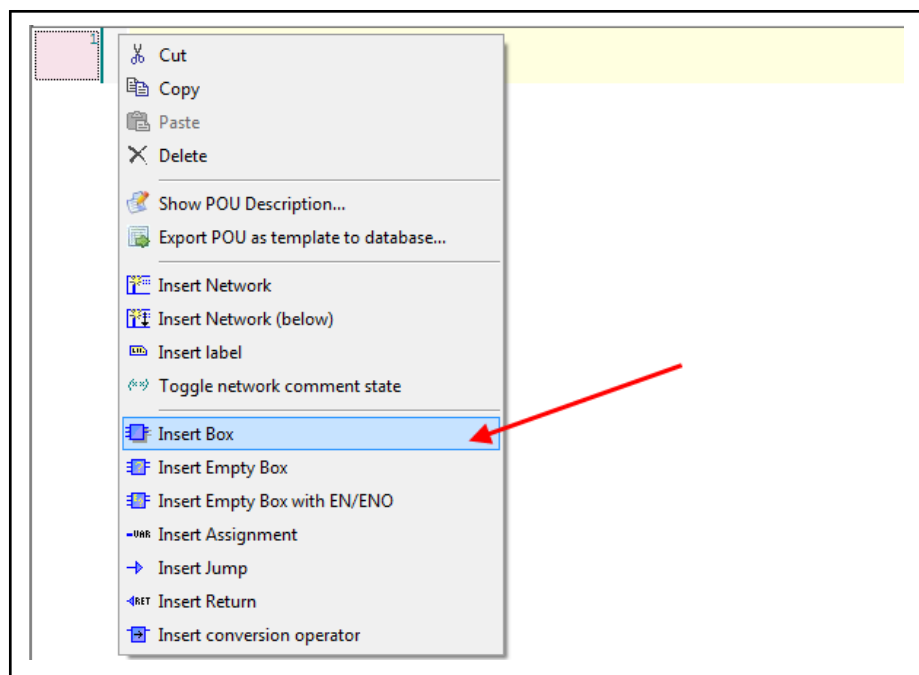


Fig. 2-20: Inserting function block call (Insert box)

The "input assistant" window opens:

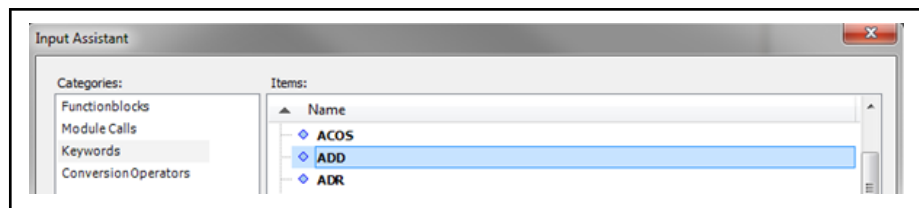


Fig. 2-21: Input assistance – Inserting function block "ADD"

A box with the operator "ADD" and two inputs is to be created in the first network.

1. Select the "Keywords" in "Input assistant" under "Categories:".
2. Select the "ADD" operator under "Elements:".

To confirm the selection, press **OK**.

The statement section looks as follows:

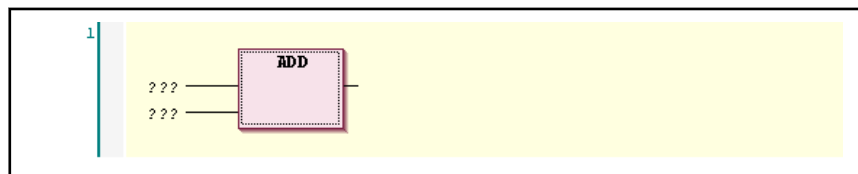


Fig. 2-22: "ADD" function block added to the network

The task is to specify another name and function for the "ADD" function block.

1. Position the cursor over the text "ADD", click to highlight the text and change it to "EQ".
2. Highlight the upper "???" input at the left of the function block called "EQ" and enter the name of the variable "STATUS".

The window to declare the variables opens.

Suggested are: VAR, STATUS, INT.

3. As "STATUS" is an input variable of the function block, change the suggestion to "VAR_INPUT, STATUS, INT".

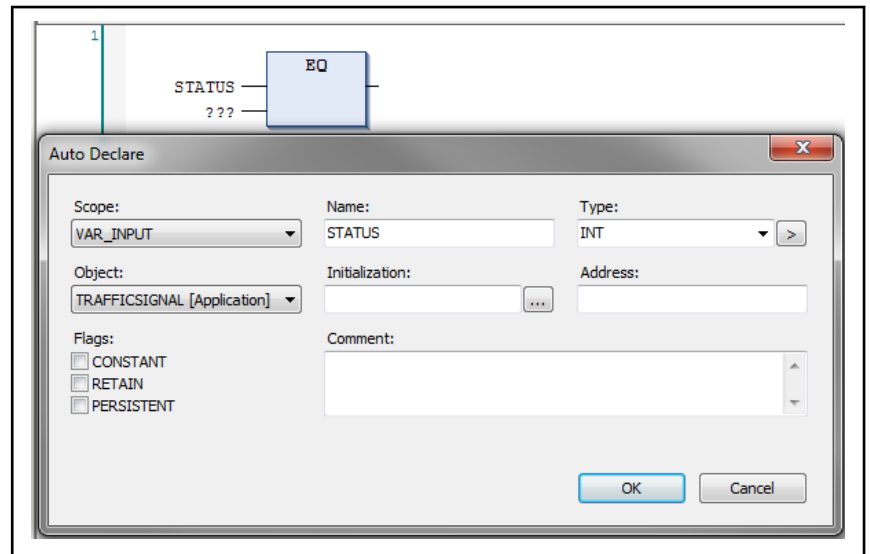


Fig. 2-23: Declaration of the "STATUS" input variables

In the "VAR_OUTPUT...END_VAR" area in the declaration section of the function block, the following declaration line is displayed: STATUS : INT; .

4. At the lower "???" input enter "1".

"STATUS" has four possible states for the different traffic light phases. These states are:

- Green
- Yellow
- Yellow-red
- Red

Decode the four states as output variables. The three following variables are required for the four states:

- GREEN
- YELLOW
- RED

1. Now, in network "1", click in an area following the "EQ" function block to add an assignment to the block. Right-click to open the context menu and select **Insert assignment**. Click on the output of the "???" function block and specify the variable name "GREEN".

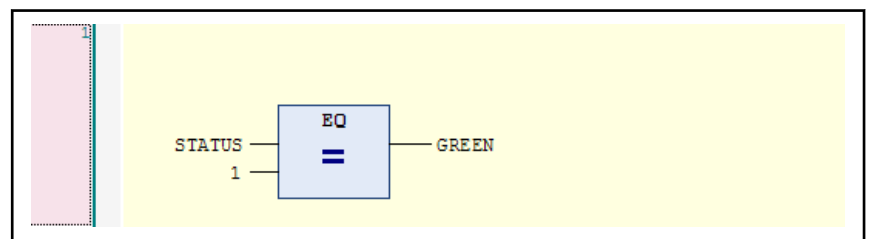


Fig. 2-24: "EQ" function block

First steps - IndraLogic

The window to declare the variables opens.

2. As "GREEN" is an output variable of the function block, change the suggestion to "VAR_OUTPUT, GREEN, BOOL".

In the "VAR_OUTPUT...END_VAR" area in the declaration section of the function block, the following declaration line is displayed: `GREEN: BOOL;`

"EQ" task The "STATUS" input variable is compared with "1". The result is "TRUE" if "STATUS" is "1". The result is "FALSE" if "STATUS" is not equal to "1". The result is assigned to the output variable "GREEN".

Adding network The evaluation of the "STATUS" input variable still has to be performed for the "yellow" and "red" traffic light phases. Therefore, two more networks have to be added.

1. Therefore, click on first the network and select **Add network (below)** via the context menu.

Editing second network Create an "EQ" function block in the second network.

1. Right-click in the second network.
2. Select **Insert box** from the context menu.
3. Select the "Keywords" in the "Input assistant" for "Insert box" under "Categories:".

Select the "EQ" element under "Elements:".

To confirm your entries, press **OK**.
4. Highlight the output of the "EQ" function block and right-click.
5. Select **Insert box** from the context menu.
6. Select the "Keywords" in the "Input assistant" for "Insert box" under "Categories:".

Select "OR" element under "Elements:".

To confirm your entries, press **OK**.

For the further assignment of the "OR" function block, proceed as follows:

1. Highlight the output of the "OR" function block and right-click.
2. At the output, insert **Insert assignment** via the context menu.
3. Enter the name of the output variable "YELLOW" at the output of the function block "???".
4. Complete the entry in the window to declare the "YELLOW" variables.

In the "VAR_OUTPUT...END_VAR" area in the declaration section of the function block, the following declaration line is displayed: `YELLOW: BOOL;`

1. To select the second (free) input of the "OR" function block, directly right-click on the function block on the connection to the right of "???".
2. Right click to select **Insert box** from the context menu.
3. Select the "Keywords" in the "Input assistant" for "Insert box" under "Categories:".

Select the "EQ" element under "Elements:".

To confirm your entries, press **OK**.

The inputs of both "EQ" function blocks still have to be assigned:

1. On the first input of the upper "EQ" function block, click in the second network.
2. Enter "STATUS" and press <Enter>.



As "STATUS" is already declared, the declaration window is not displayed.

3. Click on the second input of the same "EQ" function block.
Enter "2" and press <Enter>.
4. On the first input of the lower "EQ" function block, click in the second network.
Enter "STATUS" and press <Enter>.
5. Click on the second input of the same "EQ" function block.
Enter "4" and press <Enter>.

Copying network

The easiest method to create the third network is to copy the second network.

To copy the network, proceed as follows:

1. Highlight the second network and right-click to open the context menu.
2. Select **Copy** from the context menu.
3. Right-click again to open the context menu.
4. Select **Add**.

The copied network is now added below network "2" with the number "3".

First steps - IndraLogic

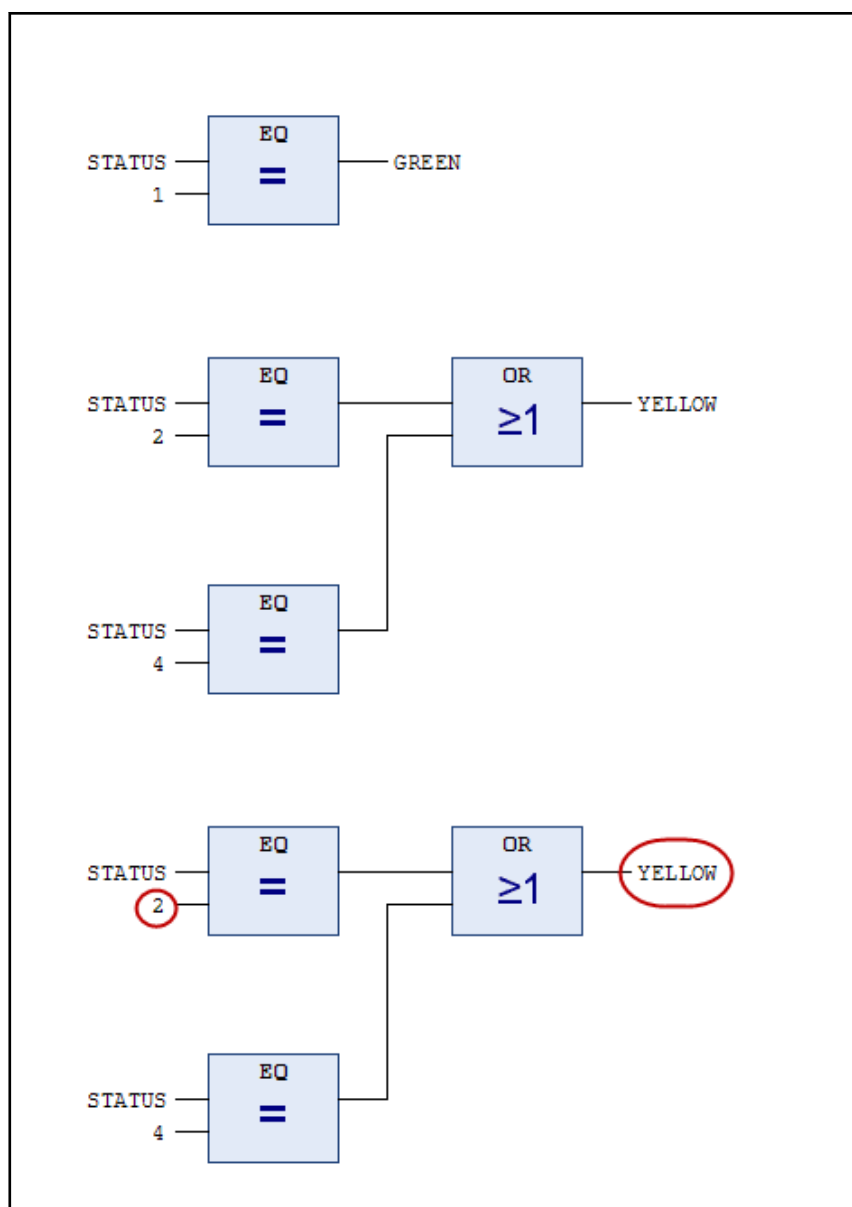


Fig. 2-25: Third network added by copying

Editing third network

To edit the third network, proceed as follows:

1. In network 3 at the upper "EQ function block", select the lower input with "2".
2. Left-click on "2" and enter "3".
3. Now, at the output of the "OR" block, left-click on the variable assignment "YELLOW" and enter "RED".
4. Complete the entry in the window to declare the "RED" variables.

In the "VAR_OUTPUT...END_VAR" area in the declaration section of the function block, the following declaration line is displayed: RED: BOOL;.

The first POU function block "TRAFFICSIGNAL" is completed:

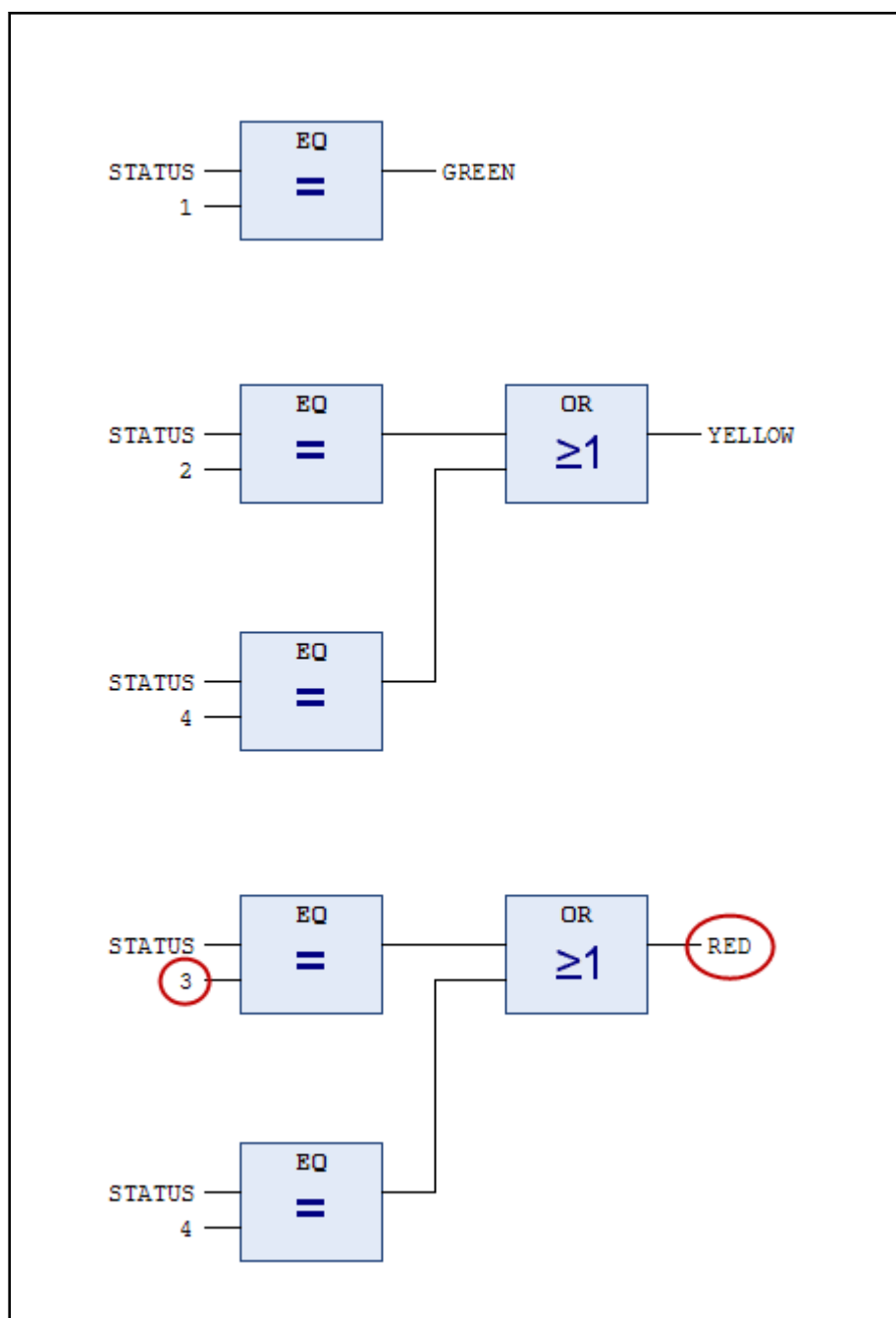


Fig. 2-26: POU "TRAFFICSIGNAL" completed

The declaration section is created incidentally to the implementation:

First steps - IndraLogic

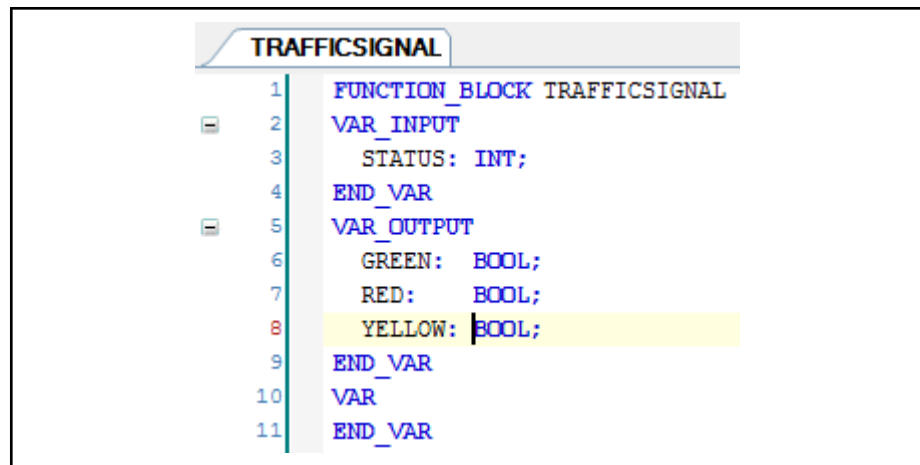


Fig. 2-27: Declaration section of the "TRAFFICSIGNAL" function block

Editing "WAIT" function block in "ST"

Opening the POU

Declaring and pre-assigning variables

To open the POU "WAIT", double-click on the object in the Project Explorer:

The "WAIT" function block is provided with a "WAITINGTIME" variable of type "TIME" as input variable. The "WAIT" function block provides an "OK" variable of data type "BOOL", as an output.

An instance of the function block "TP" `ZAB: TP;` is required.

The current time of the function block is to be written to the variable `ETVar: TIME;`.

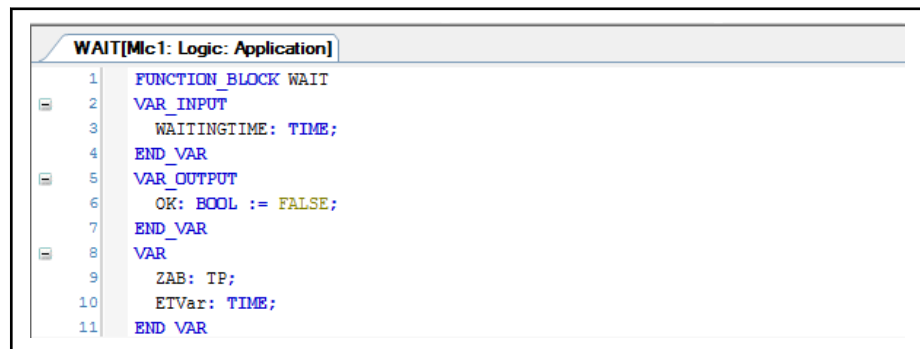


Fig. 2-28: "WAIT" POU variable declaration

"TP" function block from the standard library

The "TP" function block functioning as pulse encoder is located in the IndraWorks standard library. To access the content of the "Standard" library from the library manager, proceed as follows:

1. Double-click in the Project Explorer to open the "library manager".
2. In the "Library manager", highlight the "Standard" library.
3. To open the tree structure of the "Timer" folder, click on the "+" to the left of the "Timer" folder.

The "TP" function block is in the tree. For more information, click on the "TP" function block.

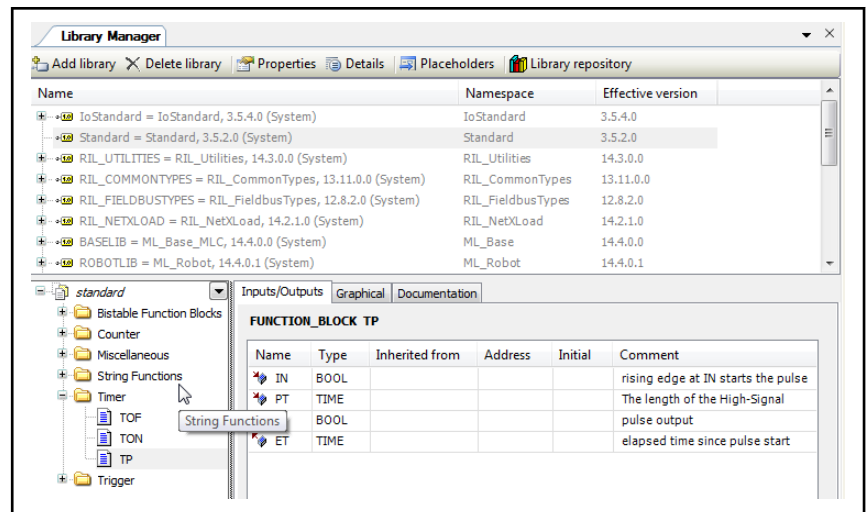


Fig. 2-29: Library manager of the standard function block "TP"

The "TP" function block is provided with two inputs ("IN" and "PT") and two outputs ("Q" and "ET").

"TP" task

As long as "IN" is "FALSE", "ET" = "0" and "Q" is "FALSE".

When "IN" provides the Boolean value "TRUE", the time is added in milliseconds at the "ET" output.

If "ET" reaches the value "PT", the time at the "ET" output is not further added.

"Q" provides "TRUE" as long as "ET" is lower than "PT".

When the value of the variable "PT" is reached, "Q" returns "FALSE".



The outputs of a function block instance can either be evaluated by copying to a variable

ZAB(IN:=TRUE, PT:=WAITINGTIME, ET=>ETVar);

or using the point operator: **ZAB.Q**.

Creating implementation in ST

Complete the implementation section as shown in the following:

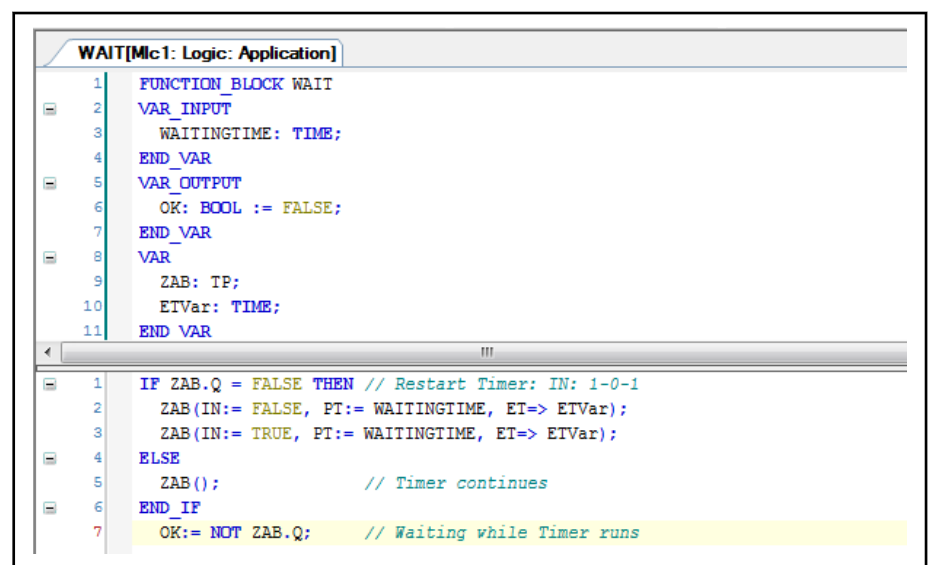


Fig. 2-30: POU "WAIT" completed in ST

First steps - IndraLogic

"WAIT" task If the timer stage elapsed, it is restarted with a 1-0-1 signal change at "IN" and with the current waiting time "WAITINGTIME". Otherwise, the time step continues by updating its outputs.

After the expiry of the timer stage (ZAB.Q = 0), the output "OK" is "TRUE" for one PLC cycle.

Editing "SEQUENCE" program in "SFC"

Opening program To open the POU "SEQUENCE", double-click on the object in the Project Explorer:

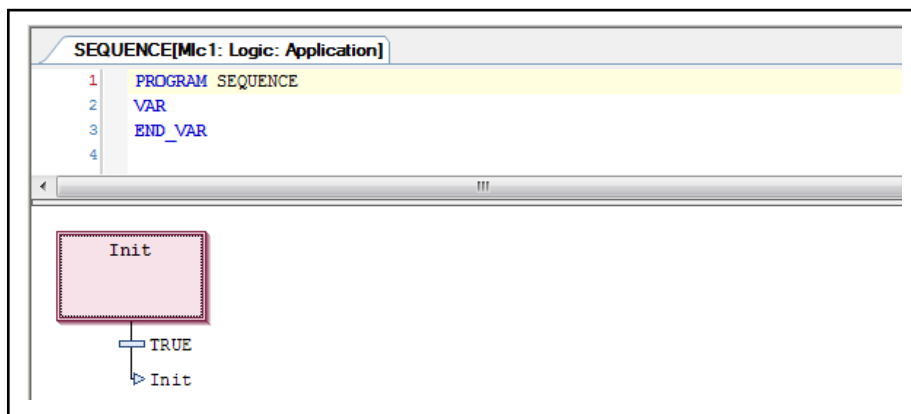


Fig. 2-31: Editor: Program "SEQUENCE" opened

Declaring variables The "SEQUENCE" function block is provided with a "START" variable of type "BOOL" as input variable.

The function block provides the output variables "TRAFFICSIGNAL1" and "TRAFFICSIGNAL2" of type "INT" and as a local variable, the function block has the "DELAY" variable of type "WAIT". First, only the variable `DELAY: WAIT;` is created. The other variables are declared later on using the input assistant in the implementation section.

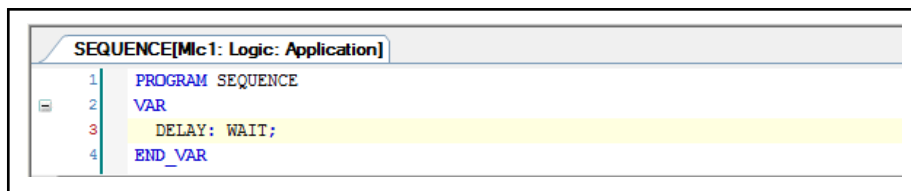


Fig. 2-32: Variable declaration POU "SEQUENCE" POU – First part

Creating user program At the beginning of a sequence order, an "Init" action, a subsequent "TRUE" transition and a return jump to "Init" are located in the body of the "SFC editor" (see [fig. 2-31 "Editor: Program SEQUENCE opened" on page 40](#)).

Specifying the structure of the sequence order

First, the structure of the sequence order is specified by adding the individual steps and transitions. After adding the necessary steps and transitions, the individual actions are programmed for the steps and transitions. One step is required for each traffic light phase. To add a step with the subsequent transition, proceed as follows:

1. Highlight the "TRUE" transition and right-click to open the context menu.
2. Select **Add step transition afterwards** from the context menu.
3. Highlight the last transition each and add three more steps and transitions.

The sequence order is as follows:

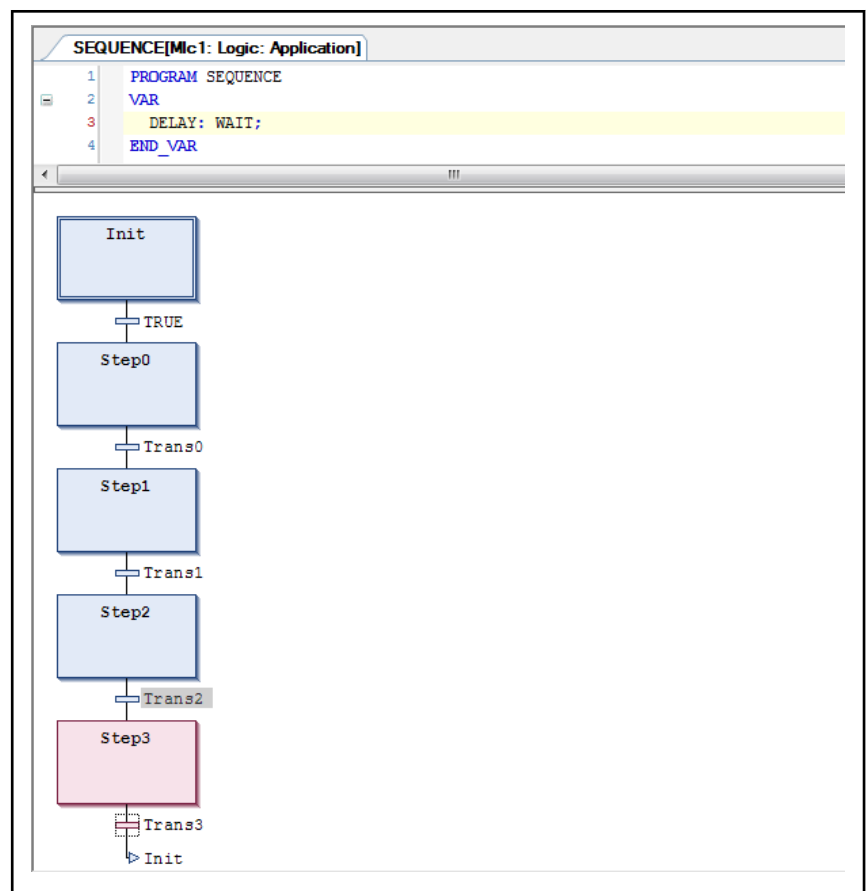


Fig. 2-33: Steps added to "SEQUENCE" program



To alternatively add steps and transitions, go to the menu **SFC ► Add step transition afterward**.

Programming transitions

There are two options to program transitions:

1. Changing the transition name: The change is only useful if only one advance condition is required, e.g. when querying a variable.
2. Programming the transition: Programming is useful if the result of linked variables is intended as an advance condition.

To program a transition, add a transition. Double-click on "transition", specify a name for the transition and the implementation language (programming language) used to program the transition. Subsequently, the transition can be programmed - like a function - in an editor.

Declare the "START" transition as first transition. To declare the "START" transition, proceed as follows:

1. Click on the transition name "TRUE" of the first transition.
2. Enter "START" into the input field.

Since the "START" variable has not yet been declared in the declaration section, a window opens to declare variables.

First steps - IndraLogic

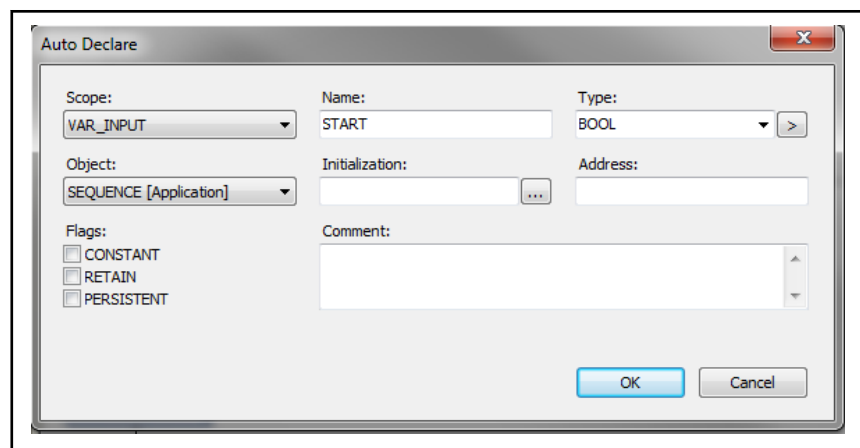


Fig. 2-34: Input assistance for variable declaration

3. Select the data type "BOOL" in "Data type:".
4. For "Scope:", select the variable type "VAR_INPUT".
5. To confirm your entries, press **OK**.

The "START" variable is assigned to the first transition. If "START" = "TRUE", the next action is performed. The advance condition is fulfilled.

For the second and all subsequent transitions, proceed as follows:

1. Click on the transition name.
2. Change transition name to `DELAY.OK`.

The other transitions are fulfilled, when the "OK" output variable of "DELAY" or of the function block "WAIT" is "TRUE".

The sequence order is as follows:

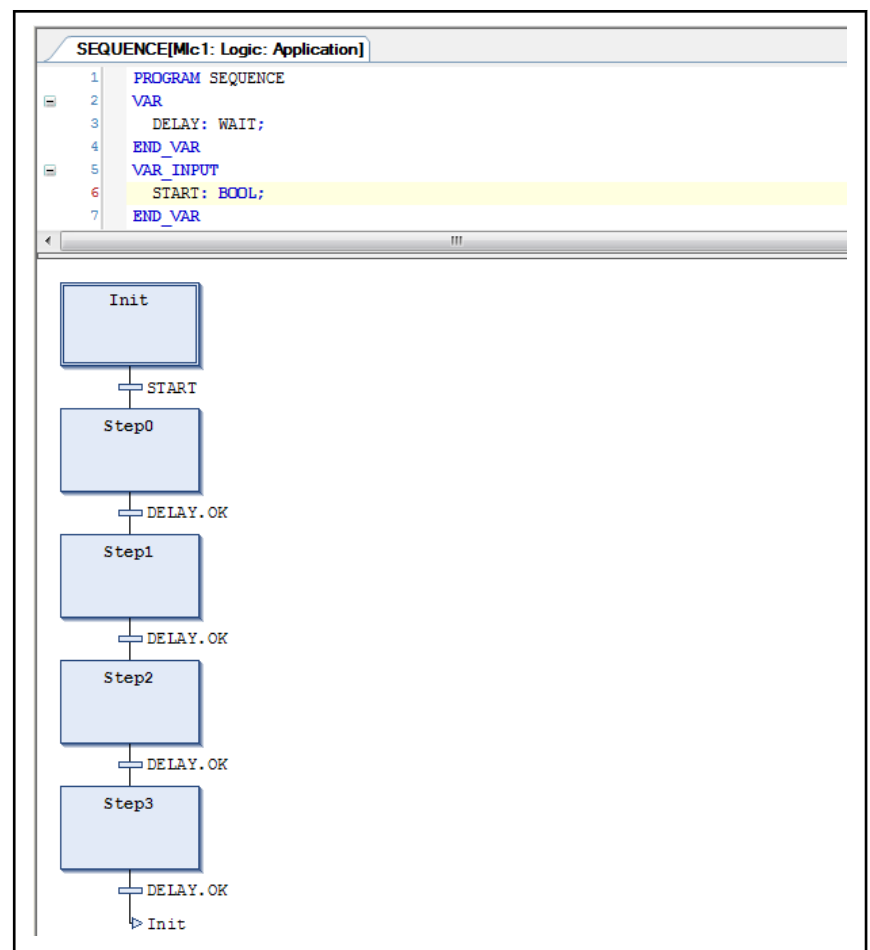


Fig. 2-35: Steps of the sequence order created

Naming the steps Only the "Init" step keeps its name. The remaining steps are renamed. The steps should be named "Switch1", "Green2", "Switch2" and "Green1" consecutively.

To rename the individual steps, proceed as follows:

1. Click on the step name.
2. Enter the desired name.
3. To confirm your entries, press <Enter>.

Step function The yellow phases are implemented in the steps "Switch1" and "Switch2".

In the step "Green1", the green phase for traffic signal 1 is implemented and in "Green2", the green phase for traffic signal 2.

Changing the return jump address After successful step processing, there should be no return jump to "Init" anymore, but instead, a jump to the step "Switch1". Click on the name of the return jump "Init" and enter the new return jump target "Switch1".

The exact procedure repeats continuously.

The sequence order is as follows:

First steps - IndraLogic

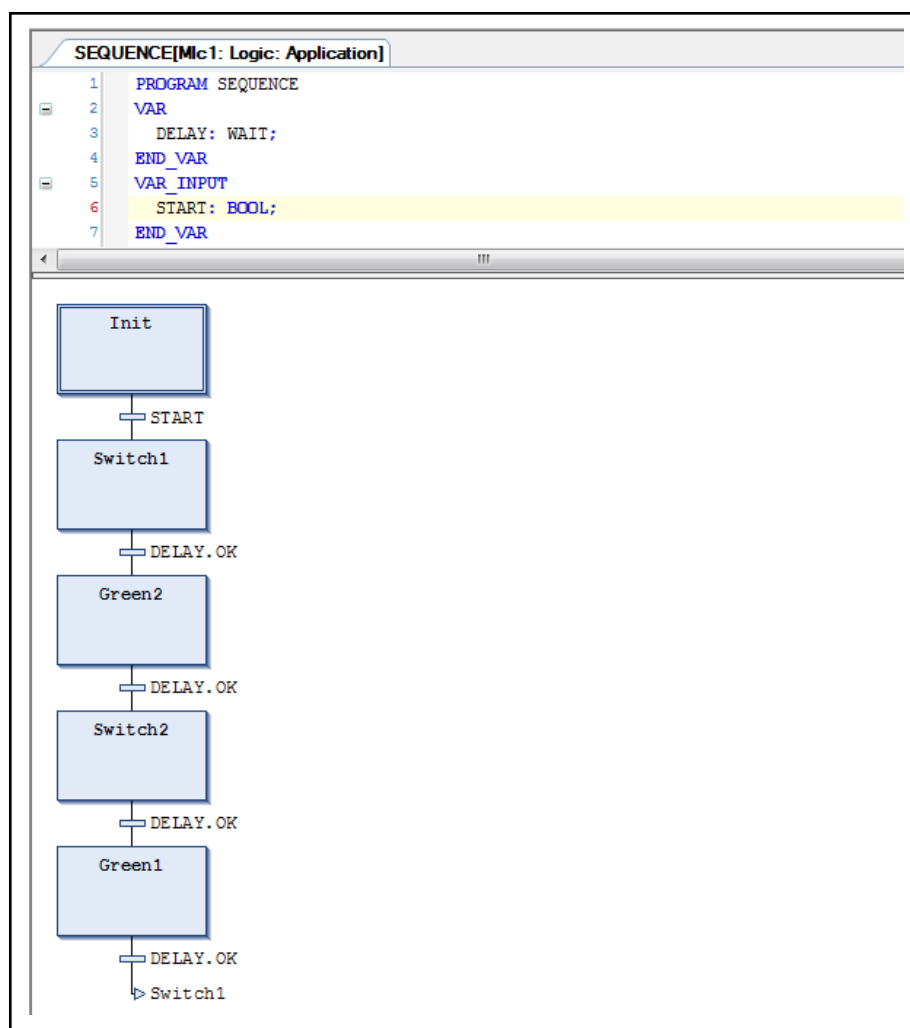


Fig. 2-36: Steps of the sequence order named

Steps and actions

A function block programmed in Sequential Function Chart (SFC) consists of a sequence of interconnected steps via directed transitions. Actions are assigned to the steps, in which the program code to be processed, is stored.

IndraLogic supports the following action types:

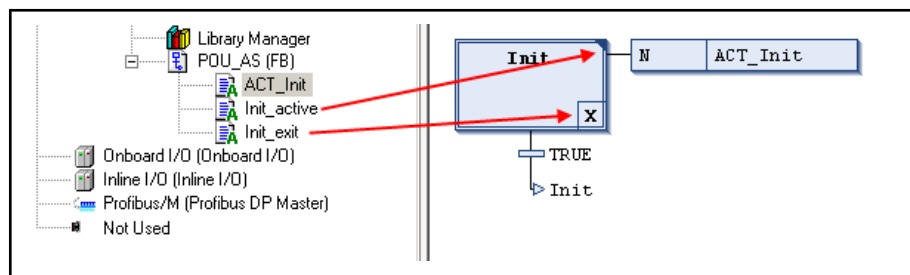


Fig. 2-37: Types of step actions

- **Input action:** An input action is processed once when the step is enabled. In this case: "Init_active".

A step with an input action is identified by a black triangle in the upper right corner

- **Associated action:** The action is created separately as part of the POU and is connected associatively with the required steps. In this case: "ACT_Init".

The time behavior is determined by the determination symbol (in this case: N). The associated actions are displayed to the right of the step. If such an action is disabled, it is post-processed exactly once

- **Output action:** An output action is only executed once before the step is disabled. In this case: "Init_exit".

A step with an output action is identified by an "X" in the lower right corner

Programming "Init" action with "ST"

In the action for the step "Init", the "TRAFFICSIGNAL1" and "TRAFFICSIGNAL2" variables are initialized. The status of "TRAFFICSIGNAL1" for traffic signal 1 should be "1" (green). The status of "TRAFFICSIGNAL2" for traffic signal 2 should be "3" (red).

Proceed as follows to edit "Init":

1. Double-click on the box of the first "Init" step in the "SEQUENCE" program.

An "Add action" window opens.

2. Select "ST" as implementation language (programming language).
3. To confirm the settings, press **Open**.

A branch is generated with the action "Init_active" in the Project Explorer under the "SEQUENCE (FB)" POU.

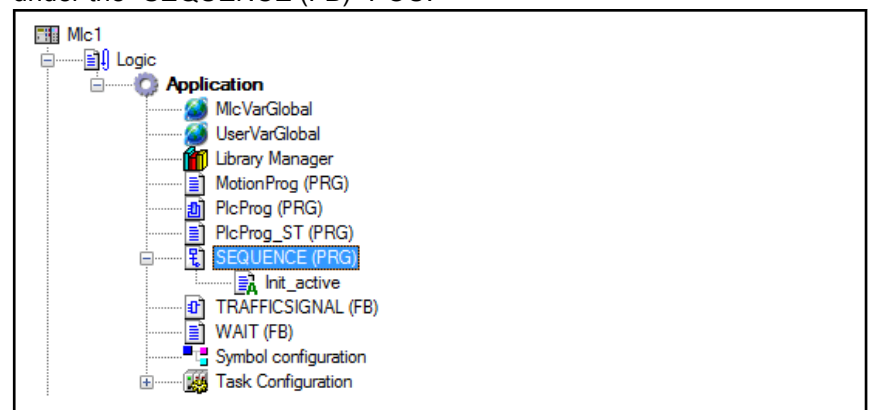


Fig. 2-38: POU "SEQUENCE" in the Project Explorer with "Init_active" action

The editor of the "Init_active" action opens.

4. Enter the following program lines into the editor:

"Init_active" action

```
TRAFFICSIGNAL1 := 1;
TRAFFICSIGNAL2 := 3;
```



When programming the "Init_active" action, the variables "TRAFFICSIGNAL1" and "TRAFFICSIGNAL2" are not yet known. The "Declaring variables" window opens automatically. For both variables, select the scope "VAR_OUTPUT" and the data type "INT".

Programming "Switch1" action with "ST"

In the action for the step "Switch1" the STATUS of TRAFFICSIGNAL1 changes to "2" (yellow) and the STATUS for TRAFFICSIGNAL2 to "4" (yellow-red). A delay time of two seconds is specified.

First steps - IndraLogic

To edit "Switch1", proceed as follows:

1. Double-click on the box of the second step, "Switch1" in the "SEQUENCE" POU.

The "Add action" window opens.

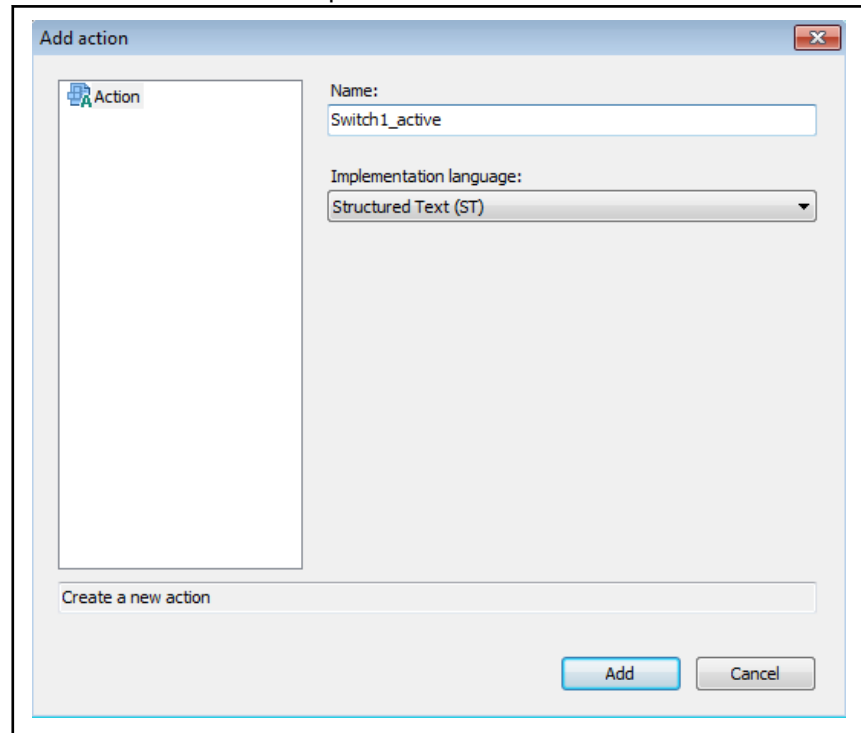


Fig. 2-39: Adding action

2. Select "Structured Text (ST)" as implementation language (programming language).
3. To confirm your entries, press **Add**.

A branch is generated with the action "Switch1_active" in the Project Explorer under the "SEQUENCE (FB)" POU. The editor of the "Switch1_active" action opens.

4. Enter the following program lines into the editor:

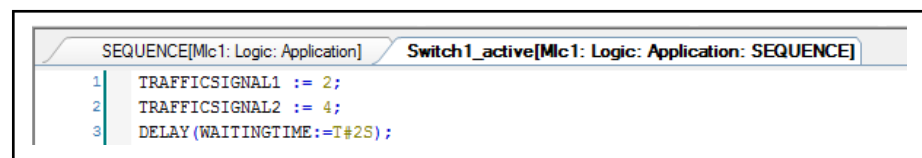


Fig. 2-40: Editor of the "Switch1_active" action

Programming "Green2" action with "ST"

In the action for the step "Green2" the STATUS of TRAFFICSIGNAL1 changes to "3" (red) and the STATUS for TRAFFICSIGNAL2 to "1" (green). A delay time of five seconds is specified.

To edit "Green2", proceed as follows:

1. Double-click on the box of the third step "Green2" in the POU "SEQUENCE".

The "Add action" window opens.

2. Select "Structured Text (ST)" as implementation language (programming language).

3. To confirm the entry, press **Open**.

A branch is generated with the action "Green2_active" in the Project Explorer under the "SEQUENCE (FB)" POU.

The editor of the "Green2_active" action opens.

4. Enter the following program lines into the editor:

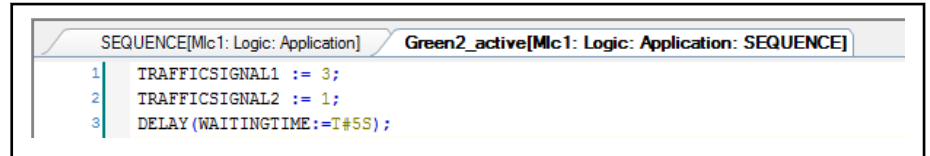


Fig. 2-41: Editor of the "Green2_active" action

Programming "Switch2" action with "ST"

In the action for the step "Switch2" the STATUS of TRAFFICSIGNAL1 changes to "4" (yellow-red) and the STATUS for TRAFFICSIGNAL2 to "2" (yellow). A delay time of two seconds is specified.

To edit "Switch2", proceed as follows:

1. Double-click on the box of the fourth step, "Switch2" in the "SEQUENCE" POU.

An "Add action" window opens.

2. Select "Structured Text (ST)" as implementation language (programming language).

3. To confirm the entry, press **Open**.

A branch is generated with the action "Switch2_active" in the Project Explorer under the "SEQUENCE (FB)" POU.

The editor of the "Switch2_active" action opens.

4. Enter the following program lines into the editor:

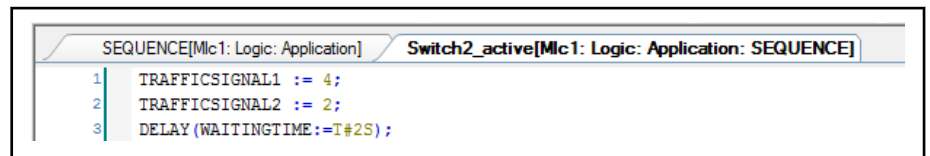


Fig. 2-42: Editor of the "Switch2_active" action

Programming "Green1" action with "ST"

In the action for the step "Green1" the STATUS of TRAFFICSIGNAL 1 changes to "1" (green) and the STATUS for TRAFFIC SIGNAL 2 to "3" (red). A delay time of five seconds is specified.

To edit "Green1", proceed as follows:

1. Double-click on the box of fifth step "Green1".

An "Add action" window opens.

2. Select "Structured Text (ST)" as implementation language (programming language).

3. To confirm the entry, press **Open**.

A branch is generated with the action "Green1_active" in the Project Explorer under the "SEQUENCE (FB)" POU.

The editor of the "Green1_active" action opens.

4. Enter the following program lines into the editor:

First steps - IndraLogic

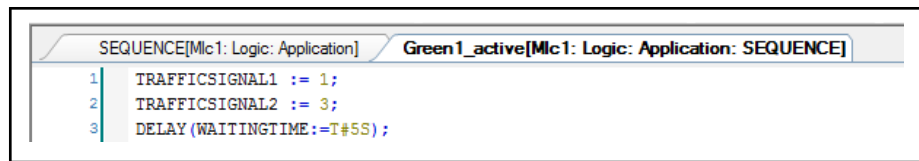


Fig. 2-43: Editor of the "Green1_active" action

Completed "SEQUENCE" function block

The "SEQUENCE" function block and the structure in the Project Explorer is now complete:

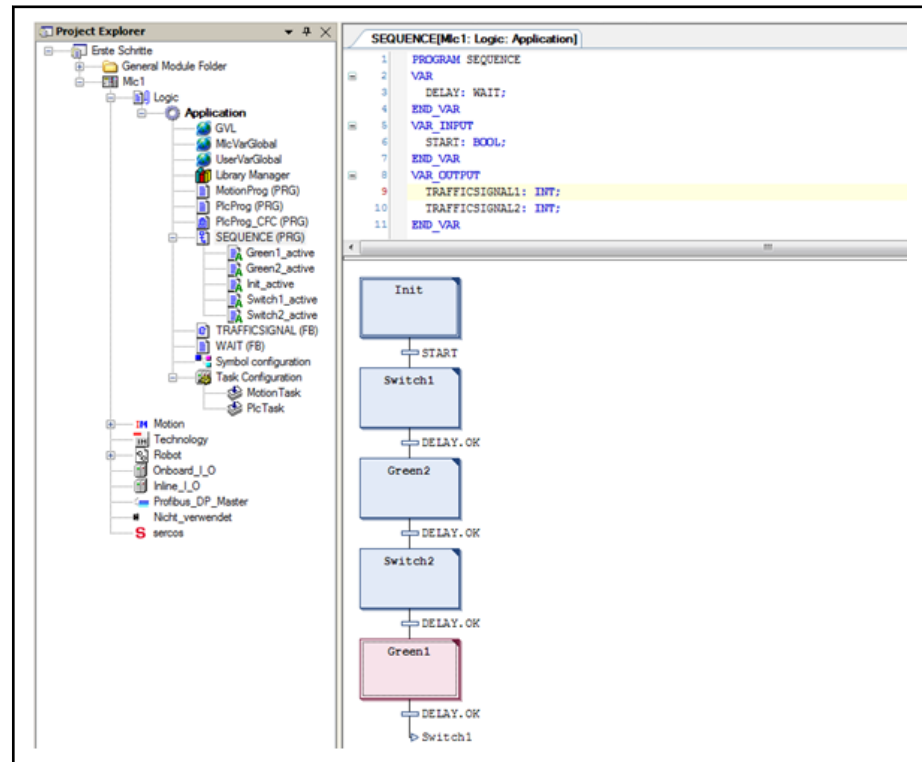


Fig. 2-44: POU "SEQUENCE" completed

2.6.4 Task configuration

Task A "task" is a job to be carried out. The task is to call programs by declaring "task types" such as "cyclic" or "free running" and by assigning runtimes. The program sequence is organized using a "task".

Task configurator The task configurator defines one or multiple tasks that check the execution of an application.

Adding "SEQUENCE" task There are two POUs of type "program" in the application: "PlcProg" and "SEQUENCE". "PlcProg" is automatically created when a control is added. A "PlcProg" task is also created when a control is added so that the "PlcProg" POU can be sequenced. Another task for "SEQUENCE" is added to call the POU "SEQUENCE".

To add the "SEQUENCE" task, proceed as follows:

1. In the Project Explorer, double-click to open the "PlcTask" configuration below "Task configuration".
2. In the "POUs" section, select the open configuration "Add call".
The "input assistant" window opens automatically.
3. Open the tree at "Mlc1" in the "Input help" under "Elements:" until "SEQUENCE" and "PlcProg" are visible.

4. Select "SEQUENCE" and press **OK**.

The "Plc task configuration" now looks as follows:

Fig. 2-45: Task configuration

2.6.5 Declaring global variables

GVL function Switch on the traffic light system with the "ON" switch.

For the two traffic lights, the individual lamps of the traffic light are controlled with regard to the program state of the respective traffic light phase.

Input and output variables have to be created for the one input and the six outputs for both traffic lights.

Access the physical addresses for the inputs and outputs using the syntax "%IX" for an input and "%QX" for an output.

The size prefix "X" describes the range in which the addressed are to be accessed. In this example, only bit addresses are accessed.

The variables should not only be available locally in the "PlcProg" function block, but also in other function blocks. Thus, the variables are not declared in the POU "PlcProg", but in a Global Variable List "GVL" instead.

Creating "Global Variable List (GVL)"

Therefore, add a "GVL" to the project (context menu of "Application", **Add... ► Global Variable List...**).



If the Global Variable List (GVL) is created under "Application", the variables are available to all POUs in the application.

Declaring variables with addresses

Create the following variables of type "BOOL" in the "Global Variable List":
Global Variable List "GVL"

```
VAR_GLOBAL
  ON : BOOL;
  A1_green AT %QX0.0 : BOOL;
  A1_yellow AT %QX0.1 : BOOL;
  A1_red AT %QX0.2 : BOOL;
  A2_green AT %QX0.3 : BOOL;
  A2_yellow AT %QX0.4 : BOOL;
  A2_red AT %QX0.5 : BOOL;
END_VAR
```

The name of the variable follows the "AT" assignment with a percent sign before the range prefix, with "Q" for outputs and "I" for inputs. The size prefix with "X" for bit addressing follows.

First steps - IndraLogic

A1_green AT %QX0.0: BOOL assigns the variable "A1_green" of the data type "BOOL" to the physical output bit "Q 0.0".



When using a real control, check that these outputs really exist.

The controls IndraMotion MLC L45/L65/L75 are provided with these outputs as onboard outputs (default mapping).

The controls IndraMotion MLC L25, IndraMotion MLC XM2 and IndraMotion MLC VPx are not provided with any onboard outputs. Thus, extend this control with Inline output terminals (only IndraMotion MLC L25) or a field bus with I/O modules. If this is not the case, do not use the absolute addressing or a warning is output when generating the code.

2.6.6 Testing "SEQUENCE" in simulation mode

The "SEQUENCE" program function block is completed to a point that the program function block can be tested. Test in simulation mode.

Compiling "Application"

To load the application to the control, the "compiler" has to compile the application first without errors. To compile the application, select **Create ► Build** from the menu. After compilation, the following is displayed in the "Task list" below the workspace: "Compiling completed -- 0 errors, 0 warnings".

Simulating a control

If no control is present for testing, simulate a control. To simulate a control, click on "Mlc1" in the "Project Explorer" and right-click to open the context menu. Click on **Simulation** to enable the simulation mode. If the simulation is active, "Simulation" is selected.

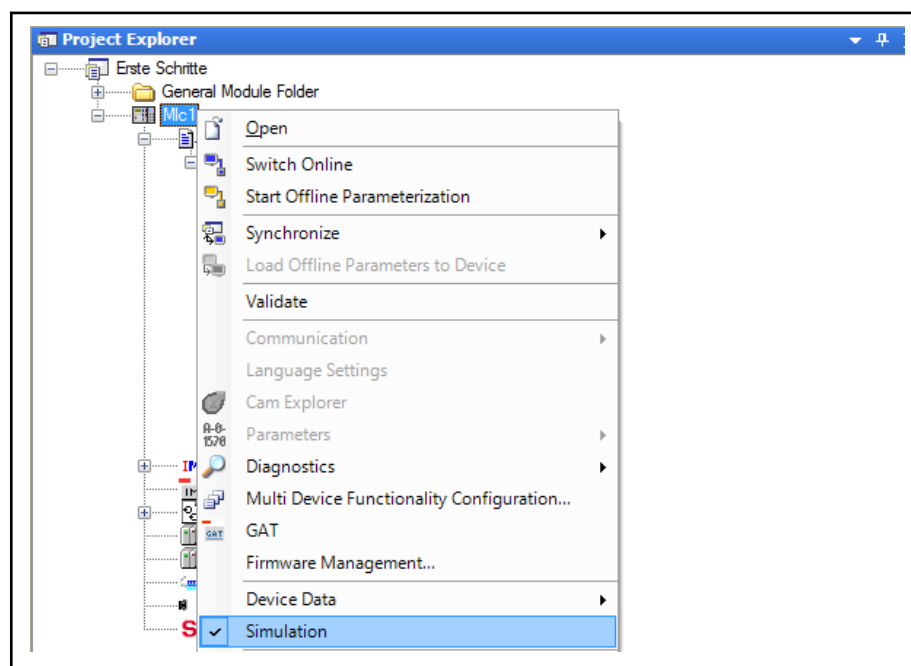


Fig. 2-46: Selecting "simulation"

Logging into the control

Now log into the control and load the application to the control. To log into the control, select **Debug ► Login** from the IndraWorks menu bar.

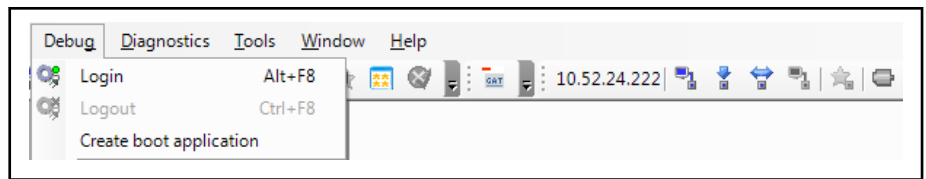


Fig. 2-47: Logging into the control

The Project Explorer looks as follows:

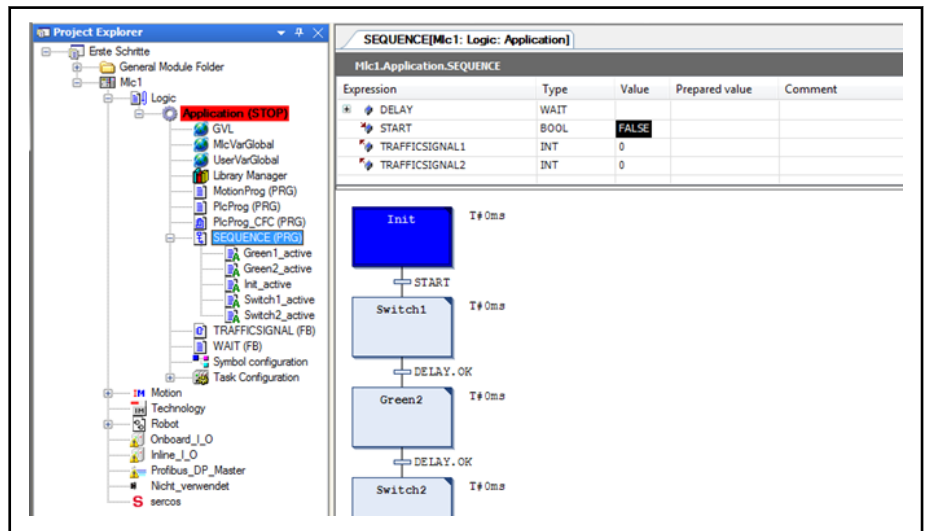


Fig. 2-48: Logged into control (simulation) – Application in "STOP"

Starting application

To start the application, right-click on the application and select **Start** from the context menu or **Debug ► Start** from the IndraWorks menu bar.

The application is now running on the control.

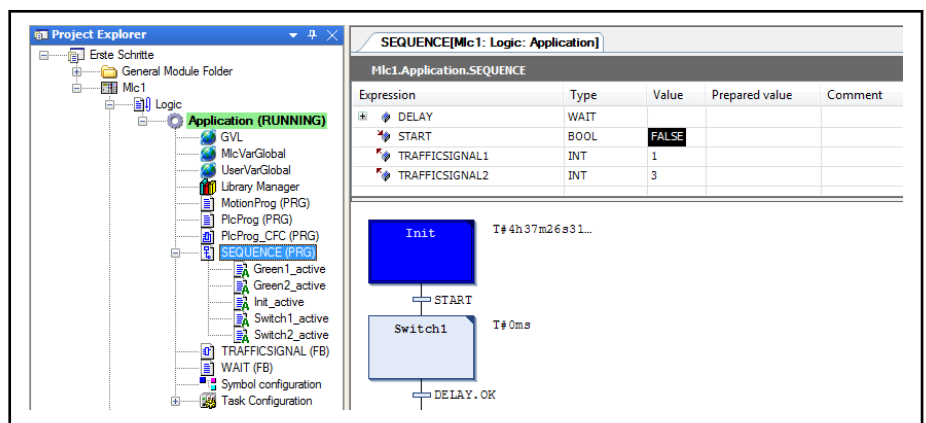


Fig. 2-49: Logged into the control - Application running

Starting "SEQUENCE"

Open the POU "SEQUENCE". The variables and their current values can be seen in the online mode of the declaration section. Click in the declaration section in the line at the "START" variable in the "Prepared value" column until "TRUE" is displayed.

First steps - IndraLogic

SEQUENCE[Mlc1: Logic: Application]				
Mlc1.Application.SEQUENCE				
Expression	Type	Value	Prepared value	Comment
DELAY	WAIT			
START	BOOL	FALSE	TRUE	
TRAFFICSIGNAL1	INT	0		
TRAFFICSIGNAL2	INT	0		

Fig. 2-50: Declaration section POU "SEQUENCE" online

Select **Debug ► Write values** (<Ctrl>+<F7>) in the IndraWorks menu bar.

The "START" variable is set to "TRUE".

Mlc1.Application.SEQUENCE				
Expression	Type	Value	Prepared value	Comment
DELAY	WAIT			
WAITINGTIME	TIME	T#5s		
OK	BOOL	FALSE		
ZAB	TP			
ETVar	TIME	T#0ms		
START	BOOL	TRUE		
TRAFFICSIGNAL1	INT	3		
TRAFFICSIGNAL2	INT	1		

Fig. 2-51: Declaration section POU "SEQUENCE" online

The individual steps and transitions of the POU "SEQUENCE" are processed. At the end, there is a return jump to the "Switch1" step.

The values for the "START" variable are set later in the POU "PlcProg".

Logging out of the control

Before logging out of the control, stop the application. To stop the application, right-click on the application and select **Stop** from the context menu or **Debug ► Stop** from the IndraWorks menu bar.

The application is stopped on the control.

Log out by selecting **Debug ► Logout from** in the IndraWorks menu bar.

IndraWorks is in offline mode.



Upon next login, the values at the time of the last logout are used.

It is possible to start with a specified values if old data is deleted prior to login (IndraWorks menu bar **Create ► Clean**).

2.6.7 Editing programs and function blocks – Part 2

Editing "PlcProg" program in "ST"



The "PlcProg" program can be implemented in two programming languages:

- For programming in "Continuous Function Chart (CFC)", refer to [chapter "Editing PlcProg program in CFC" on page 55](#)
- For programming in "Structured Text (ST)", continue in the text

"PlcProg" task

The temporal sequence of the traffic light phases for both traffic lights was implemented in the "SEQUENCE" function block. The inputs and outputs of a PLC are now to be controlled and evaluated. Input and output variables for

the one input and the six outputs for the two traffic lights were created in the "Global Variable List (GVL)". These variables can now be used in the "PlcProg" program. The application was previously tested only in simulation mode. The application can now also be tested in a real control.

Declaring an instance

The "PlcProg" program receives a "LIGHT1" variable of type "TRAFFICSIGNAL", and a "LIGHT2" variable, also of type "TRAFFICSIGNAL".

This instantiates the "TRAFFICSIGNAL" function block.

To create the variable, refer to ["Declaring variables" on page 31](#).

Create the variables `LIGHT1: TRAFFICSIGNAL;`

and `LIGHT2: TRAFFICSIGNAL;;`

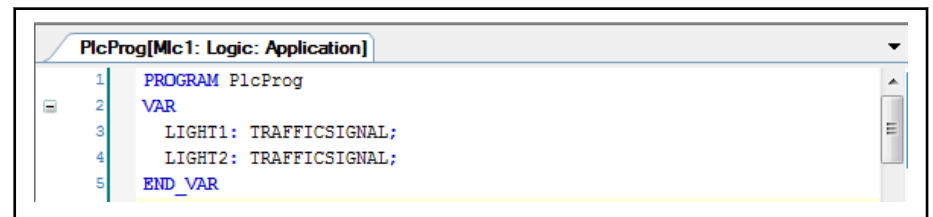


Fig. 2-52: Declaration section POU "PlcProg"

Editing body

The "PlcProg" program is edited in the "ST" programming language. The first function block to call is the "SEQUENCE" function block. At the "START" input of the "SEQUENCE" program, the global variable "ON" is created.

The two function block instances "LIGHT1" and "LIGHT2" of type "TRAFFICSIGNAL" are called in the following.

The output "TRAFFICSIGNAL1" of the "SEQUENCE" program is created at the "STATUS" input of "LIGHT1". The three outputs "GREEN", "YELLOW" and "RED" of the "LIGHT1" instance are written to the global variables "A1_green", "A1_yellow" and "A1_red".

Proceed with the function block instance "LIGHT2" as with "LIGHT1".



"A1_green" is correctly called "GVL.A1_green" or ".A1_green" for short, and so on, since these are global variables.

However, if there is no locally declared "A1_green", the system uses the global variable of the same name as substitute.

The "PlcProg" program is as follows:

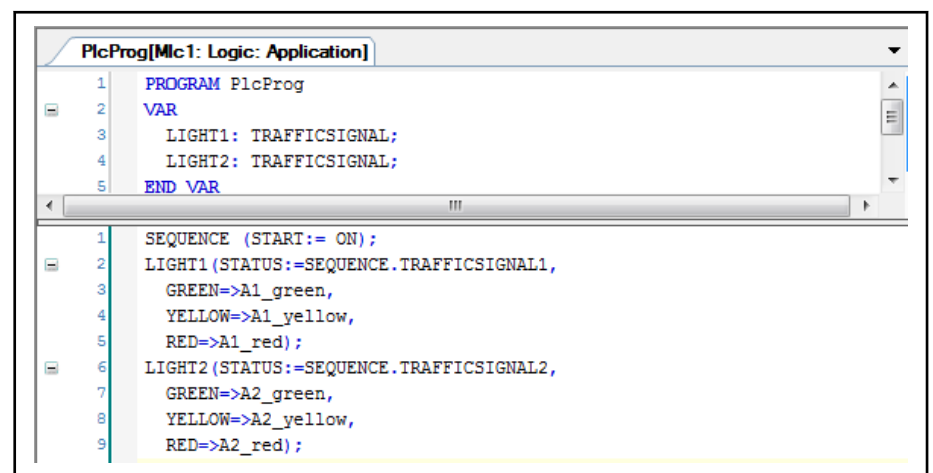


Fig. 2-53: "PlcProg" program completed

Testing the application in simulation mode, PlcProg in ST

Build application To load the application to the control, the "compiler" has to compile the application first without errors. To compile the application, select **Create ► Build** from the menu. After compilation, the following is displayed in the "Task list" below the workspace: "Compiling completed -- 0 errors, 0 warnings".

 Only the currently active application is compiled.
If **multiple** applications are **available**, the application to be compiled has to be set to active (context menu "Application" **Set active application**).

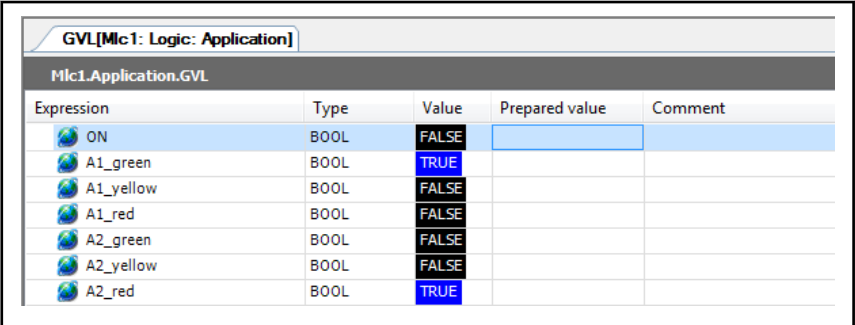
Simulating a control If there is no control for testing, simulate one. To simulate a control, click on "Mlc1" in the Project Explorer and right-click to open the context menu. Click on **Simulation** to activate the simulation mode. If the simulation is active, "Simulation" is selected.

Logging into the control Now log into the control and load the application to the control. To log into the control, select **Debug ► Login** from the IndraWorks menu bar.
The application is then in the "STOP" state.

Starting application To start the application, right-click on the application and select **Start** from the context menu. The application is now running as simulation.

Switching on the traffic light and observing the traffic light cycle After the application is started, "LIGHT1" is in the state "Green" and "LIGHT2" is in the state "red". The traffic light has to be switched on so that the traffic light cycle can be run through. To switch on the traffic light, proceed as follows:

1. To open the "Global Variable List", double-click on "GVL" in the Project Explorer.



GVL[Mlc1: Logic: Application]				
Mlc1.Application.GVL				
Expression	Type	Value	Prepared value	Comment
 ON	BOOL	FALSE		
 A1_green	BOOL	TRUE		
 A1_yellow	BOOL	FALSE		
 A1_red	BOOL	FALSE		
 A2_green	BOOL	FALSE		
 A2_yellow	BOOL	FALSE		
 A2_red	BOOL	TRUE		

Fig. 2-54: "GVL" online

2. At the "ON" variable, click into the "Prepared value" column so that "TRUE" is displayed in this column.
3. To write the value to the control, click on **Debug ► Write values** in the IndraWorks menu bar.

The traffic light cycle is now run. The traffic light cycle can be observed in the "PlcProg" function block.

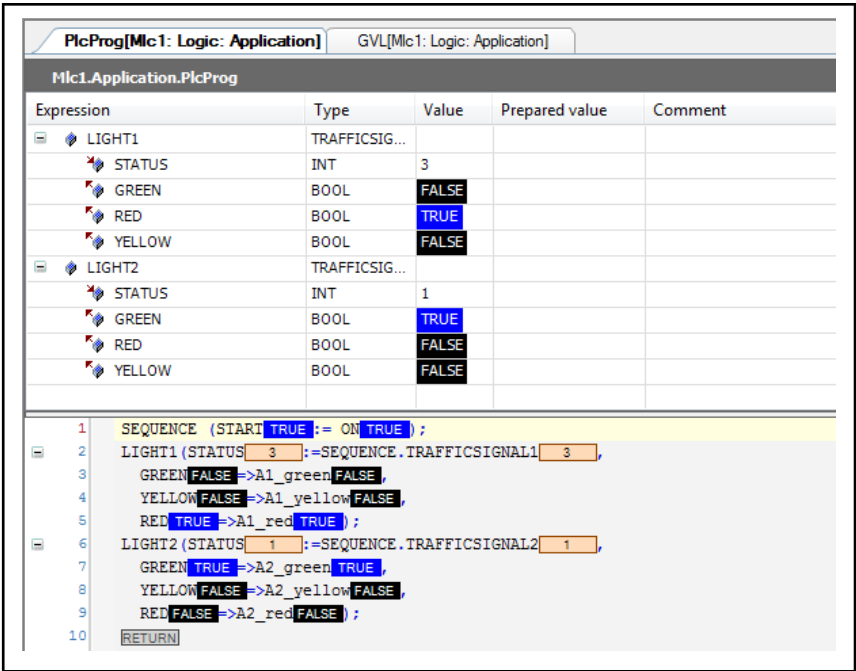


Fig. 2-55: "PLCProg" program online

Stopping application To stop the PLC program, click on **Debug ► Stop** in the IndraWorks menu bar.

Logging out of the control To log out of the control, select **Debug ► Logout** from the IndraWorks menu bar.

Editing "PlcProg" program in "CFC"



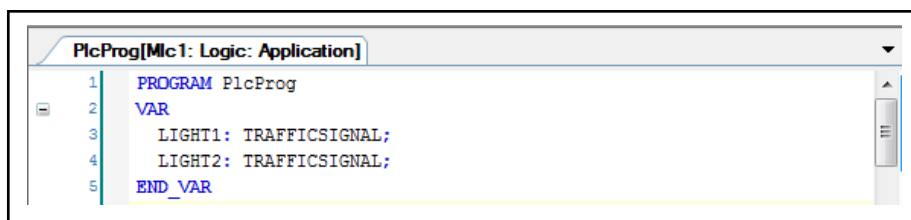
The "PlcProg" program can be implemented in two programming languages:

- For programming in "Structured Text (ST)", refer to [chapter "Editing PlcProg program in ST" on page 52](#)
- For programming in "Continuous Function Chart (CFC)", continue in the text

"PlcProg" function The temporal sequence of the traffic light phases for both traffic lights was implemented in the "SEQUENCE" function block. The inputs and outputs of a PLC are now to be controlled and evaluated. Input and output variables for the one input and the six outputs for the two traffic lights were created in the "Global Variable List (GVL)". These variables can now be used in the "PlcProg" POU. The application was previously tested only in simulation mode. The application can also be tested in a control.

Declaring an instance The "PlcProg" program receives a "LIGHT1" instance of type "TRAFFICSIGNAL", and a "LIGHT2" instance, also of type "TRAFFICSIGNAL". This instantiates the "TRAFFICSIGNAL" function block.
Create the LIGHT1: TRAFFICSIGNAL; and the LIGHT2: TRAFFICSIGNAL variables:

First steps - IndraLogic

Fig. 2-56: Declaration section POU *PlcProg***Editing body**

The "PlcProg" POU is edited in the "CFC" programming language. The first function block to call is the "SEQUENCE" function block:

1. In the "ToolBox" window, click to select "Block (Box)".

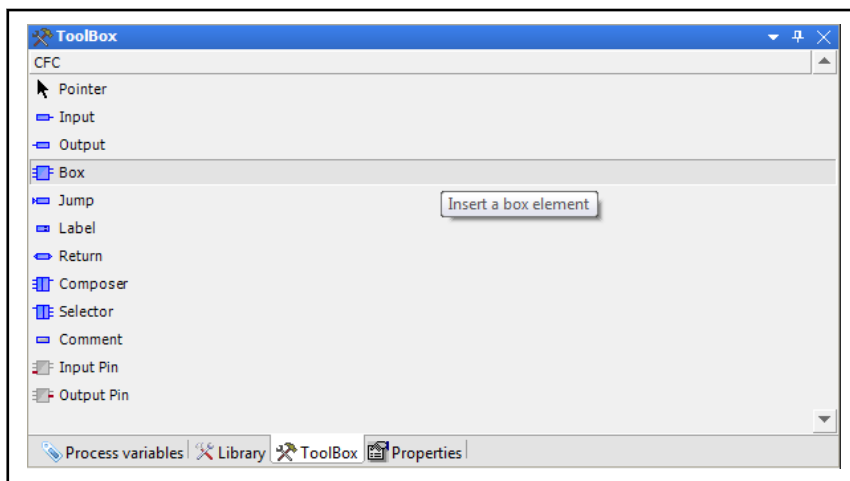


Fig. 2-57: Tools – "Box" (function block) element

2. Move the mouse cursor into the body of the "PlcProg" editor and left click to add the function block.
3. Click on the "???" name of the function block and specify the name "SEQUENCE" for the function block.

The function block is as follows:

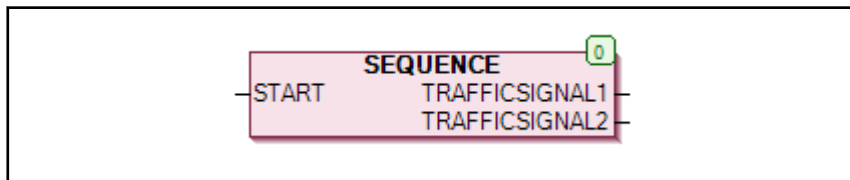


Fig. 2-58: SEQUENCE function block

At the "TRAFFICSIGNAL1" output of the "SEQUENCE" function block, a "LIGHT1" function block with the "TRAFFICSIGNAL" function is added:

1. In the "ToolBox" window, click to select the "Block (Box)" element.
2. Move the mouse cursor into the body of the "PlcProg" editor and left click to add the function block.
3. Click on the "???" name of the function block and specify the type name "TRAFFICSIGNAL" for the function block.
4. At "???" specify the instance name "LIGHT1" for the "TRAFFICSIGNAL" function block.

The function block is as follows:

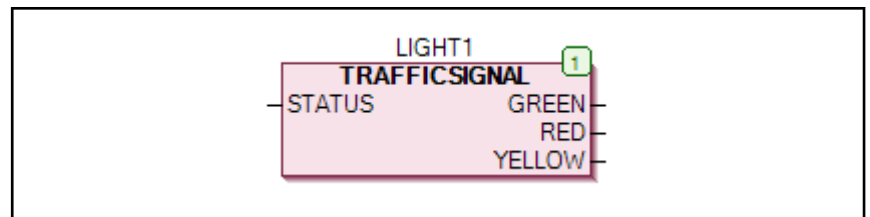


Fig. 2-59: TRAFFICSIGNAL – LIGHT1 function block

Then, at the "TRAFFICSIGNAL2" output of the "SEQUENCE" function block, a "LIGHT2" function block is added with the "TRAFFICSIGNAL" function:

1. In the "ToolBox" window, click to select the "Block (Box)" element.
2. Move the mouse cursor into the body of the "PlcProg" editor and left-click to add the function block.
3. Click on the "???" name of the function block and specify the name "TRAFFICSIGNAL" for the function block.
4. At "???" specify the instance name "LIGHT2" for the "TRAFFICSIGNAL" function block.

The function block is as follows:

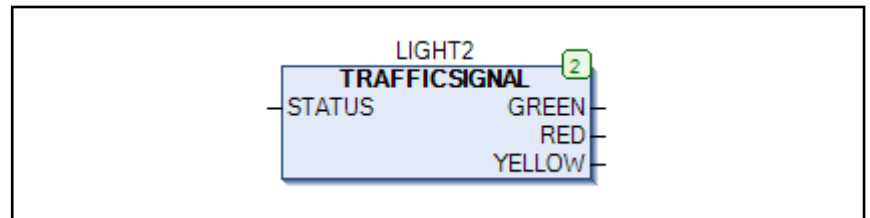


Fig. 2-60: TRAFFICSIGNAL – LIGHT2 function block

Then, the "Input" and "Output" elements are added:

1. In the "ToolBox" window, click to select the "Input" element.
2. Move the mouse cursor into the body of the "PlcProg" editor and left-click to add the input.
3. Click on the "???" name of the input and specify the name "ON" for the input.
4. In the "ToolBox" window, click to select the "Output" element.
5. Move the mouse cursor into the body of the "PlcProg" editor and left-click to add the output.
6. Add five more outputs in "PlcProg".
7. Name the outputs "A1_green", "A1_yellow", "A1_red", "A2_green", "A2_yellow", "A2_red".



"A1_green" is correctly called "GVL.A1_green" or ".A1_green" for short, and so on, since these are global variables.

However, if there is no locally declared "A1_green", the system uses the global variable of the same name as substitute.

To complete the network, proceed as follows:

1. Left-click on the output of the "ON" element and keep the mouse button pressed. Move the mouse cursor to the "START" input of the "SEQUENCE" function block to connect both elements with each other.

First steps - IndraLogic

- Left click on the "TRAFFICSIGNAL1" output of the "SEQUENCE" element and keep the mouse button pressed. Move the mouse cursor to the "STATUS" input of the "TRAFFICSIGNAL" function block to connect both elements with each other.
- Left click on the "TRAFFICSIGNAL2" output of the "SEQUENCE" element and keep the mouse button pressed. Move the mouse cursor to the "STATUS" input of the "LIGHT2" function block to connect both elements with each other.
- Left click on the "GREEN" output of the "LIGHT1" element and keep the mouse button pressed. Move the mouse cursor to the "A1_green" input of the output to connect both elements with each other.
- Repeat the final step for the other outputs accordingly.

The "PlcProg" program is as follows:

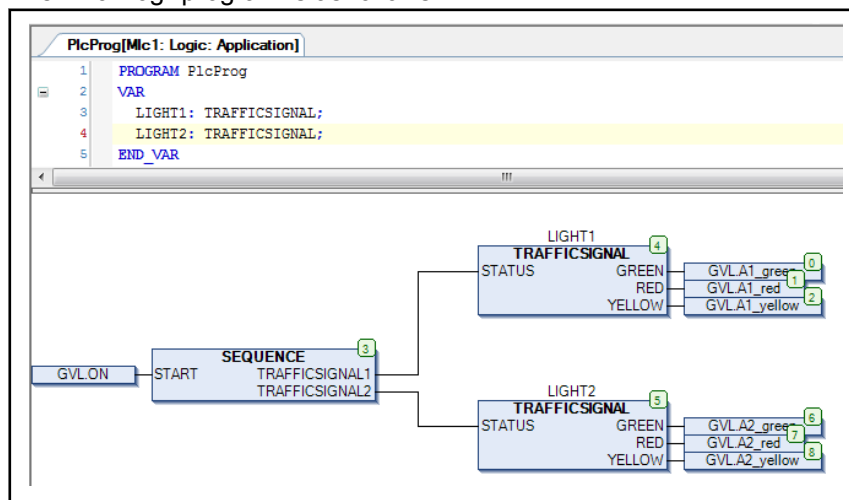


Fig. 2-61: POU "PlcProg" completed



The green numbers at the right upper corner of the elements indicate the execution sequence of the elements.

This sequence - and thus the function of the CFC circuit - can be changed by the programmer if required.



The elements, such as "Input", "Output" and "Function block" can also be added using the **CFC** menu.

Testing the application in simulation mode, PlcProg in CFC

Build application

To load the application to the control, the "compiler" has to compile the application first without errors. To compile the application, select **Create ► Build application** from the menu. After compilation, the following is displayed in the "Task list" below the workspace: "Compiling completed -- 0 errors, 0 warnings: Ready for download!".



Only the currently active application is compiled.

If **multiple** applications are **available**, the application to be compiled has to be set to active (context menu "Application" **Set active application**).

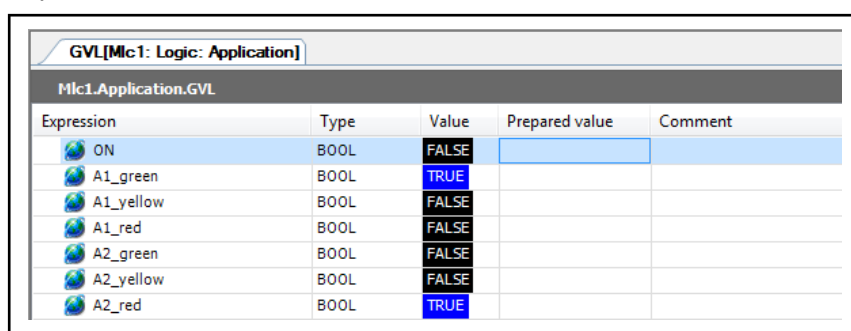
Simulating a control

If there is no control for testing, simulate one. To simulate a control, click on "Mlc1" in the "Project Explorer" and right-click to open the context menu.

First steps - IndraLogic

- Click on **Simulation** to activate the simulation mode. If the simulation is active, "Simulation" is selected.
- Logging into the control** Now log into the control and load the application to the control. To log into the control, select **Debug ► Login** from the IndraWorks menu bar. The new application can be loaded to the control either via online change and as recompilation.
- Starting application** To start the application, right-click on the "Application" node and select **Start** from the context menu. The "application" is now running as simulation.
- Switching on the traffic light and observing the traffic light cycle** After the application is started, "LIGHT1" is in the state "Green" and "LIGHT2" is in the state "red". The traffic light has to be switched on so that the traffic light cycle can be run through. To switch on the traffic light, proceed as follows:

1. To open the "Global Variable List", double-click on "GVL" in the Project Explorer.



GVL[Mlc1: Logic: Application]				
Mlc1.Application.GVL				
Expression	Type	Value	Prepared value	Comment
ON	BOOL	FALSE		
A1_green	BOOL	TRUE		
A1_yellow	BOOL	FALSE		
A1_red	BOOL	FALSE		
A2_green	BOOL	FALSE		
A2_yellow	BOOL	FALSE		
A2_red	BOOL	TRUE		

Fig. 2-62: "GVL" online

2. At the "ON" variable, click into the "Prepared value" column so that "TRUE" is displayed in this column.
3. To write the value to the control, click on **Debug ► Write values** in the IndraWorks menu bar.

The traffic light cycle is now run. The traffic light cycle can be observed in the "PlcProg" function block.

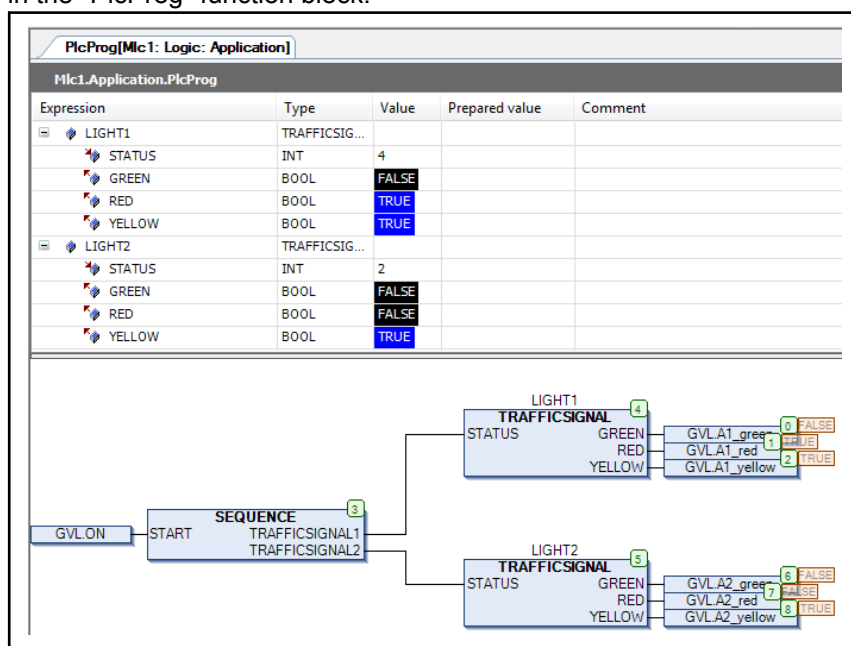


Fig. 2-63: "PlcProg" program online

First steps - IndraLogic

Stopping application To stop the PLC program, click on **Debug ► Stop** in the IndraWorks menu bar.

Logging out of the control To log out of the control, select **Debug ► Logout** from the IndraWorks menu bar.

2.6.8 Testing the application on the control

Disabling the simulation mode If available, test the application in a control.

Therefore, [chapter 2.5.2 "Communication settings" on page 18](#) and the desired control have to match.

To end the simulation mode, right-click on the "Mlc1" control in the "Project Explorer" to open the context menu. To disable the simulation mode, click on **Simulation**. If the simulation is disabled, the "Simulation" checkbox is deselected.

Build application To load the application to the control, the application has to be compiled first without errors by the "compiler". To compile the application, select **Create ► Build** from the IndraWorks menu bar. After compilation, the following is displayed in the "Task list" below the workspace: "Compiling completed -- 0 errors, 0 warnings"



Only the currently active application is compiled.

If **multiple** applications are **available**, the application to be compiled has to be set to active (context menu "Application" **Set active application**).

Logging into the control Now log into the control and load the application to the control. To log into the control, select **Debug ► Login** from the IndraWorks menu bar.

The application is then in the "STOP" state.

Starting application To start the application, right-click on the application and select **Start** from the context menu. The application is now running on the control.

Generating boot application Select **Debug ► Generate boot application** in the IndraWorks menu bar to create a boot project. If this command is executed online, the compiled project is stored on the control so that it loads automatically upon next startup.

Switching on the traffic light and observing the traffic light cycle After the application is started, "traffic light 1" is "green" and "traffic light 2" is in "red". The traffic light has to be switched on so that the traffic light cycle can be run through. To switch on the traffic light, proceed as follows:

1. To open the Global Variable List, double-click on "GVL" in the Project Explorer.

GVL[Mlc1: Logic: Application]				
Mlc1.Application.GVL				
Expression	Type	Value	Prepared value	Comment
ON	BOOL	FALSE		
A1_green	BOOL	TRUE		
A1_yellow	BOOL	FALSE		
A1_red	BOOL	FALSE		
A2_green	BOOL	FALSE		
A2_yellow	BOOL	FALSE		
A2_red	BOOL	TRUE		

Fig. 2-64: "GVL" online

2. At the "ON" variable, click into the "Prepared value" column so that "TRUE" is displayed in this column.

3. To write the value to the control, click on **Debug ► Write values** in the IndraWorks menu bar.
- The traffic light cycle is now run.
- Follow the processing of the traffic signal cycle in the individual program POU's (PlcProg, SEQUENCE) and the Global Variable List (GVL).
- Open the "TRAFFICSIGNAL" function block. A selection window is displayed first providing the alternative "PlcProg.LIGHT1" or "PlcProg.LIGHT2".

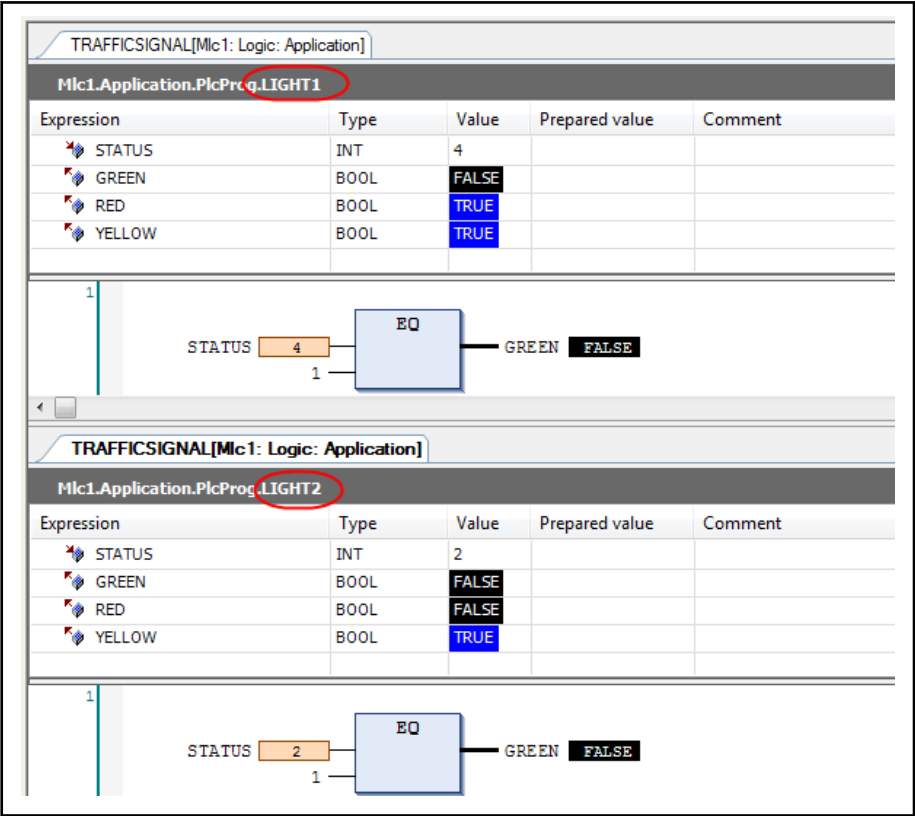


Fig. 2-65: Status display in the two function block instances of TRAFFICSIGNAL

Checking real inputs and outputs

The controls IndraMotion MLC L45, L65, L75 are provided with onboard outputs assigned to the addresses %QX0.0 ... %QX0.7 (mapping). The traffic light cycle can also be followed at these outputs.

Stopping application

To stop the PLC program, click on **Debug ► Stop** in the IndraWorks menu bar.

Logging out of the control

To log out of the control, select **Debug ► Logout** from the IndraWorks menu bar.

2.7 Creating visualization for the PLC program

2.7.1 Visualization in IndraLogic

When visualizing with IndraLogic, dynamic procedures can be created and represented easily and quickly. The status of the variables is queried at defined intervals and the corresponding graphic elements are displayed according to their states. The PLC program described in [chapter 2.6 "Creating a PLC program" on page 25](#) is to be visualized. After the programmed traffic light system, draw two traffic lights and one on-switch for visualization purposes and represent them according to their states.

First steps - IndraLogic

2.7.2 Creating a new visualization

To create new visualization, proceed as follows:

1. Add a visualization to the application by dragging the "Visualization" element from the "Library (PLC objects/ VI Logic objects)" into the Project Explorer and dropping it onto the application.

The following window opens:

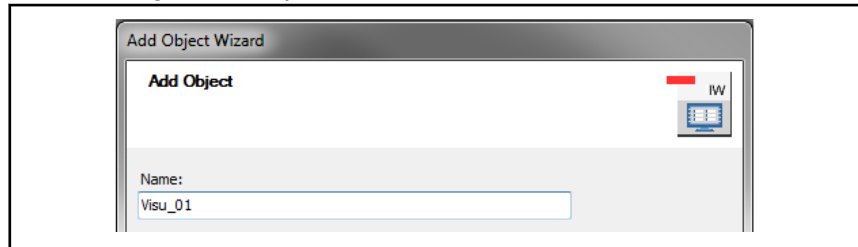


Fig. 2-66: Adding a visualization to the application

2. Enter any name for the visualization and press **Finish**.

Visualization is added together with an instance of the visualization manager:

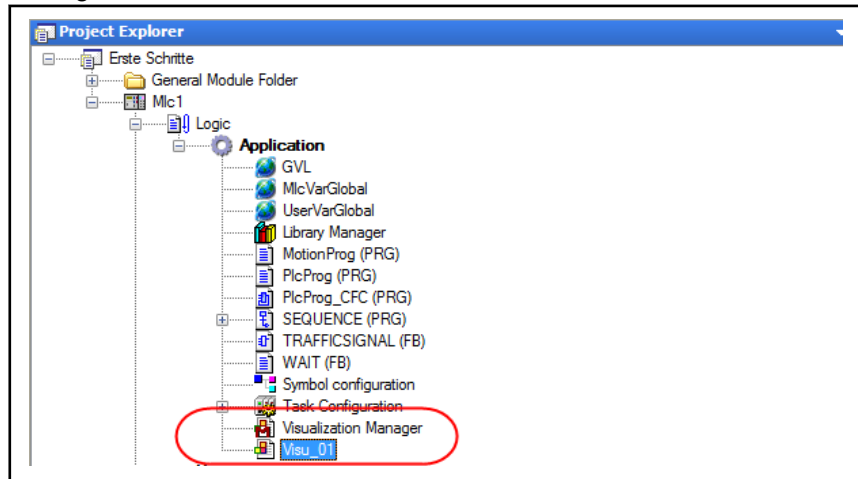


Fig. 2-67: Visualization added to application

Further elements, which are not required, are additionally generated in the "General module" folder.

2.7.3 Editing visualization

Adding and editing the first element "Traffic light lamp"

Opening visualization editor Double-click to open the visualization editor on the visualization in the Project Explorer.

A "drawing sheet" is provided. The position (X, Y) = (0, 0) is located in the upper left corner.

A grid can be made visible and enabled via **Tools ► Options ► IndraLogic 2G ► Visualization**.

Five folders are provided in the toolbox:

- **Basis:** General visualization objects
- **General controls:** Buttons, checkboxes, loading bars...

- **Measuring devices:** Bar graph displays, pointer instruments, potentiometers, histogram,...
- **Lamps/switches. Images:** Lamps, switches, bitmaps...
- **Special controls:** Trace objects, waiting icons...

Adding "Ellipse" element

The first element of the "lamp" of the "traffic light" has to be added to the visualization. Elements that can be added to a visualization are located in the "ToolBox" window (refer to the following figure).

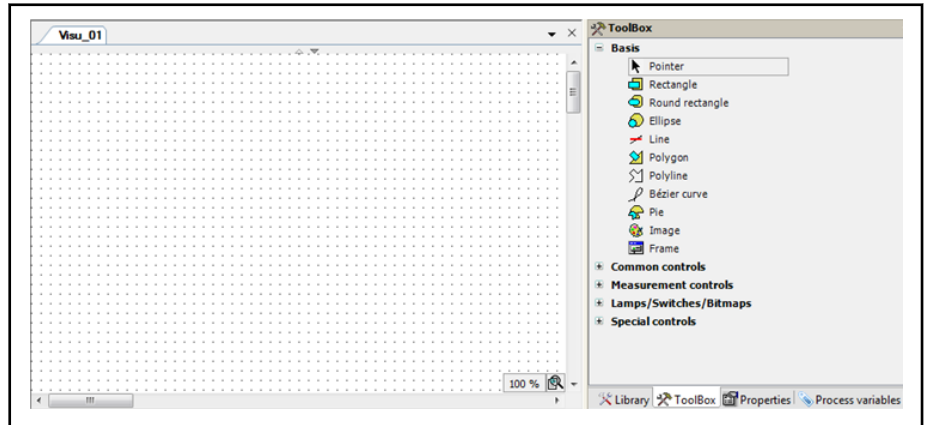


Fig. 2-68: "ToolBox" window

To add the first element "Lamp", proceed as follows:

1. In the "ToolBox" window, left-click to select the "Ellipse" element.
2. Move the mouse cursor to the visualization window and left-click again.

An "ellipse" element is added to the visualization editor.

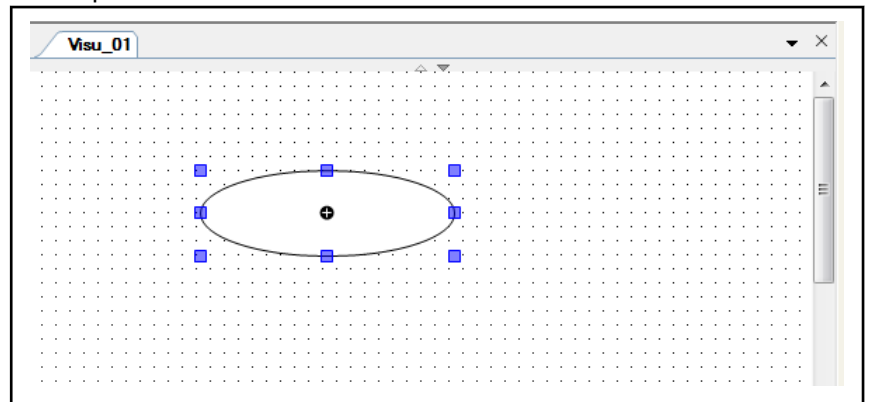


Fig. 2-69: Visualization editor - Ellipse

Changing shape of ellipse

The "ellipse" added is to be changed in its shape to match the shape of a "traffic light lamp".

To change the shape and position of the "Ellipse", proceed as follows:

1. To highlight the "ellipse", left-click on the "ellipse".
2. To open the "Properties" window, left-click on the "Properties" tab.

First steps - IndraLogic

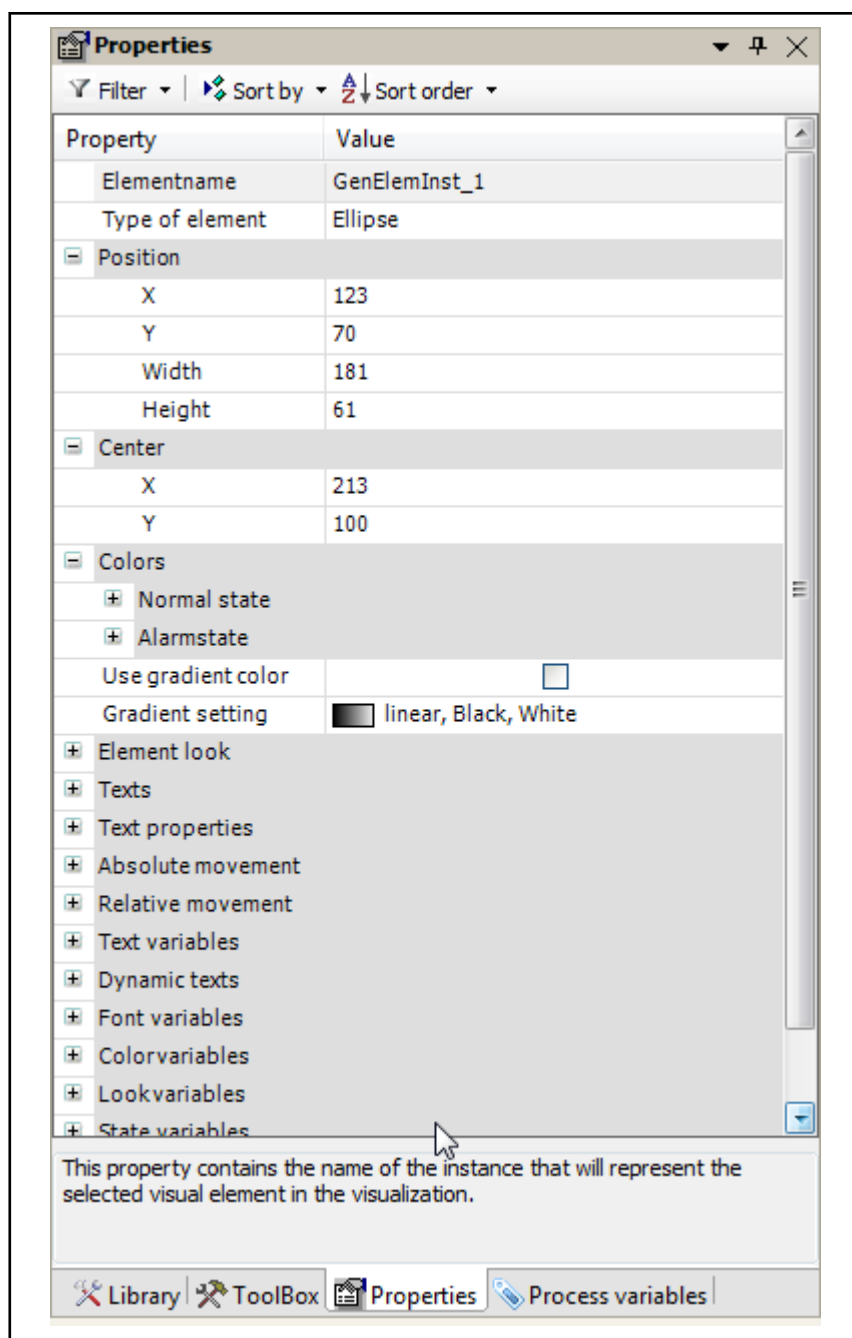


Fig. 2-70: "Properties" window of the ellipse

3. To change the position (X,Y) and the size (width, height), select the following under the property "Position":
 - X = "60"
 - Y = "150"
 - Width = "30"
 - Height = "30"
4. To change the center point of the ellipse, select (X, Y) as follows under "Center":
 - X = "75"
 - Y = "165"

Changing properties of traffic light lamp

The fill color of the traffic light lamp is to be changes. The traffic light lamp should be gray when the traffic light system is inactive and red when it is active.

To change the fill color, proceed as follows:

1. To change the fill color for the normal state, click on ... under the "Colors" property and "Normal state" under "Fill color".
The "Colors" window opens.

2. In the "Colors" window, click on **Fill color**. Select "Grey".

3. To change the fill color for the alarm state, click on ... under the "Colors" property and "Alarm state" under "Fill color".
The "Colors" window opens.

4. In the "Colors" window, click on **Fill color**. Select "Red".

In normal state, the traffic light lamp is gray. The traffic light lamp should be red, if the traffic light status is "red". The traffic light is red when the "A1_red" variable of the "GVL" in the POU "PlcProg" is "TRUE". To make the state to the traffic light lamp dynamic, enter the variable for changing the signal color in the traffic light lamp properties.

Proceed as follows:

1. To communicate the variable for changing the signal color to the visualization, double-click in the "Value" column under "Color variables" in "ToggleColor".

2. Click on the ... button.

The "Input assistant" window opens.

3. "Application", "GVL" is displayed under "Elements" in the "Input assistant" window.

4. Select the "A1_red" variable under "GVL variables".

The "GVL variable" was added to "ToggleColor".

First steps - IndraLogic

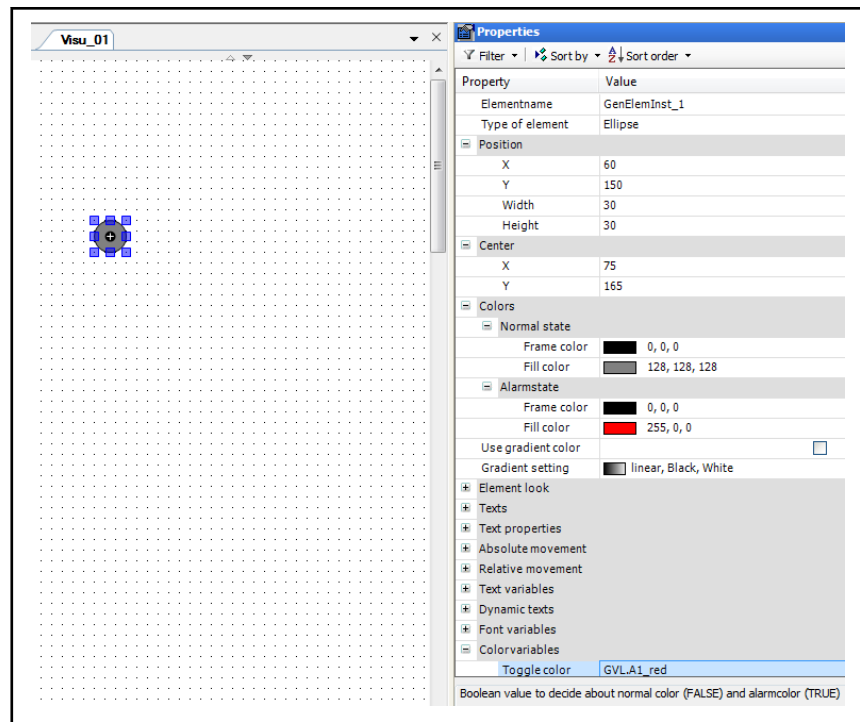


Fig. 2-71: Properties – ToggleColor

Copying, adding and editing the "Traffic light lamp" element

Copying and inserting "traffic light lamp" element

The first traffic light lamp with the signal color "red" is now complete. Only the signal lamps for the signal colors "yellow" and "green" are missing for the first traffic light lamp. Since the two other traffic light lamps have the same shape as the first traffic light lamp, the easiest method is to copy the first traffic light lamp light.

To copy and insert the traffic light lamp, proceed as follows:

1. Click to select the traffic light lamp.
2. Right-click to open the context menu and select **Copy**.
3. Click in the visualization area and select **Insert** from the context menu.
4. To add another traffic light lamp, click again in a free area in the visualization editor and select **Insert** in the context menu.
5. Left-click to select the first inserted traffic light lamp.

Changing the position of the "traffic light lamp" elements

Position the two newly inserted traffic light lamps.

To position the traffic light lamps, proceed as follows:

1. Left-click to select the first inserted traffic light lamp.
2. To change the position, select 60 for X and 195 for Y in the "Properties" window under the property "Position".
3. To change the center point, select 75 for X and 210 for Y under the property "Center".
4. Left-click to select the second inserted traffic light lamp.
5. To change the position, select 60 for X and 240 for Y in the "Properties" window under the property "Position".
6. To change the center point, select 75 for X and 255 for Y under the property "Center".

Changing the properties of the "traffic light lamp" elements

New colors for the "Alarm state" are assigned to both new traffic light lamps.
Proceed as follows:

1. Highlight the traffic lamp in the center.
2. To change the fill color for the alarm state, click on ... under the "Colors" property and "Alarm state" under "Fill color".
3. The "Colors" window opens.
4. In the "Colors" window, click on **Fill color**. Select "Yellow".
5. Repeat the steps 1 - 3 for the lower traffic light lamp.
6. In the "Colors" window, click on **Fill color**. Select "Green".

Adapt the variables for addressing the different traffic light colors.

Proceed as follows:

1. Highlight the traffic lamp in the center.
2. To communicate the variable for changing the signal color to the visualization, double-click in the "Value" column under "Color variables" in "ToggleColor".

Click on the ... button.

The "Input assistant" dialog opens.

3. "Application", "GVL" is displayed under "Elements" in the "Input assistant" window.
4. Select the "A1_yellow" variable under "GVL variables".
5. Select the lowest traffic light lamp.
6. Select the "A1_green" variable under "GVL variables".

Adding and editing the "Traffic light housing" element

Elements of the traffic light housing

The three lamps to display the signal states are now complete. The traffic light should look like a traffic light. The following three elements are added for the visual display of the traffic light system:

1. Housing for signal lamps.
2. Base for traffic light housing.
3. Traffic light base

Inserting "Rectangle" element

To add the signal lamp housing, proceed as follows:

1. Select the "Rectangle" element from the "ToolBox" window.
2. Click in a free area of the visualization editor to insert the "Rectangle" element.

Changing the position of the "traffic light housing" elements

To change the shape and position of the housing, proceed as follows:

1. Select the inserted "Rectangle" element and open the "Properties" window.
2. To change the position, select 50 for X and 135 for Y in the "Properties" window under the property "Position".
3. To change the shape, select 50 for width and 150 for height in the "Properties" window under the "Position" property.
4. To change the center point, select 50 for X and 210 for Y under the property "Center".
5. Right-click at the highlighted object to open the context menu. Select **Sequence ► Send to back**.

Changing the properties of the "traffic light housing" elements

To change the fill color for the traffic light housing, proceed as follows:

First steps - IndraLogic

1. In the "Properties" window under "Colors", "Normal state", "Fill color", click on the ... button.
The "Colors" window opens.
2. In the "Colors" window, click on the **Define color** button.
3. For the "Red", "Green" and "Blue" fields, enter the following:
(Red, green, blue) = (5, 5, 10).

Adding and editing "traffic light base" element

Inserting "Rectangle" element Next, the traffic light base is constructed.

Proceed as follows:

1. Select the "Rectangle" element from the "ToolBox" window.
2. Click into a free area of the visualization editor to insert the "Rectangle" element.

Changing the position of the "traffic light base" element

1. Select the inserted "Rectangle" element and open the "Properties" window.
2. To change the position, select 70 for X and 285 for Y in the "Properties" window under the property "Position".
3. To change the shape, select 10 for width and 195 for height in the "Properties" window under the "Position" property.
4. To change the center point, select 75 for X and 380 for Y under the property "Center".

Changing the properties of the "traffic light base" element

Select dark gray (DimGray) (colors, normal state, fill colors, (red, green, blue) = (105, 105, 105)) as the fill color for the base of the signal lamps.

Adding and editing "traffic light base" element

Inserting "Rectangle" element The first traffic light still needs a base.

To construct the traffic light base, proceed as follows:

1. Select the "Rectangle" element from the "ToolBox" window and add the element to the workspace.
2. Select the inserted "Rectangle" element and open the "Properties" window.
3. To change the position, select 15 for X and 480 for Y in the "Properties" window under the property "Position".
4. To change the shape, select 120 for width and 30 for height in the "Properties" window under the "Position" property.
5. To change the center point, select 75 for X and 495 for Y under the property "Center".

Changing the properties of the "Traffic light base" element

Select dark gray (DimGray) (colors, normal state, fill colors, (red, green, blue) = (105, 105, 105)) as fill color for the traffic light base of the signal lamps.

The first traffic light looks as follows:

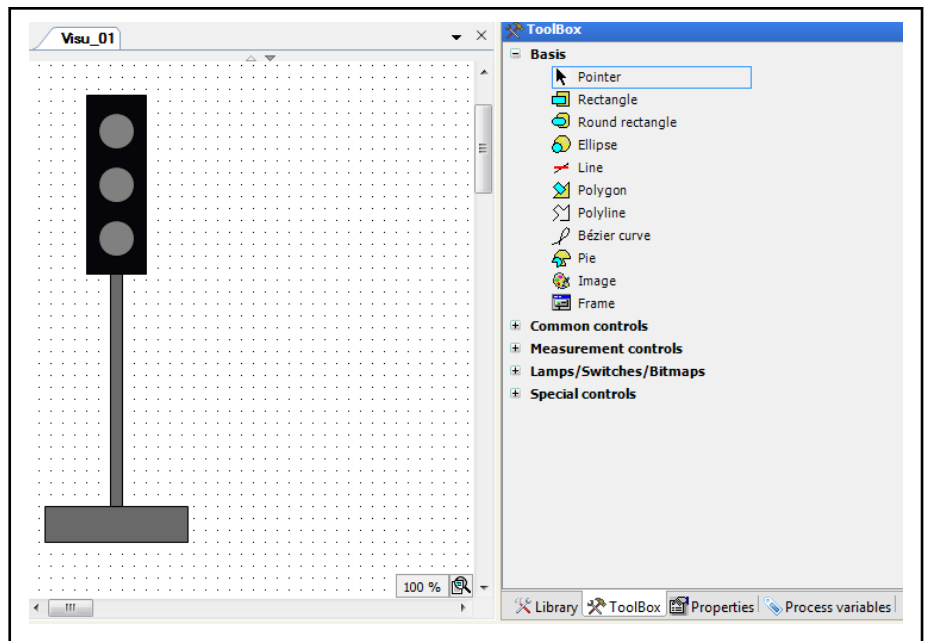


Fig. 2-72: First traffic light

Adding and editing second traffic light

The second traffic light

To create the second traffic light, highlight all traffic light elements. Left-click in the right area below next to the traffic light and keep the mouse button pressed. Move the mouse cursor to a position on the left above the traffic light. Release the mouse button. All elements are highlighted.



Elements can also be highlighted by pressing and holding the <Ctrl> key while individually highlighting each element.

In the context menu, select **Copy** and **Insert**. Position the highlighted object next to the first traffic light.

First steps - IndraLogic

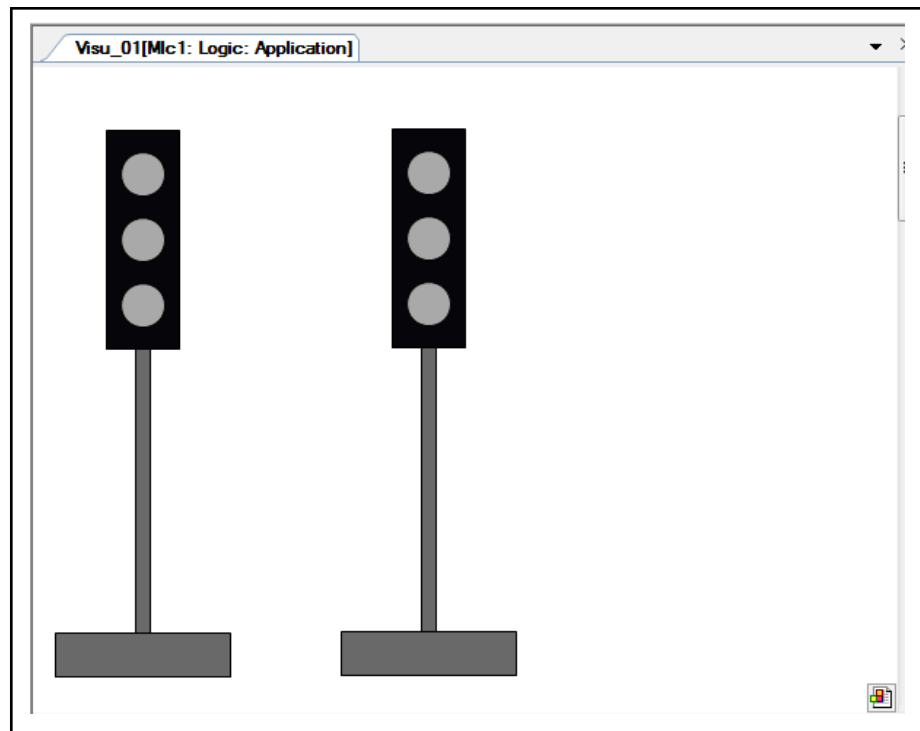


Fig. 2-73: Two traffic lights

Adjust the variables for switching among the different traffic light phases. Modify the variables accordingly in the "Properties" window in "ToggleColor".

"GVL.A1_red" becomes "GVL.A2_red",

"GVL.A1_yellow" becomes "GVL.A2_yellow" and

"GVL.A1_green" becomes "GVL.A2_green".

Adding and editing on-switch

Adding "Button" element

It should be possible to switch on the traffic light at the beginning. Therefore, insert a "button":

1. Select the "Button" element from the window **ToolBox ► Common controls**.
2. Click in a free area of the visualization editor to insert the "Button" element.

Changing position of "Button" element

1. To change the position, select 130 for X and 55 for Y in the "Properties" window under the property "Position".
2. To change the shape, select 70 for width and 40 for height in the "Properties" window under "Position".
3. To change the center point, select 165 for X and 75 for Y under the property "Center".

Changing properties of "Button" element

1. Enter "ON" under "Texts" property and "Text".
2. Click on "Configure" in the "Input configuration" property under "OnMouseClick".

The "Input configuration" window opens.

3. In the "Input configuration" window in the left window area, highlight "Switch variable" and click on ►.

"Switch variable" is added to the right window area.

4. Click in the "Variable" input field and enter "GVL.ON".
5. To confirm the entry, press **OK**.

Adding and editing labeling

Inserting "Rectangle" element

Now, label both traffic lights with "Traffic light 1" and "Traffic Light 2".

To label the traffic lights, proceed as follows:

1. Select the "Rectangle" element from the "ToolBox" window.
2. Click into a free area of the visualization editor to insert the "Rectangle" element.

"Rectangle" element, changing position

1. Select the inserted "Rectangle" element and open the "Properties" window.
2. To change the position, select 15 for X and 520 for Y in the "Properties" window under the property "Position".
3. To change the shape, select 120 for width and 30 for height in the "Properties" window under the "Position" property.
4. To change the center point, select 75 for X and 535 for Y under the property "Center".

Changing the properties of the "Rectangle" element

1. Enter "Traffic light 2" into the "Text" field next to "Traffic light 1".
2. In the "Properties" window under "Colors", "Normal state", "Frame color", click on the ... button.

The "Colors" window opens.

3. In the "Colors" window, click on the **Define color** button.
4. For the "Red", "Green" and "Blue" fields, enter the following: (Red, green, blue) = (255, 255, 255).

Copying and adapting "Rectangle" element

1. Copy the label and place it under traffic light 2.
2. Enter "Traffic light 2" into the "Text" field next to "Texts".

The visualization is complete:

First steps - IndraLogic

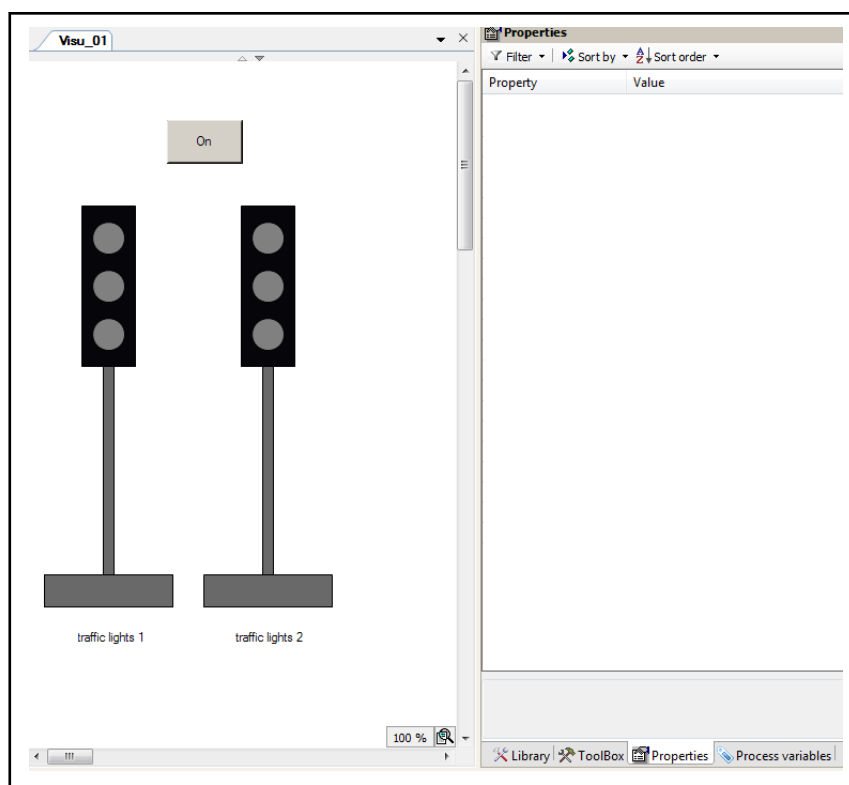


Fig. 2-74: Visualization completed

Displaying variable values

Inserting "Rectangle" element

It can be useful to display the value of a variable in a format to commission a visualization or also as a permanent solution.

Now, provide a display of their respective "STATUS" variable below the labeling "Traffic light 1" and "Traffic light 2" of the two traffic lights.

To label the traffic lights, proceed as follows:

1. Select the "Rectangle" element from the "ToolBox" window.
2. Click into a free area of the visualization editor to insert the "Rectangle" element.

"Rectangle" element, changing position

1. Select the inserted "Rectangle" element and open the "Properties" window.
2. To change the position, select 15 for X and 570 for Y in the "Properties" window under the property "Position".
3. To change the shape, select 120 for width and 30 for height in the "Properties" window under the "Position" property.
4. To change the center point, select 75 for X and 585 for Y under the property "Center".

Changing the properties of the "Rectangle" element

1. Enter "Traffic light 2" into the "Text" field next to "Traffic light 1: %s".
2. Click in the "Text variable" field on "Text variables" and select the variable **PlcProg.LIGHT1.STATUS** using the "..." button.

The value of the "PlcProg.LIGHT1.STATUS" variable is displayed instead of the format specification "%s" in online mode.

Copying and adapting "Rectangle" element

1. Copy the display and place it under traffic light 2.

2. For display 2, replace:

- Text "Traffic light 1: %s" by "Traffic light 2: %s"
- Text variable "PlcProg.LIGHT1.STATUS" by "PlcProg.LIGHT2.STATUS"

Visualization with supplemented display:

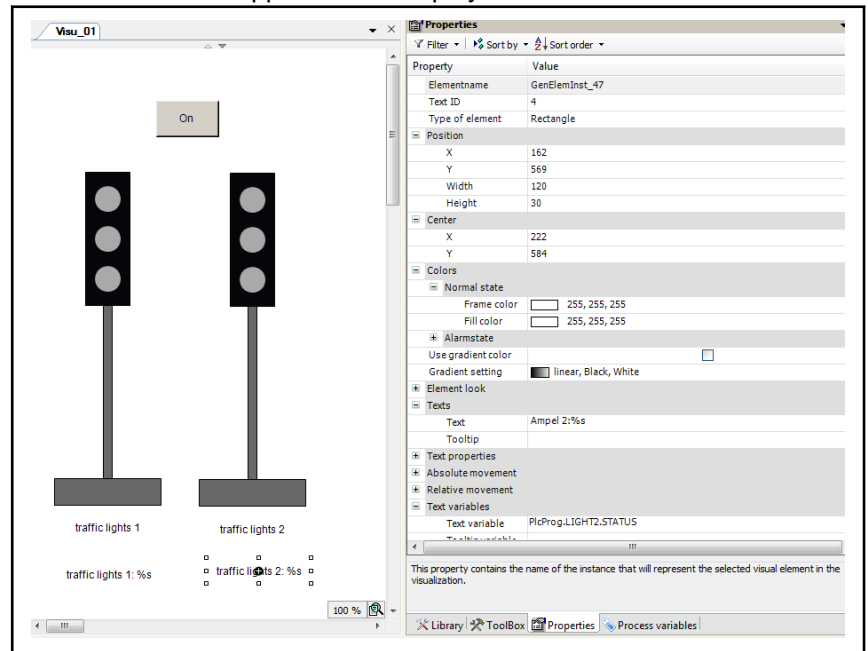


Fig. 2-75: Visualization supplemented by display

2.7.4 Testing visualization

The visualization can be tested in simulation mode or, if a real control is available, with a real control.

To test the visualization in simulation mode, refer to [chapter "Testing the application in simulation mode, PlcProg in ST"](#) on page 54 or [chapter "Testing the application in simulation mode, PlcProg in CFC"](#) on page 58.

To test the visualization with a control, refer to [chapter 2.6.8 "Testing the application on the control"](#) on page 60.

Open the "Visualization" window when the application is running:

First steps - IndraLogic

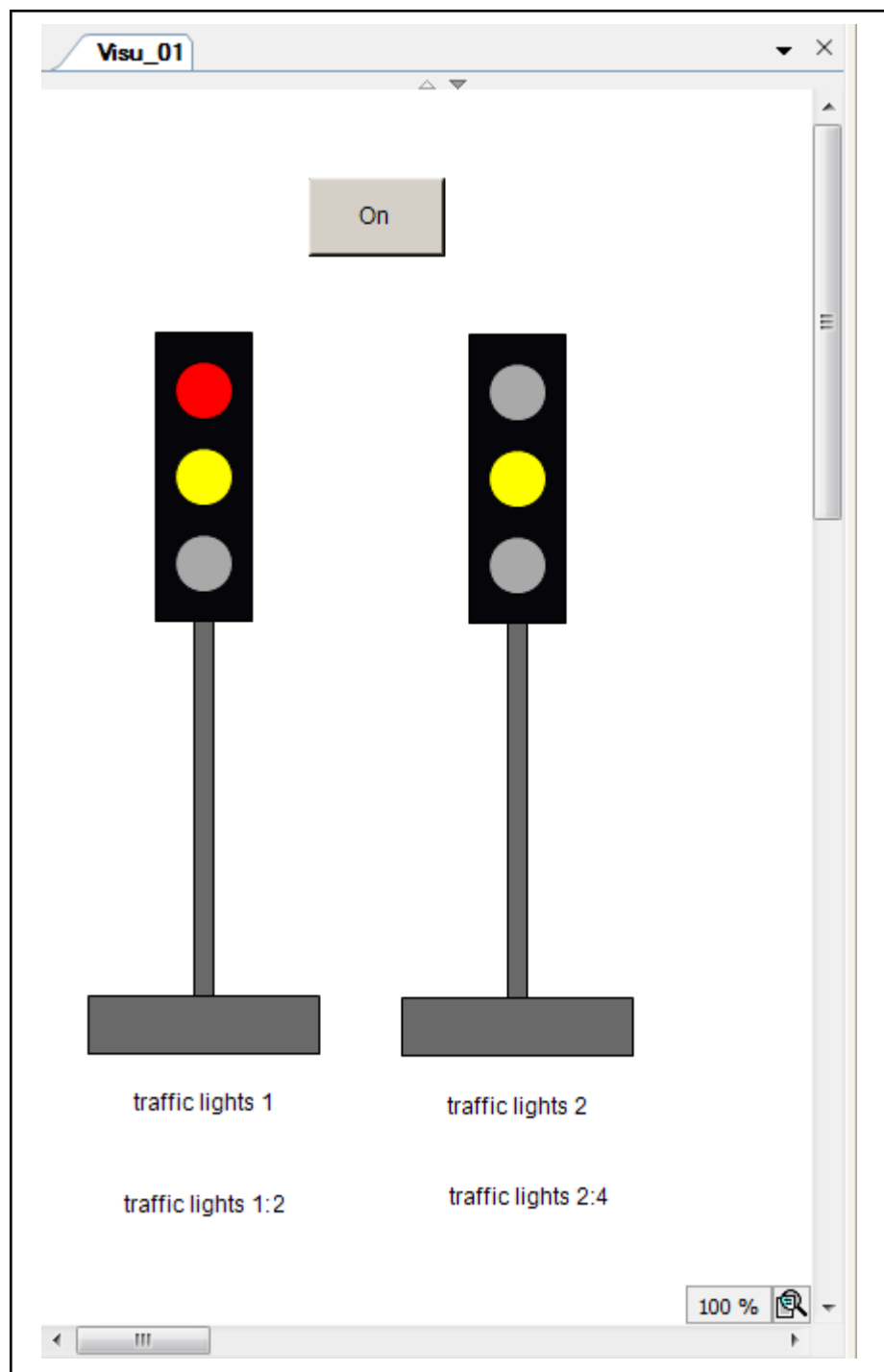


Fig. 2-76: Online visualization

Click on **ON**. The traffic light cycles are run through.

2.8 Restoring a project

Storage location An export file was generated from the created project allowing to restore the project irrespective of the control hardware (IndraMotion MLC L25, L45, L65 L75, XM2, VPx). The restored project can run as simulation or by using your control.

The export file was created for the "Application" node and includes the following components:

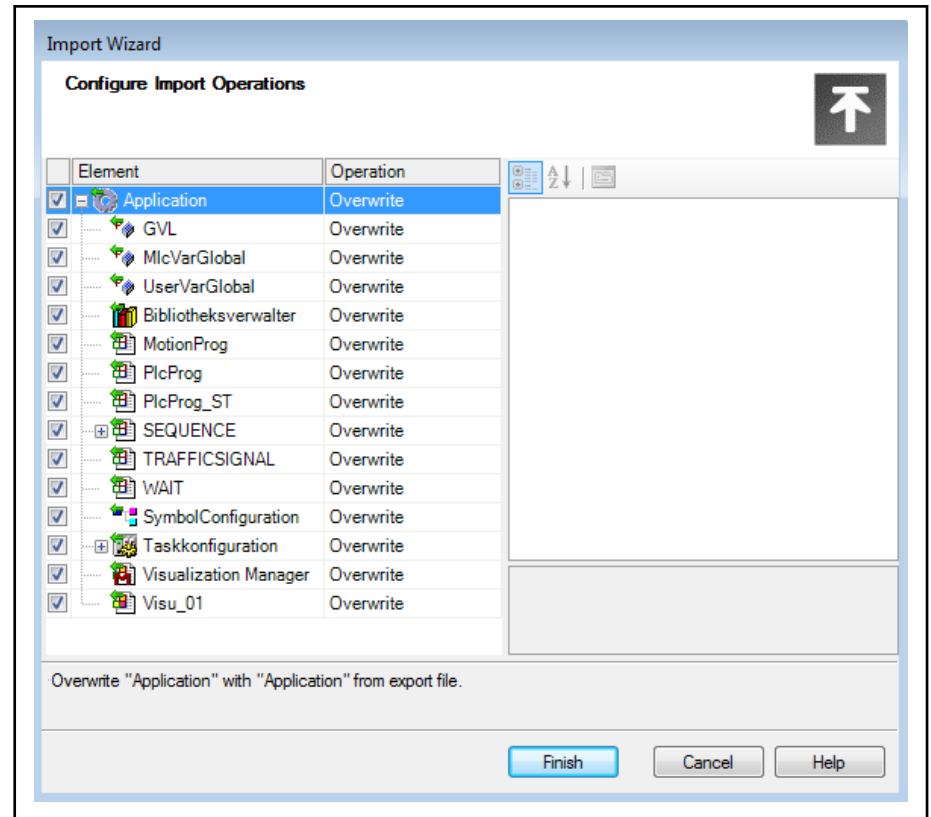


Fig. 2-77: Components of the export file

The specified components are overwritten during import.

Restoring

To make the project run on the control (IndraMotion MLC L25, L45, L65, L75, XM2, VPx), proceed as follows:

1. Clear old data from the control.
 Clear the old memory content of the control.
 For details, refer to [chapter 2.3.1 "Resetting the control \(CMLx5\)" on page 16](#).
 The control memory is cleared and any possible Motion component present and the PLC component are ready for operation.
2. Create a new project.
 Create a new empty project (**File ► New ► Project**). Select any project, although "First steps" is recommended.
 A new empty project has been created.
3. Create the control.
 Drag a control (IndraMotion MLC L25, L45, L65, L75, XM2, VPx) from the device library and drop it onto the empty project. Any control name can be selected. However, "Mlc1" is recommended.
 Complete device type and IP address.
 Test the connection (not required in case of pure "simulation").
 Complete the insertion as usual.

First steps - IndraLogic

The control is in the project now.

4. Import the export file "Application_FirstSteps_IndraLogic_Basics.iwx".
In the Project Explorer, expand the control node up to **Logic ► Application**.
Select **Import...** from the control context menu of the "Application" node.
Select the export file "Application_FirstSteps_IndraLogic_Basics.iwx" in "<Drive>:\All Programs\Rexroth\IndraWorks\Projects". The "Configure import operations" dialog opens. Select the "Overwrite" operation for each element in this import dialog. To start the import, press **Finish**.
The required files were copied, the project is in simulation mode and can be run on the control.

Testing project



The project contains "PlcProg" in CFC and "PlcProg_ST" in Structured Text (ST).

To test the project in simulation mode, refer to [chapter "Testing the application in simulation mode, PlcProg in ST" on page 54](#) or [chapter "Testing the application in simulation mode, PlcProg in ST" on page 54](#).

To test the project with a control, refer to [chapter 2.6.8 "Testing the application on the control" on page 60](#).

3 First steps - Motion

3.1 Project goal "First steps - Motion"

A virtual axis to run at a constant velocity is specified. This axis functions as master axis for a real axis coupled via a gear.

The real axis contains, for instance, an "IndraDrive M" drive connected to the control via a Sercos III cable.



If no drive is available, the step "Add Sercos III drives (real axes) to the project" is omitted.

Create a second virtual axis instead of the real axis and use the virtual axis instead of the real axis.

Subtasks

- Create an IndraWorks project, control and drives
- Load the Motion project from the control
- Program the motion sequence
- Load the PLC program on the control and test it
- Record the motion of the axes using trace



The motion sequences can either be programmed in Function Block Diagram (FBD) or Structured Text (ST).

- [chapter 3.4 "Programming the motion sequence in function block language \(FBD\)" on page 91](#)
- [chapter 3.5 "Programming the motion sequence in Structured Text \(ST\)" on page 113](#)

3.2 Technical requirements

- Software:
 - PC with installed IndraWorks Engineering for IndraMotion MLC 14VRS
- Control:
 - IndraMotion MLC L25
 - IndraMotion MLC L45
 - IndraMotion MLC L65
 - IndraMotion MLC L75
 - IndraMotion MLC XM2
 - IndraMotion MLC VPx

Meeting these prerequisites is described in:

- IndraMotion MLC 14VRS Functional Description, chapter "Commissioning and operation"

A control, e.g. of type "**IndraMotion MLC L65**" has to be available with the current firmware 14VRS. Additionally, the following requirements have to be met:

- The IP address is known
- The control is connected to the programming device (PC) via Ethernet

Drive controller, e.g. type "**IndraDrive M**" with current firmware:

First steps - Motion

- For a real axis:
The drive controller is connected to the control via Sercos III
- The address of the drive controller at the Sercos bus is scanned
or
- a bridged Sercos bus if there is no drive

3.3 Creating the Motion subproject

3.3.1 Resetting the control (CMLx5)

From previous use, the control can contain old program components, parameterizations and PLC data that are removed in this step by clearing the memory.

After the control is switched on, press the outer buttons (<Esc> and <Enter>) below the display and when **BOOT 1.02** is displayed, keep them pressed until **BOOTSTOP** is displayed.

Subsequently, press the two inner keys. **CLEANUP?** is displayed.

Confirm with <Enter>. **CLEANUP** is displayed. If the clean-up was completed successfully, **SUCCESS** and subsequently **REBOOT** are displayed.

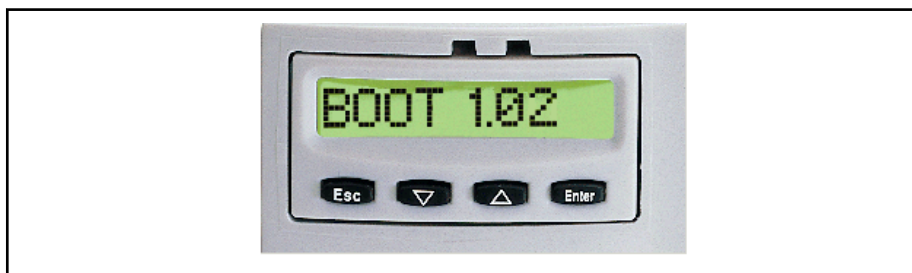


Fig. 3-1: Rebooting the control



The error "F0140001, Device command configuration incorrect" can sporadically occur and has to be acknowledged with <ESC>.

The control runs up to **P2 RDY**. The memory is cleared and any possible Motion component present in phase P2 and the PLC component are ready for operation.

3.3.2 Starting IndraWorks

Start IndraWorks via **Start ► All Programs ► Rexroth ► IndraWorks ► Engineering**.

The IndraWorks engineering environment is started. IndraWorks can also be

started via the desktop icon .

3.3.3 Creating a new project

To create a new project, click on "Create an empty project" in the workspace.

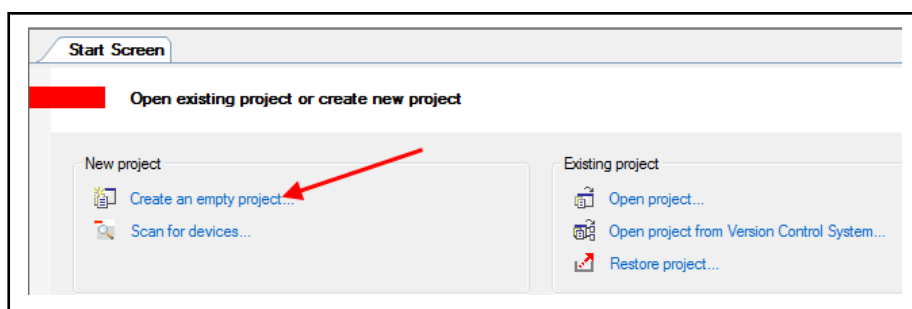


Fig. 3-2: Creating empty project

A new project can also be created via **File ► New ► Project...**

The wizard to specify the project name and the storage location settings for a new IndraWorks project starts.

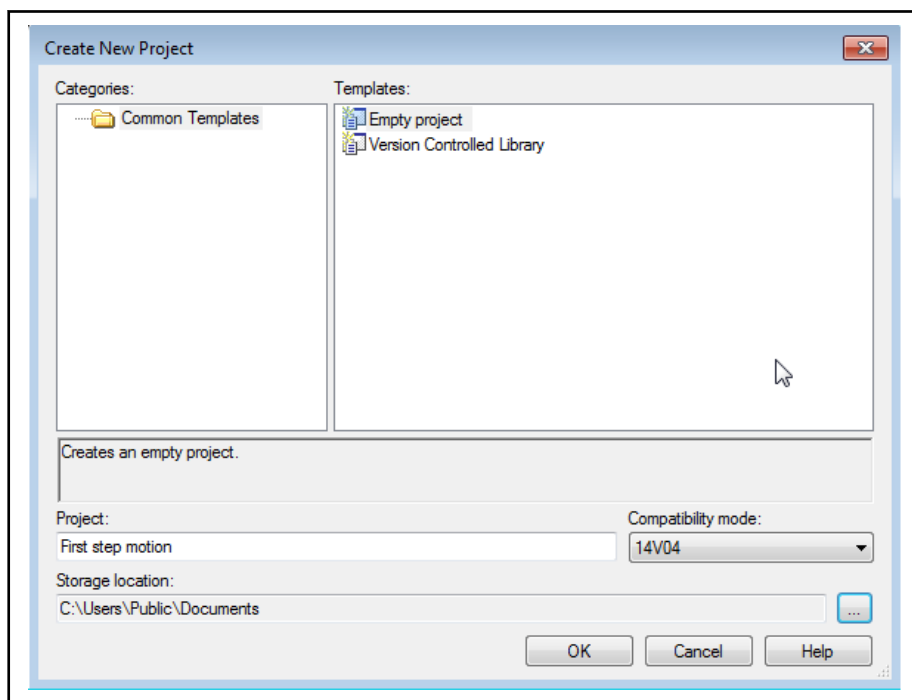


Fig. 3-3: Settings to save a new IndraWorks Project

- Project name** Specify a name for the project. Suggestion: "First steps Motion".
 - Storage location** The field contains a predefined storage location in the directory "C:\Documents and Settings\username\MyDocuments".
A different directory can also be specified.
 - Compatibility mode** The IndraWorks interface scope corresponds to 14V04.
 - Project language** When selecting the project language, the language of the user texts is determined.
- The newly created project is displayed in the Project Explorer with the previously defined project name:



Fig. 3-4: "First steps Motion" project

First steps - Motion

3.3.4 Adding a new device to a project

Drag the device (e.g. folder "IndraMotion MLC", "IndraControl L65") from the "Drive and Control" library and drop it onto the newly created project in the Project Explorer.



(1) Device: IndraMotion MLC L65

(2) Project: First steps Motion

Fig. 3-5: Adding a device

Subsequently, a wizard starts automatically to make the necessary basic settings as described below.

3.3.5 Specifying basic settings for the device

Device settings

Enter the general information on the device to be added.

Fig. 3-6: Device settings

- Device name** Enter any device name, e.g. "Mlc1" in "Device name". By default, the field includes the name from the library. The device is subsequently displayed in the Project Explorer under the specified name.
- Comment** Add a comment that describes the device in detail. This field is optional.
- Author** Enter the name of the project author. This field is optional.



These entries can also be changed later on.

Enter the name of the project author. This field is optional.

Communication settings

In the "Configuration" dialog, define the settings for the firmware and communication between IndraWorks and the device (in this case: IndraMotion MLC L65).

Fig. 3-7: Creating a control (shown for the "IndraMotion MLC L65") - Hardware/communication

- | | |
|--------------------------|--|
| Device type | The control selection predetermines the device type. If there are several different variants for one device type, select the device variant from the drop-down list. |
| Firmware version | Select the firmware version here. This setting can generally remain. |
| Firmware release | The firmware release in a firmware version is set here. This setting should also remain. |
| IP address | Enter the target address under "Device IP address", i.e. the IP address of the control to which the application is to be loaded. |
| PLC gateway | In "Gateway IP address", define the address of the computer on which the gateway server is running. By default, "localhost" is provided as computer name (address), "localhost" means that the locally installed gateway is addressed. Do not change this setting. |
| PLC communication | In 14VRS, only TCP is still supported at the Engineering interface.

Due to compatibility reasons, UDP is provided for projects of 13VRS. |
| Connection test | Click on Execute to check whether the communication between computer and target control is successful. The test result is logged in the "Connection test" field.

If the PLC communication test was successful, the IP address is shown.
Any possible problems are signaled by flashing red or yellow icons in the window.

The following problems can occur: <ul style="list-style-type: none"> • Output: "Communication to the control <Control name> with the IP address <IP address> failed!" |

First steps - Motion

Causes:

- The control is not switched on
- The network cable is not connected
- The IP address of the control and the IP address set do not match

Solution: Change the IP address in the dialog or the IP address of the control (see IndraMotion MLC 14VRS Functional Description, chapter "Commissioning and operation", subchapter "Setting the IP address")

- The network mask of the PC and/or control is incorrectly set

Solution: Change the IP address in the dialog or the IP address of the control (see IndraMotion MLC 14VRS Functional Description, chapter "Commissioning and operation", subchapter "Setting the IP address")

- Output: "The configured firmware and the firmware in the control do not match."

Causes:

- The control firmware is not up-to-date

Solution: Update firmware (see IndraMotion MLC 14VRS Functional Description, chapter "Control", subchapter "Firmware management")

- The selected control does not correspond to the real control type

Click on **Next >>** to go to the next page.

PLC characteristics

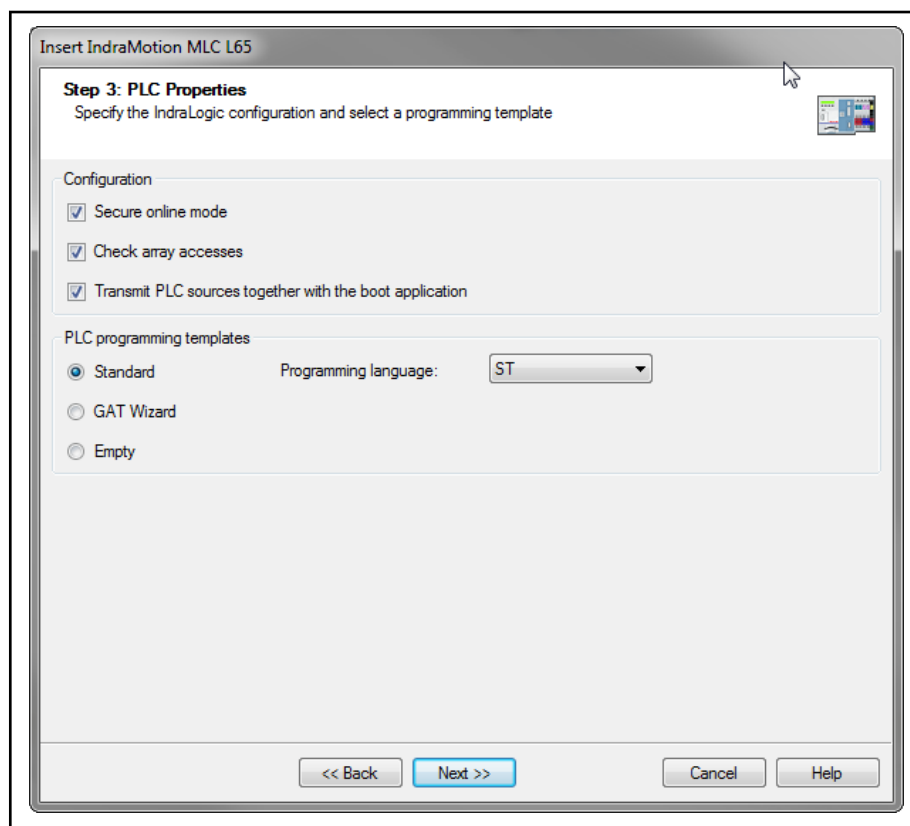


Fig. 3-8: Creating a control - PLC properties

Secure online mode	If the option "Safe online mode" is selected, a prompt is displayed asking for confirmation before any security-relevant actions are executed.
Checking array access	If this switch is enabled, the check function "CheckBounds" from the template database is automatically inserted below the PLC application node. If this switch is disabled again in case of an already existing control, the check function "CheckBounds" is only automatically removed if the following comment block is still present in the implementation section of the check function: <pre>(***** This function was generated by the 'check array access' option of the "Properties" dialog of the controller. If you delete this comment, this function is not removed by disabling the 'check array access' option. *****)</pre> To ensure that customized adaptations are not deleted in the implementation section of the check function when the switch is disabled, remove the comment block.  As the automatically inserted check function is not removed anymore when deleting the comment block, the "CheckBounds" function has to be deleted manually from the application to disable the monitoring of the array limits. A disabled switch does not mean that no CheckBounds functionality is active. If the check function is inserted or deleted, login via online change is not possible. A download is required. The checkbox is only selected if the PLC application type "Standard" or "GAT Wizard" is enabled and if there is no user-defined CheckBounds function.
Transferring PLC sources together with the boot application	When transferring the boot application, PLC program sources (POUs, tasks, etc.) are transferred to the control in addition to the compilation. Use this function to additionally display a detailed comparison for the PLC elements during online comparison.
Programming templates	Select between the following variants for the PLC application to be created: Default: The empty default programs "MotionProg" and "PlcProg" are automatically created in the selected programming language. For other PLC objects, the programming language is selected when creating the object. PLC programming language: Set the preferred PLC programming language. Select "ST" (Structured Text). GAT wizard: Example projects or programming templates can be selected in an additional dialog. GAT is a programming frame ready for operation with predefined locations for the application code. Professional and well-structured application programs with prepared elements are automatically created using the dialog-based code generation of the GAT Wizard. Empty: Creates an empty, non-executable PLC application without programs and tasks. Click on Next >> to go to the next page.

First steps - Motion

Selecting function package



Due to the acquired firmware, the function packages are predefined and can thus not be selected for the controls based on the IndraControl L25/L45/L65/L75. For the controls based on IndraControl XM and VPx, select the function packages according to the purchased licenses!

1. If a function package is deselected, all function packages dependent on that function package are also deselected.
2. If a function package, which is subject to license, is selected, a license icon is automatically generated. This icon displays license information via tooltip or mouse click.
3. Depending on the activated function packages, the nodes below the created control are created in the IndraWorks project tree.
4. Even after creating a control, the function packages can be added or deleted via the corresponding "Properties" dialog.
5. When converting the device to a control type with a different functional scope, a conversion preview informing on the upcoming changes is still displayed.



If the function package "Motion" is removed for example, all configured axes are also deleted.

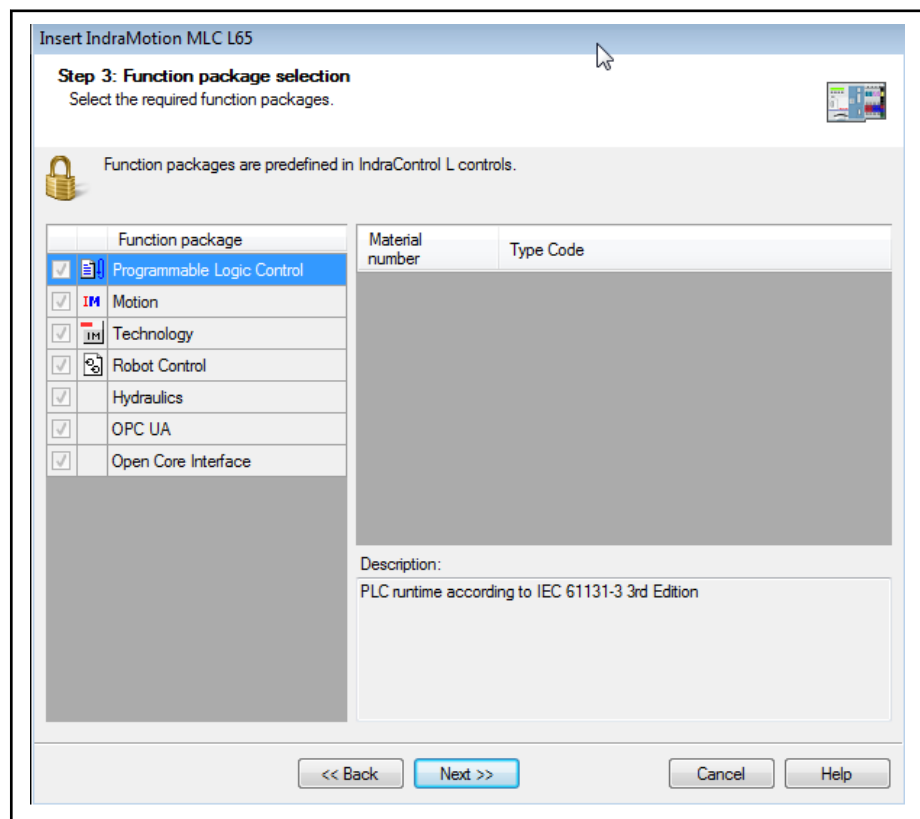


Fig. 3-9: Overview on the acquired function packages

Programmable Logic Control

The function package "Programmable Logic Control" includes the PLC runtime system based on the IEC 61131-3 3rd edition. The function package is enabled by default and cannot be deselected.

First steps - Motion

Motion	Firmware-based control of single-axis and synchronized multiple-axis motions.
Technology	PLC libraries for technology functions.
Robot Control	General kinematics and transformations to control robots.
Hydraulics	Central control of hydraulic axes.
OPC-UA	OPC UA server, OPC unified architecture, standardized variable exchange between control and visualization.
Open Core Interface	Interface for Rexroth Open Core Engineering. The programming interface called Motion Logic programming interface (MLPI/OCI) allows to access functionalities encapsulated in the firmware of the IndraMotion MLC system. Thus, it is easy and fast to develop Motion Control applications in the programming languages C/C++/C#/VBA/JAVA using the engineering environment "Visual Studio" (see Engineering network).

Function module settings



Depending on the device type, the "Interfaces" dialog can differ. In this project, the dialog for the "IndraMotion MLC L65" device is exemplarily described.

Enter the device and function module settings.

Fig. 3-10: Function module settings

Profibus DP (X7P)	Current Profibus configuration that can be selected when adding a device. "Profibus DP master", "Profibus DP slave" or "Not used" can be selected.
Real-time Ethernet (X7E3/X7E4)	Current Profinet or Ethernet/IP configuration that can be selected when adding a device. "Not used" or a Profinet or Ethernet/IP interface can be selected.
Ethernet configuration (X7E5)	As this interface is needed as Engineering interface (connection directly to your PC or via the operating network), the Ethernet/IP Adapter may not be enabled.
Function modules	Up to four function modules can be connected to an "IndraMotion MLC L65/L75 / IndraLogic XLC L65/L75". The function modules can be selected from the drop-down list if required.

First steps - Motion

To confirm the entries, press **Finish**.

The newly created device is displayed in the Project Explorer.

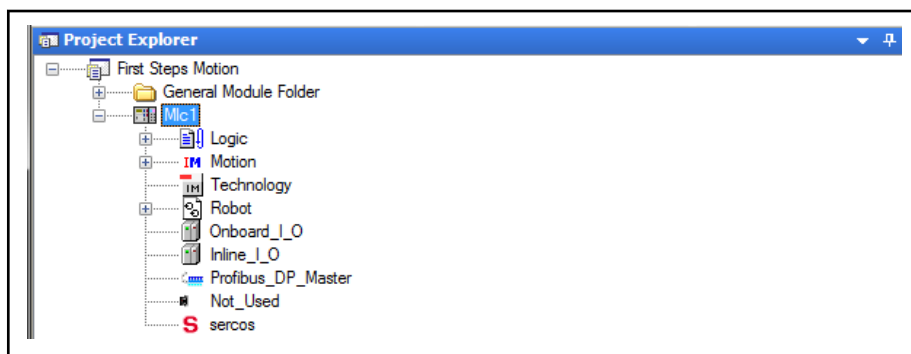


Fig. 3-11: Project Explorer: The "First steps Motion" project with an IndraMotion MLC L65 control with Sercos III bus



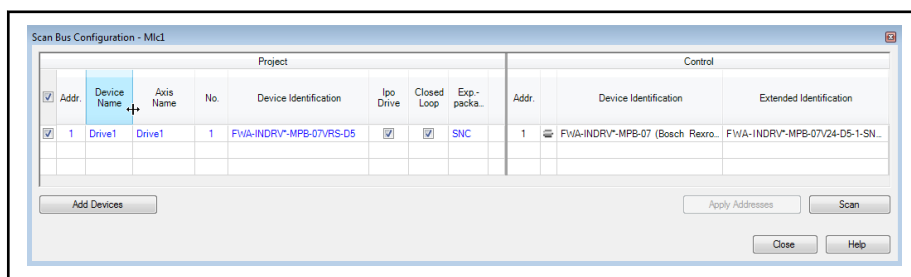
The Project Explorer mirrors the components selected in the "Configuration" and "Function modules" dialogs.

3.3.6 Adding Sercos III drives (real axes) to the project

The simplest way to determine the data for all the connected drives is to scan the Sercos III bus (context menu "Sercos" node **Scan bus configuration**).

For scanning, IndraWorks switches the control to parameterization mode (P2) after a prompt.

The result is shown in the following:



Addr. Sercos address

Device name Default drive_n, renamed

Axis name Name of the axis / I/O device

No. Axis number

Device ID Drive family / firmware ID

Closed-loop Position control possible

Ipo drive Interpolation in the control or drive, in this case in the control

Fig. 3-12: List of the Sercos drives, device name "Drive1" can be edited

Press **Add devices**.

The devices selected in column 1 from the scanned bus configuration are applied to the Project Explorer.

The drive is mirrored as "real axis" in the Motion node.

Use axis number "1" and axis name "Drive1" with regard to the addressing type.

The project includes the control "Mlc1" e.g. of type "IndraMotion MLC L65" and the real axis "Drive1" (folder **Motion** ► **Real axes**) or drive "Drive1" (folder **Sercos**):

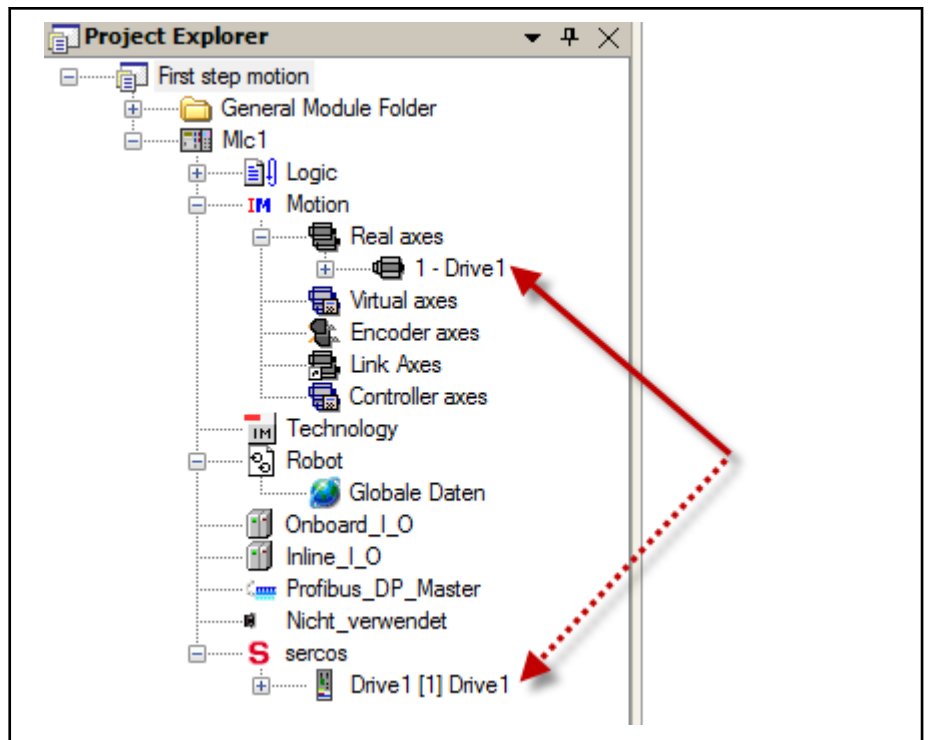


Fig. 3-13: Drive/real axis "Drive1" is added to the project

3.3.7 Adding a virtual axis to the project

Unlike real axes that can run on a variety of interfaces, e.g. Sercos or Profibus and that have to be declared or scanned accordingly, a virtual axis only has to be declared. In the Project Explorer, right-click on the folder **Motion** ► **Virtual axes** and select the context menu item **Add**.

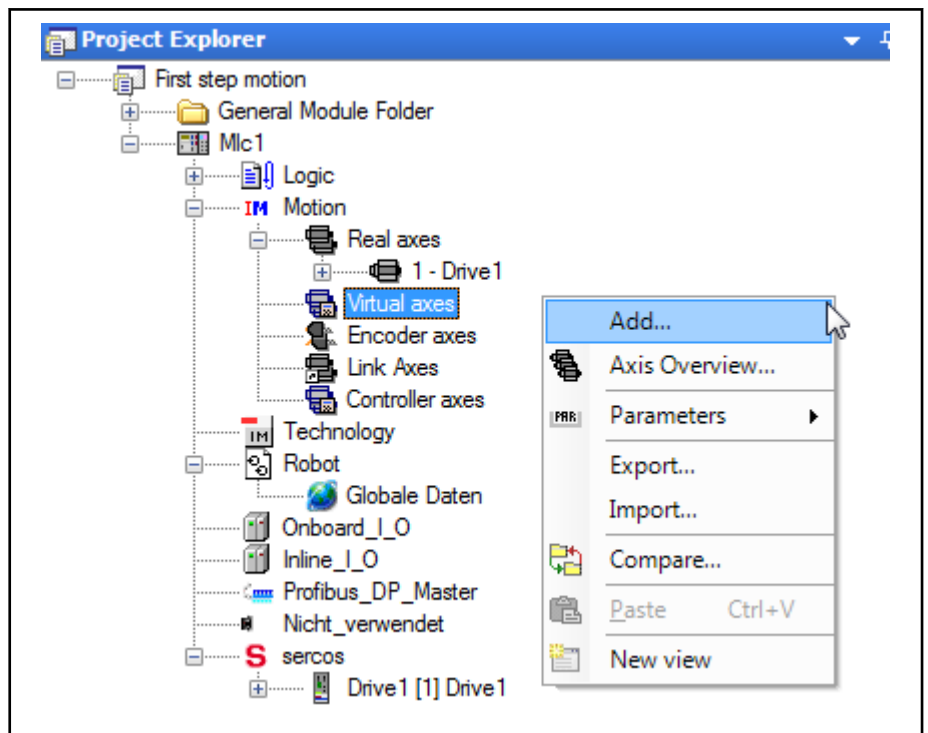


Fig. 3-14: Declaring a virtual axis

First steps - Motion

Similarly to the creation of the control, a wizard provides the required dialogs for filling out.

Fig. 3-15: "General settings" dialog for a virtual axis

Fig. 3-16: "Configuration" dialog for a virtual axis

Use axis number "2" and axis name "vAxis1" with regard to the addressing type. Click on **Finish**.

3.3.8 Downloading the Motion subproject

The first processing part is completed after creating the control and all axes in the "First steps - Motion" project and parameterizing control and axis components.

Load the information collected up to now to the control. Therefore, select the device (Mlc1) and click on the icon of the toolbar "Project" or on the menu item **Project ► Switch device online**:

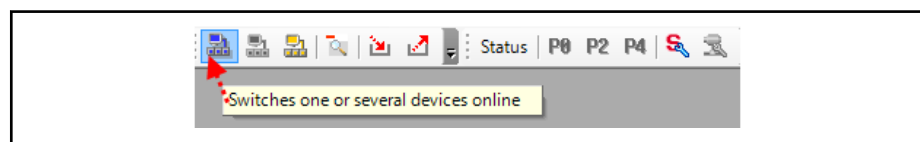


Fig. 3-17: Switching device online

Data is now exchanged between the IndraWorks user interface and the active control.

(1): The IndraMotion MLC is **partially synchronized** and expects the download of the Motion subproject.

(2): The Sercos field bus runs in phase P2 (parameterization mode), visible at the Sercos node and at the icon of the "Sercos" toolbar

First steps - Motion

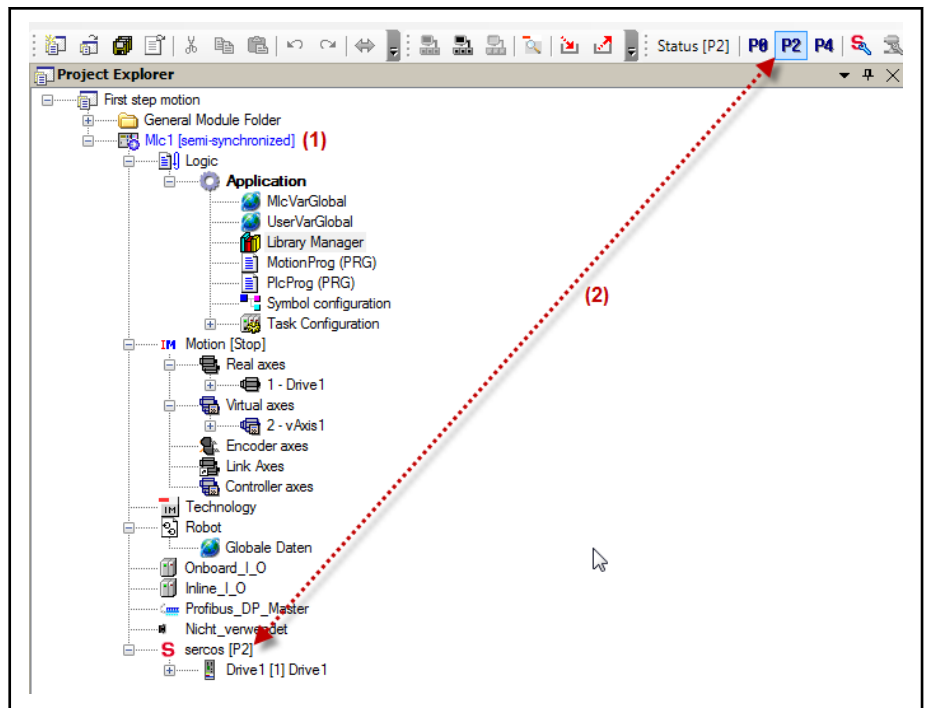


Fig. 3-18: Partially synchronized IndraMotion MLC with Sercos III field bus in P2
To synchronize control and project, synchronize the axis configuration data (refer to the following figure).

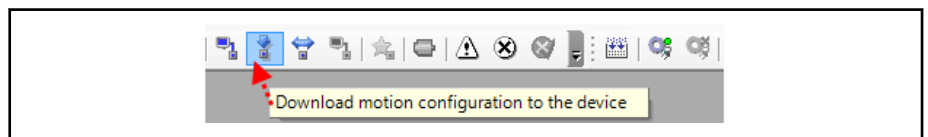


Fig. 3-19: Downloading the Motion configuration

Use the following dialog to transfer project data to the control ("use the IndraWorks axis configuration") or to apply control data to the project ("use the axis configuration of the control"). Press **OK** to apply the selected axes.

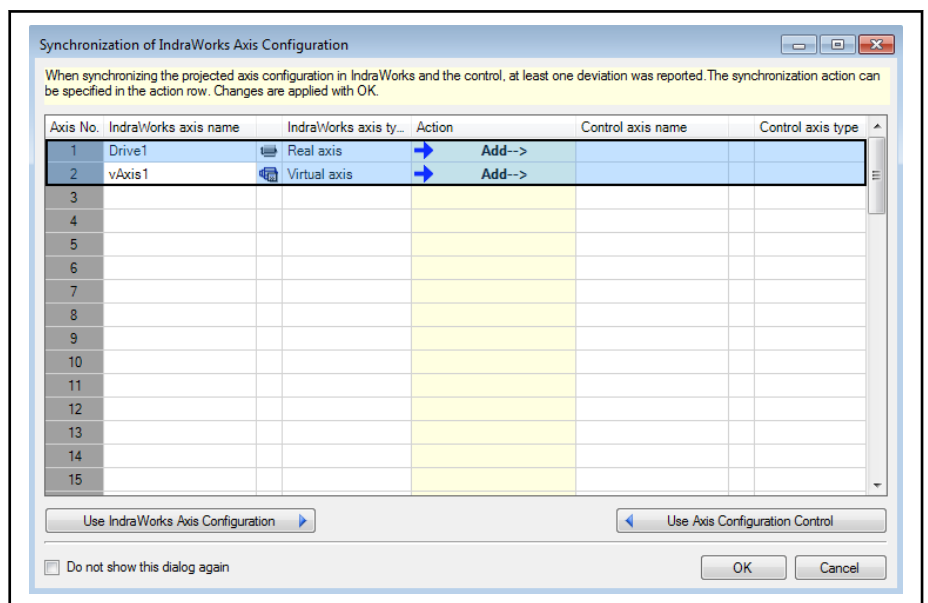
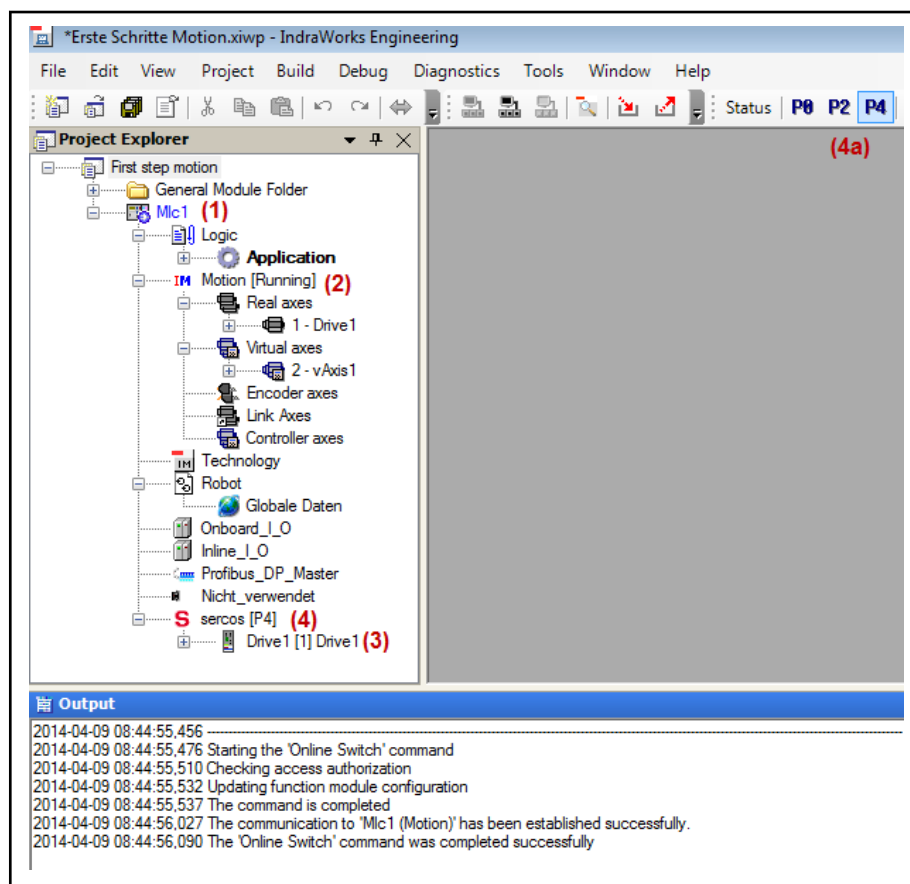


Fig. 3-20: "Synchronization of IndraWorks axis configuration" dialog
The control is now synchronized.

First steps - Motion



Without this download step, the data determined in the background is missing in the control.



- (1) Active control (online)
 - (2) Motion component runs without errors
 - (3) Active drive (online)
 - (4, 4a) Communication in "P4" mode, operating mode
- Fig. 3-21: Download completed successfully (output)



The "online" information in the footer is only visible if the desired control or one of its subfolders is active.



Before switching to BB (ready for operation), the parameterization of the drives (axes) required in P2 has to be completed with plausible values. Otherwise, phase-switching errors result that do not allow to reach phase BB and require another phase switching to P2 (plausibility check during phase switching by drives and control).

This generally applies if an attempt is made to switch the control to BB or drives to Ab (drive ready).



If the operation mode of the drive was switched from IndraDrive (calculation in the drive) to IndraDrive (calculation in the control) or vice versa, an error message can be output when changing from P2 to BB. In this case, the basic parameters of the drive have to be reloaded. Load basic parameters subsequently via the context menu item **Parameters ▶ Load basic parameters...** of the axis (Drive1).

If the control is in "parameterization mode (P2)", switch the control into "operating mode (BB)" (context menu of the Sercos node **Sercos state ▶ Phase P4** or Sercos toolbar, icon **P4**).



Depending on the motor encoder type, an error "F2174, Reference loss by motor encoder" can result.

The error can either be acknowledged with <ESC> at the display of the drive and the control or even better, it can be removed via **Mlc1 ▶ Motion ▶ Real axis ▶ Drive1 ▶ Motor, brake, measuring systems ▶ Reference motor encoder "Set absolute position"**.

Press the "Clear error" button .

The drive is now in "Ab" status (drive ready).

The control and the Motion component are in "BB" (ready for operation), the PLC component is in "STOP" (PLC stopped) or "RDY" (ready).

3.4 Programming the motion sequence in function block language (FBD)

3.4.1 General information

The motion sequence is programmed in a PLC program using the PLCopen function blocks (programming system IndraLogic 2G acc. to EN 61131-3).

When creating a project using the "Mlc1" control, the real axis "Drive1" and the virtual axis "vAxis1", the first actions to program the motion sequence were started below the "Logic" folder:

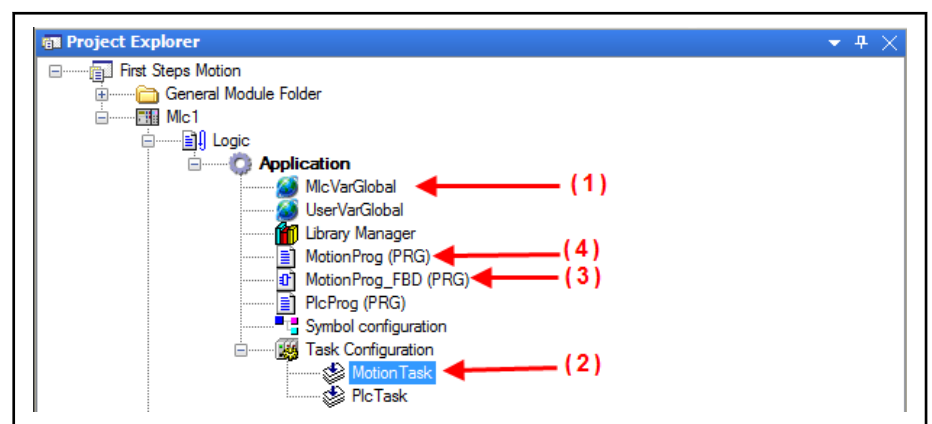


Fig. 3-22: Project Explorer after expanding the "Logic" node

The following programs were created:

- **MotionProg (4)** (preselection of ST, Structured Text)
- **PlcProg** (preselection of ST, Structured Text), not used

First steps - Motion

The third program, **MotionProg_FBD (3)**, was manually supplemented: **Application**, right-click, **Add ► POU**, enter the name and select FBD as programming language.

The programs are still empty.

The following tasks were created under "Task configuration":

- **MotionTask (2)**, (priority 2, externally event-triggered by the Sercos task, watchdog active) for "MotionProg".

The program(s) running under this task, has (have) to be processed completely within the specified time period of a Sercos cycle. If they are not within this part, the control is stopped

- **PlcTask**, (low-priority cyclic task) for PlcProg, not used.

In the remaining time, the program is processed at least piece by piece, that means during several cycles, if required

Global variables/global constants were created:

- **UserVarGlobal**, reserved for the user, not used in "First steps - Motion"
- **MlcVarGlobal (1)** includes the address constants for the axes "Drive1" and "vAxis1"

Program:

```
(* Automated generated code by MLC. *)
(* Please don't edit, the code will be overwritten...*)
VAR_GLOBAL CONSTANT
(*virtual axis "vAxis1" with axis number 2*)
vAxis1: AXIS_REF := (CntlNo:=LOCAL_CNTRL,AxisNo:=AXIS_2);
(*real axis "Drive1" with axis number 1*)
Drive1: AXIS_REF := (CntlNo:=LOCAL_CNTRL,AxisNo:=AXIS_1);
(*...*)
END_VAR
```



Only the MotionTask is required for the "First steps Motion" project.

According to the programming language required, either **MotionProg** or **MotionProg_FBD** is assigned.

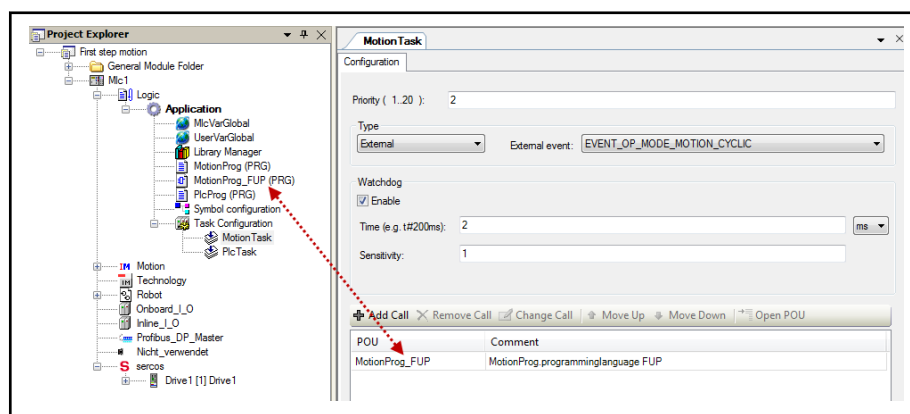


Fig. 3-23: Deleting call: MotionProg. Add call: MotionProg_FBD

3.4.2 Programming the virtual axis "vAxis1", programming language FBD



To graphically present your programming in FBD (Function Block Diagram), continue in this section.

If ST (Structured Text) should be used, refer to section [chapter 3.5.2 "Programming the virtual axis vAxis1, programming language ST" on page 115.](#)

First steps - Motion

Similarly to [chapter 3.1 "Project goal First steps - Motion" on page 77](#), instances of the PLCopen blocks "MC_MoveVelocity" and "MC_Stop" are called using "vAxis1".



The instance of the "MC_Power" function block is not required for virtual axes.

Proceed as follows:

1. Assign the "MotionTask" to the "MotionProg_FBD" program.
2. Left double-click to open the "MotionProg_FBD" program.

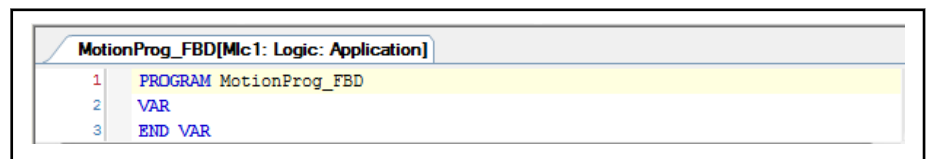


Fig. 3-24: New program in FBD (function block diagram)

Right-click to insert the first function block into the implementation, network 1:

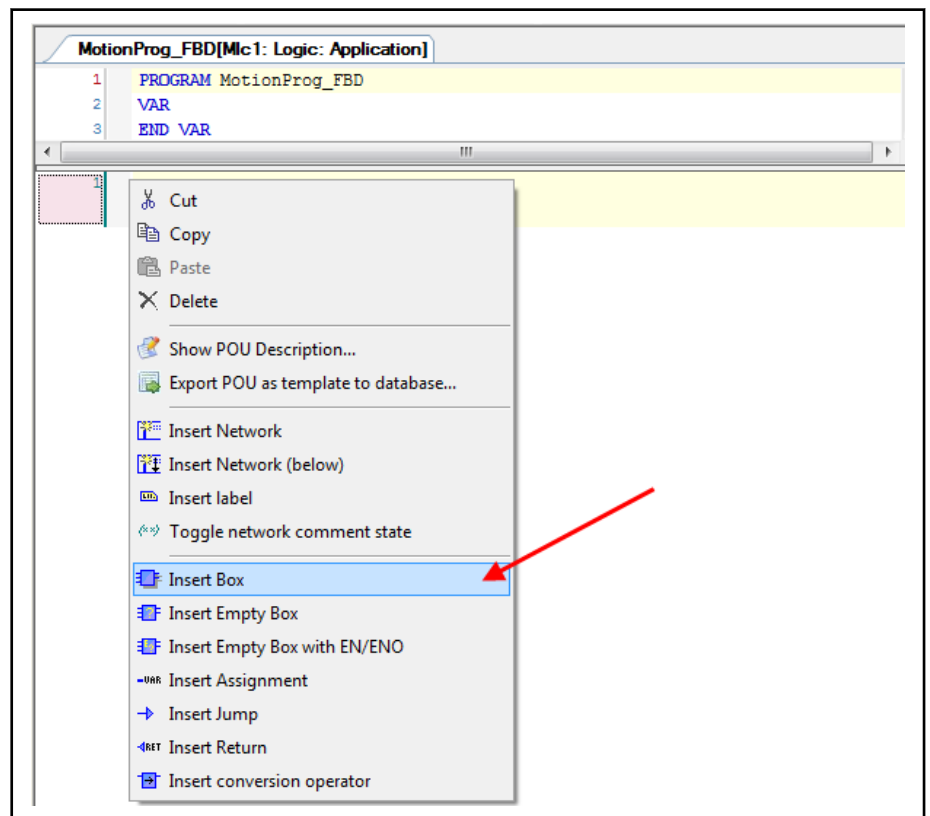


Fig. 3-25: Generating a new function block

Therefore, select the following in the open selection window:

Function blocks ► ML_PLCOpen ► POU ► Motion ► MC_MoveVelocity.

First steps - Motion

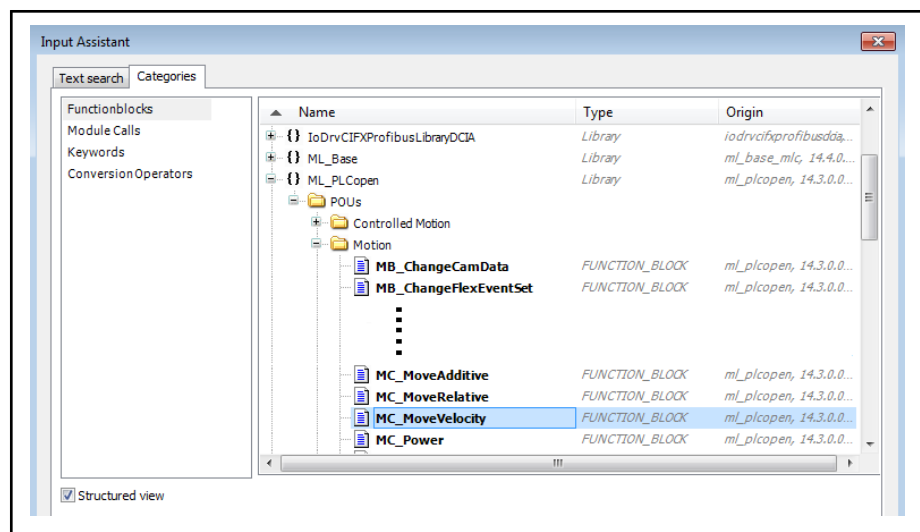


Fig. 3-26: Selection window for function block types

To confirm the insertion of the function block "MC_MoveVelocity", press **OK** and supplement the instance name "MC_MoveVelocity_0" suggested in the "MotionProg_FBD" program above the function block by "MC_MoveVelocity_vAxis1" and press <Enter> to close the entry. The "Variable declaration" window opens. This can be immediately acknowledged with **OK**:

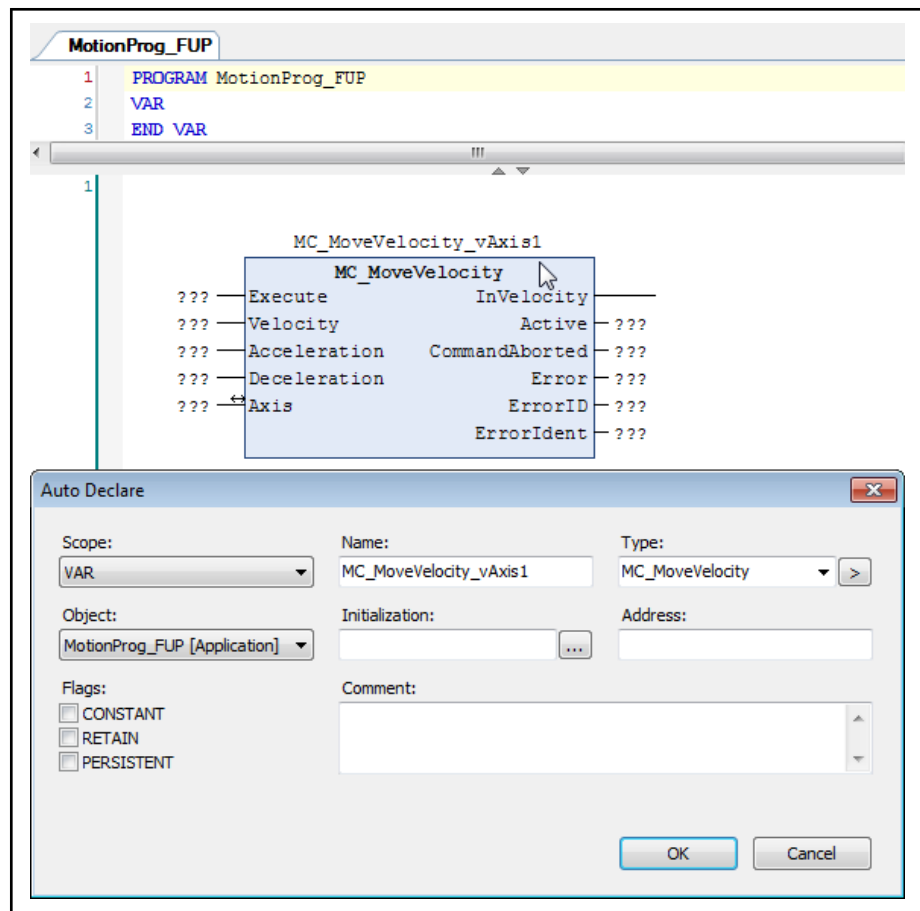


Fig. 3-27: Declaration of the instance name of a function block, type "MC_MoveVelocity" using the "Variable declaration" window

The "MC_MoveVelocity_vAxis1" instance of the PLCopen function block "MC_MoveVelocity" is thus declared. Assign the function block inputs.

First steps - Motion

As for the instance name of the function block, assign the "Execute" input with a variable called "ex_MoveVel_vAxis1" of type "BOOL":

- Highlight the three question marks in front of the "Execute" input
- Enter the name "ex_MoveVel_vAxis1" and press <Enter>
- Confirm the open "Auto declare" window with <Enter>

To assign the "Velocity" input, proceed as follows:

- Highlight the three question marks in front of the "Velocity" input
- Enter the name "Vcom_vAxis1" and press <Enter>
- Enter "REAL" as type and 200 (rpm) as initial value and press <Enter>
- Enter 20 (rad/s²) for the inputs "Acceleration" and "Deceleration"



Enter the values directly at the function block inputs.

Alternatively, a variable with the respective value can be used.

To assign the "Axis" input, proceed as follows:

- Click on ??? and select the adjacent input assistant :
 Select **Variables** ► **Application** ► **MlcVarGlobal** ► **vAxis1**, Insert with namespace prefix.

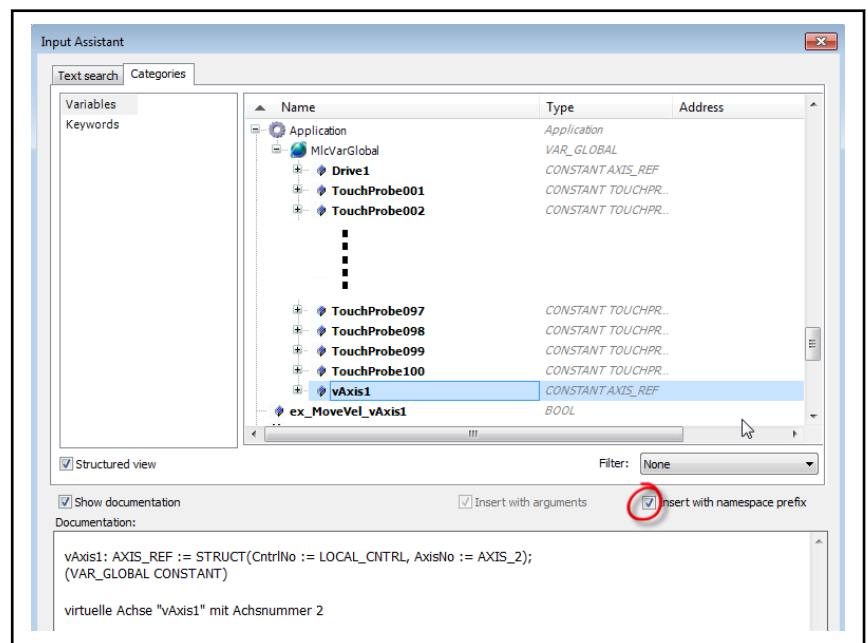


Fig. 3-28: Selecting the axis instance when assigning a function block of the type "MC_MoveVelocity"

- Confirm with **OK**.
- To complete the entry, press <Enter>.



The name of a global constant consists of the folder name and the element name:

MlcVarGlobal.vAxis1

For feedback whether the function block operated correctly, its outputs are assigned.

First steps - Motion

Click to highlight the area on the right next to the output "InVelocity" and select the following in the context menu:

Insert assignment:

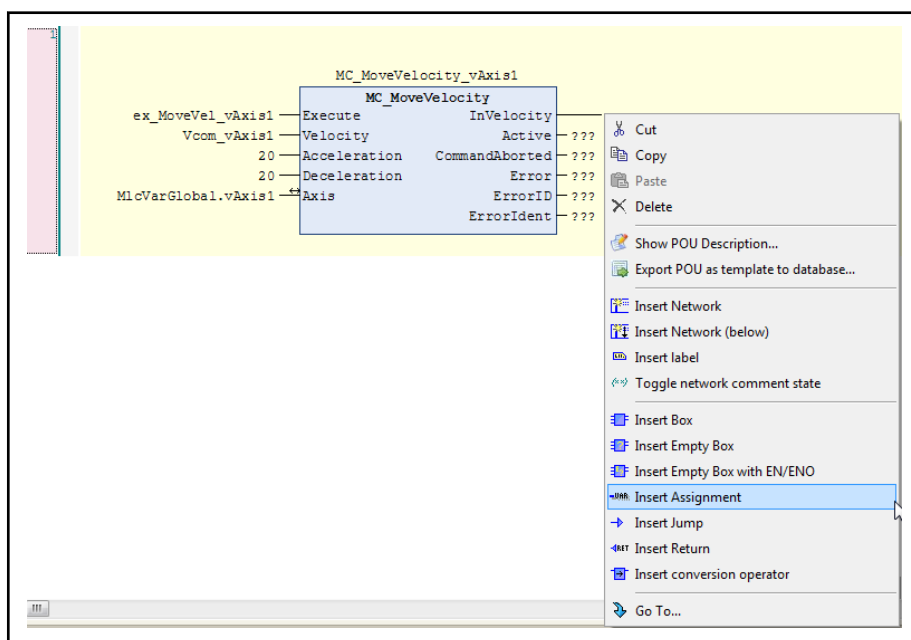


Fig. 3-29: Assignment at the function block output

Assign the "InVelocity" output of the function block as usual using the "in_MoveVel_vAxis1" variable, type "BOOL":

- Highlight the three question marks **???** at the output "InVelocity"
- Enter the name "in_MoveVel_vAxis1" and press <Enter>
- Confirm the open "Auto declare" window with <Enter>
- Assign the "Active" output with "ac_MoveVel_vAxis1"
- Assign "ca_MoveVel_vAxis1" to the "CommandAborted" output
- Assign the "Error" output with "er_MoveVel_vAxis1"
- Assign the "ErrorID" output with "ID_MoveVel_vAxis1"
- Assign the "ErrorIdent" output with "Ident_MoveVel_vAxis1"

The PLCopen function block of type "MC_MoveVelocity" is thus assigned according to its task. The declaration section was automatically completed.

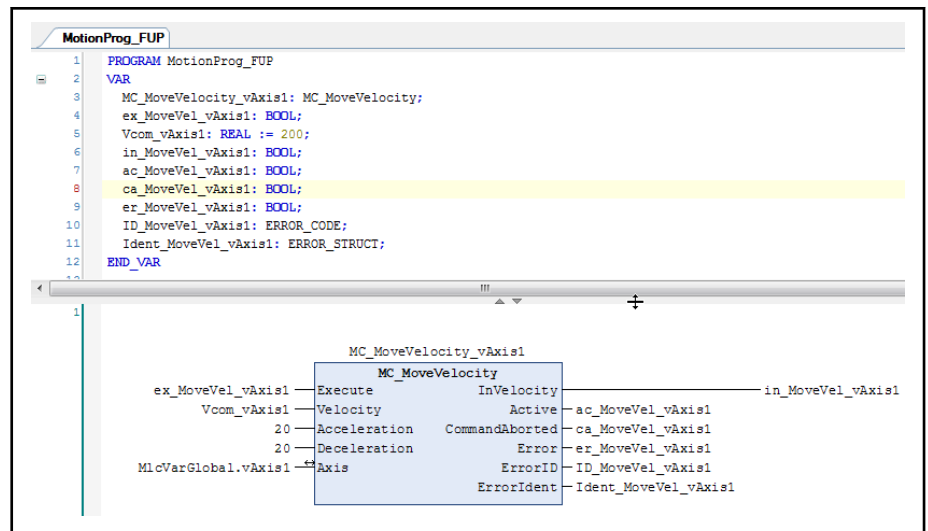


Fig. 3-30: PLCopen FB, "MC_MoveVelocity" type is inserted and assigned



Whenever a function block is completed, run a test compilation using <Shift>+<F11>. Clear any errors immediately.

Generate an instance of the PLCopen function block "MC_Stop".

MC_Stop

Insert MC_Stop and declare an "MC_Stop_vAxis1" instance

To add a new network, right-click on network 1

Insert network (below)

Insert "MC_Stop" and declare an "MC_Stop_vAxis1" instance

Assign inputs and declare input variables if necessary:

"Execute" → "ex_Stop_vAxis1" of type "BOOL"

"Deceleration" → 50

"Axis" → "MlcVarGlobal.vAxis1" of type "AXIS_REF")

Outputs:

"Done" → "do_Stop_vAxis1" of type "BOOL"

"Active" → "ac_Stop_vAxis1" of type "BOOL"

"CommandAborted" → "ca_Stop_vAxis1" of type "BOOL"

"Error" → "er_Stop_vAxis1" of type "BOOL"

"ErrorID" → "ID_Stop_vAxis1" of type "ERROR_CODE"

"ErrorIdent" → "Ident_Stop_vAxis1" of type "ERROR_STRUCT"

Actual velocity "Vact_vAxis1"

The final step is the implementation of a network in which the current velocity of the axis "vAxis1" "Vact_vAxis1" (data type "REAL") can be read.

The current velocity is read from the data set of the axis (shared memory).

Data of all axes that can control an IndraMotion MLC, is included in the array

GVL_Base.AxisData: ARRAY [1..xx¹⁾] OF ML_AXISDATA_SM

The axis number has to be used as index.

1) The number of controllable axes depends on the control type and its firmware, currently for the L65: 64 axes

First steps - Motion

The variable is provided with the name

GVL_Base.AxisData[2].rActualVelocity_i

for the virtual axis, axis number 2, ("Drive1" is provided with axis number 1, as "Drive1" was scanned before "vAxis1" was declared).

- To add a new network, right-click on network 2
Insert network (below)
- **Insert assignment**
- Insert "Vact_vAxis1" on the right ??? and press <Enter>, select the type "REAL" in the declaration window and press <Enter>
- Go to the left ??? and click on [...] using the input assistant to navigate to **Variables ▶ ML_Base ▶ Global Variable List ▶ GVL_Base ▶ AxisData**:

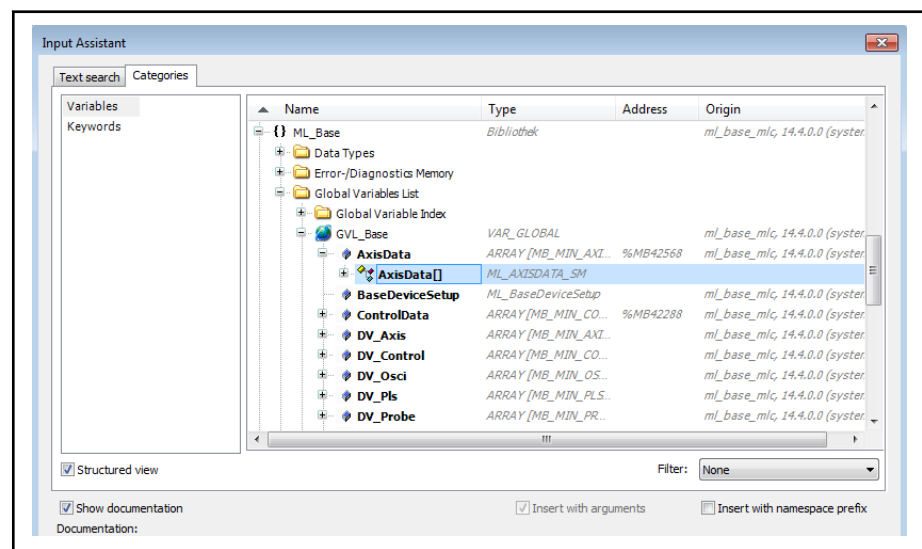


Fig. 3-31: Actual velocity value "vAxis1" from the Global Variable List of the "ML_Base.library"

Further entries are supported by the Intellisense function (automatic completion), i.e. after entering "." at "AxisData[2].", all available elements are shown and can be selected.

Confirm "rActualVelocity_i" with <Enter>.

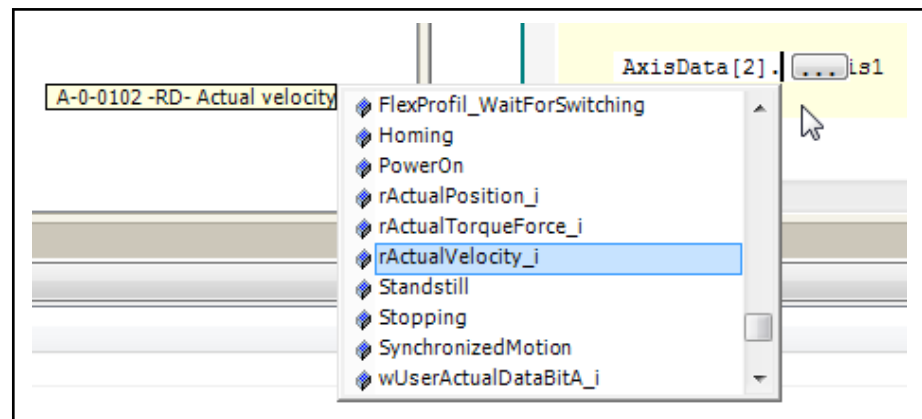


Fig. 3-32: Selecting the variables of the actual velocity

The implementation of the motion sequence according to the task assignment is thus completed:

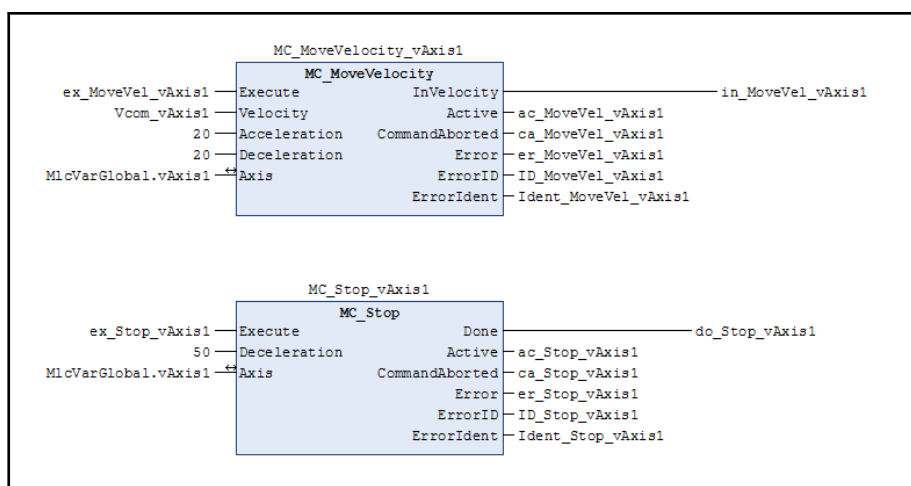


Fig. 3-33: Implementing the virtual axis "vAxis1"



AxisData[].rActualVelocity_i; and
GVL_Base.AxisData[].rActualVelocity_i;
Are equally used by the compiler.

3.4.3 Downloading the PLC subproject "virtual axis" "vAxis1" (FBD)

The previously implemented motion sequence still has to be compiled, applied to the control and started.

For the compilation, go to the via IndraWorks menu item

- **Build ► Build**, compilation of the added part of the application
- **Build ► Generate code**, the command allows to generate a code and to check compilation errors

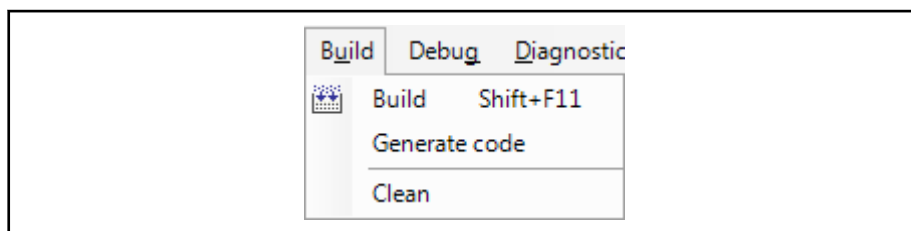


Fig. 3-34: Build application



The menu items **Create ► Recompile** and **Create ► Clean all** are not available by default anymore as in 13VRS. However, these commands can be added again via **Tools ► Customize ► Commands ► Create**.

To transfer the compiled data to the target control, select **Debug ► Login** from the IndraWorks menu bar (refer to the figure below).

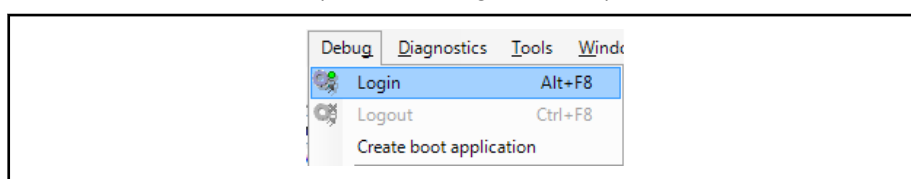


Fig. 3-35: Transferring compiled data to the control

First steps - Motion

A prompt asking for agreement to download is displayed depending on whether an application is already loaded and whether this application is a previous version of the current application.

The application is still stopped after a successful transmission (red in the Project Explorer and in the status bar) and the variables in declaration and implementation are prepared to display the values. However, the values are invalid.

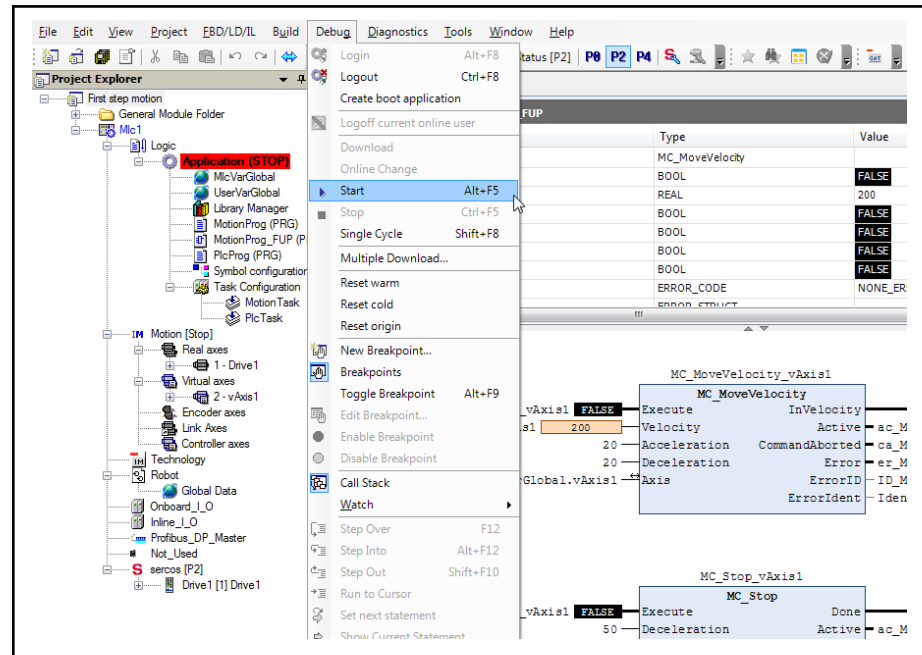


Fig. 3-36: Starting the program processing in the control (1)

Start the processing in the control via **Debug ▶ Start**.

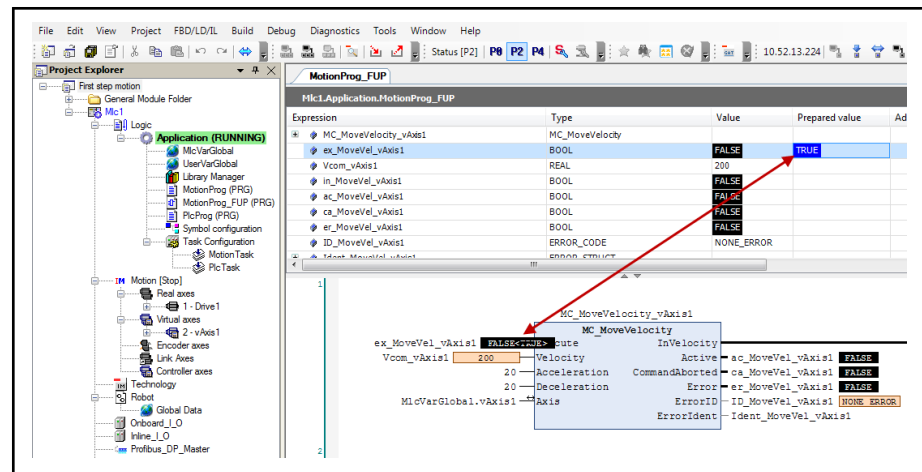


Fig. 3-37: Starting the program processing in the control (2)

The motion process starts automatically, since the processing of the programmed function blocks is subsequently enabled.

"Write values" of input variables

To enable the processing of "MC_MoveVelocity_vAxis1" of type "MC_MoveVelocity", there has to be a positive edge at the "Execute" input of the function block.

Left double-click on "FALSE" behind the input variable "ex_MC_MoveVel_vAxis1". There is a selected "FALSE" now behind "ex_MoveVel_vAxis1"

First steps - Motion

and at first "TRUE" in the declaration line, then "FALSE" after the next click and then an empty space (disabled).

Mlc1.Application.MotionProg_FUP			
Expression	Type	Value	Prepared value
MC_MoveVelocity_vAxis1	MC_MoveVelocity		
Execute	BOOL	FALSE	TRUE
Velocity	REAL	200	
Acceleration	REAL	20	
Deceleration	REAL	20	

Fig. 3-38: "Write variables" in the declaration section of the program

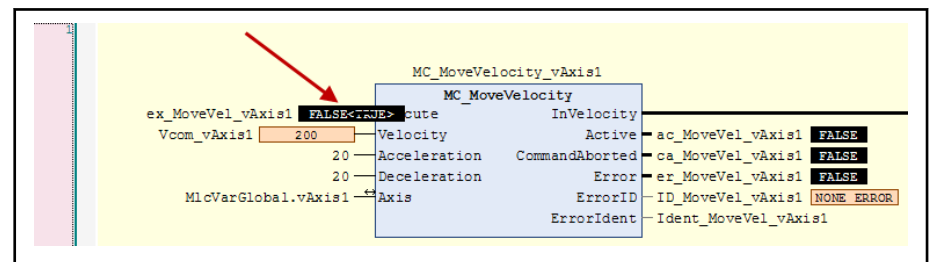


Fig. 3-39: "Write values" at the interface of the function block

Via **Debug** ► **Write values** of <Ctrl>+<F7>, "ex_MC_MoveVel_vAxis1" is set to "TRUE" in the control.



- "Write values" means that the variable is set to the desired value **once** at the beginning of the next cycle
- "Force values" means that the variable is **permanently** set to the desired value at the beginning of the following cycles

The positive edge thus created sets the outputs "InVelocity" and "Active" of the "MC_MoveVelocity_vAxis1" function block to "TRUE" (1).

The axis travels at 200 rpm (2).

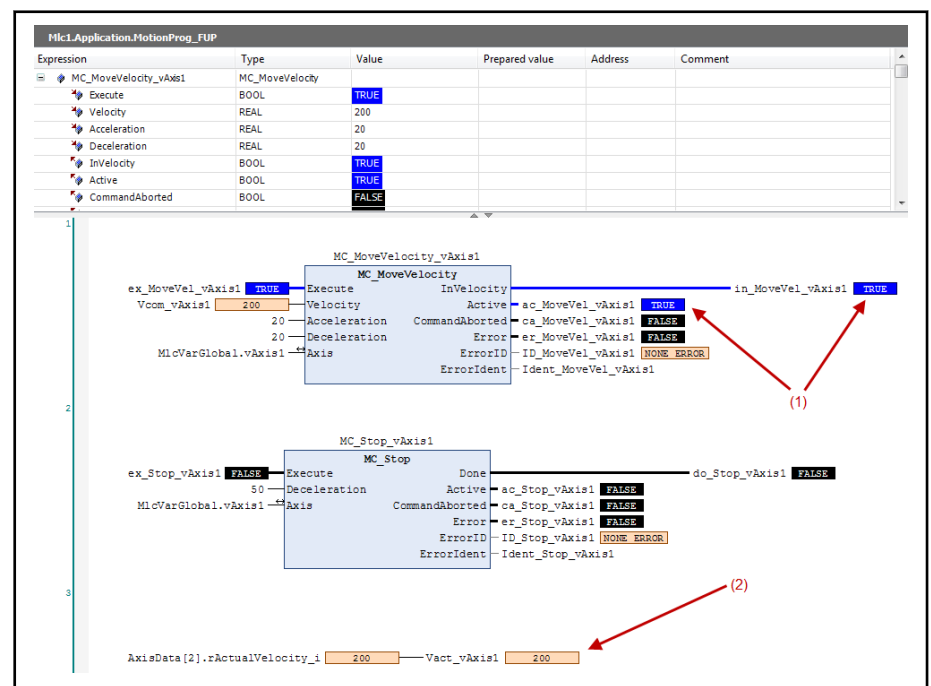


Fig. 3-40: After "Write values" at the "Execute" input

First steps - Motion

Switch "ex_MoveVel_vAxis1" again to "FALSE". The axis continues traveling at 200 rpm.

Set the variable "ex_Stop_vAxis1" at the "Execute" input of the "MC_Stop_vAxis1" function block of type "MC_Stop" to stop the axis motion.

Stop the application and log out.

3.4.4 Recording the motion of the axis "vAxis1" (trace), FBD

IndraLogic provides a trace tool to trace motion processes. Use this tool to record the following:

- the axis velocity "Vact_vAxis1"
- for "MC_MoveVelocity_vAxis1" (MC_MoveVelocity)
 - "ex_MoveVel_vAxis1" (Execute)
 - "in_MoveVel_vAxis1" (InVelocity)
 - "ca_MoveVel_vAxis1" (CommandAborted)

and

- for "MC_Stop_vAxis1" (MC_Stop)
 - "ex_Stop_vAxis1" (Execute)

Trace recording has to be configured first.

Insert the object "Trace_vAxis1" via the context menu of "Application" **Add ► Trace....** Double-click to open the object.

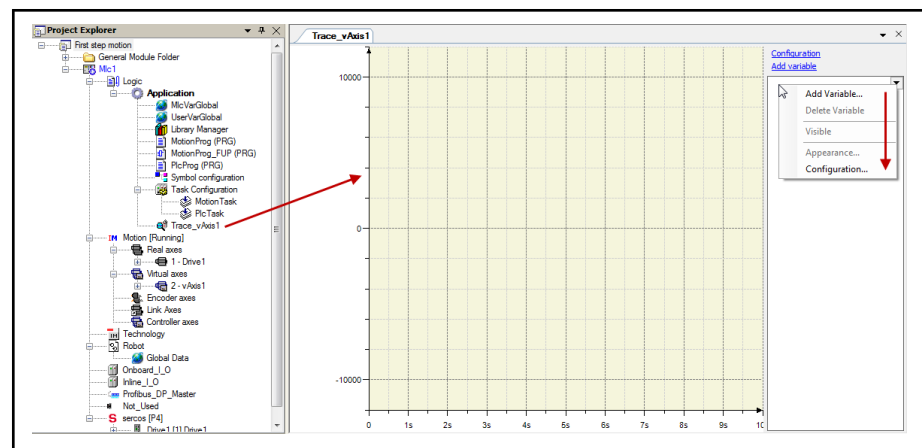


Fig. 3-41: Dialog: New trace object "Trace_vAxis1" with context menu

The following settings can be made for the "Trace" object via the context menu on the right:

- Configuration
- Display settings
- Add new variable

Configuration... Configure the new trace object first:

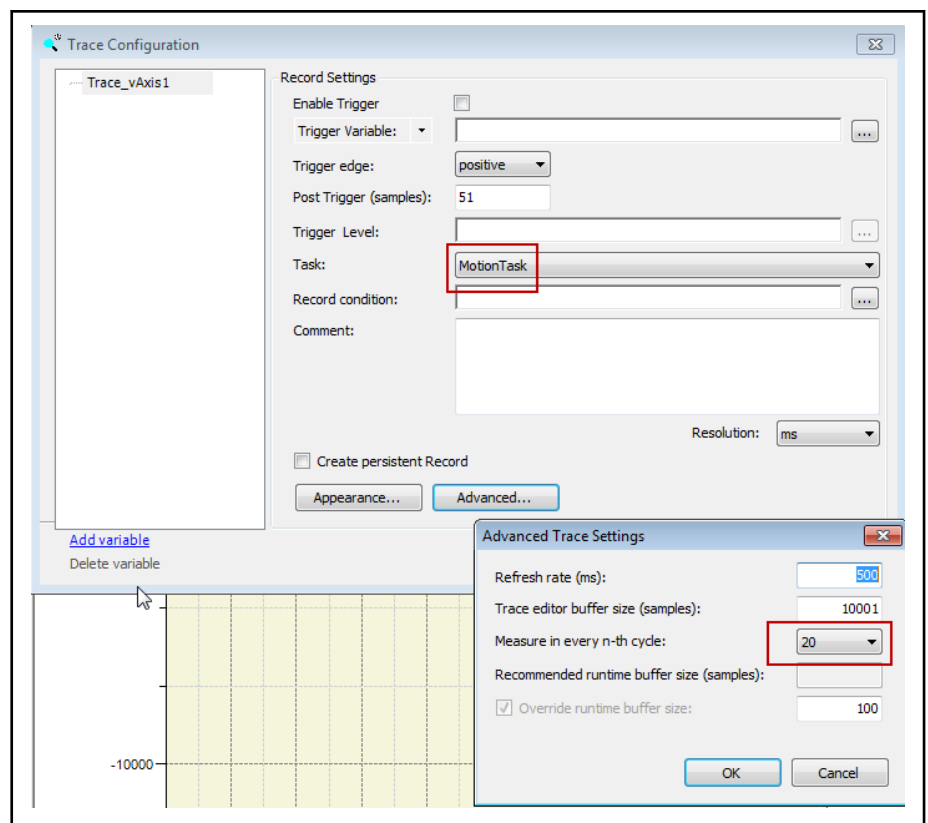


Fig. 3-42: Dialog: Trace configuration with extended "Trace" setting

The required settings are

- Task: **MotionTask**
- Measurement in every **20** th cycle, i.e. $20 \times 2 \text{ ms} = 40 \text{ ms}$

Task: Select the task - for which the values of trace variables are to be read after processing - from the list of available tasks. Default: First task in the task configuration tree.

Recording condition: A Boolean variable, a value or a Boolean expression can be entered here. When this condition is met (value is "TRUE"), value recording is started. If nothing is entered, value recording starts when the trace configuration is loaded to the control and the application is running.

Measure in every: Enter the number of cycles after which the trace variable value is to be read again. Use the formula (Buffer size $\times$ Record in every x-th cycle $\times$ Task interval) to estimate the minimum recording time span.

Recommended runtime buffer size: Size of the ring buffer in bytes, which is currently provided in the runtime system to collect trace variable values in the currently configured record. The buffer content is then displayed as value curve(s) in the trace window. Default: 100.



If the configuration changes, the Trace object has to be logged in again.

Display settings

Settings for the display can now be made. These settings are also applied without logging in again.

First steps - Motion

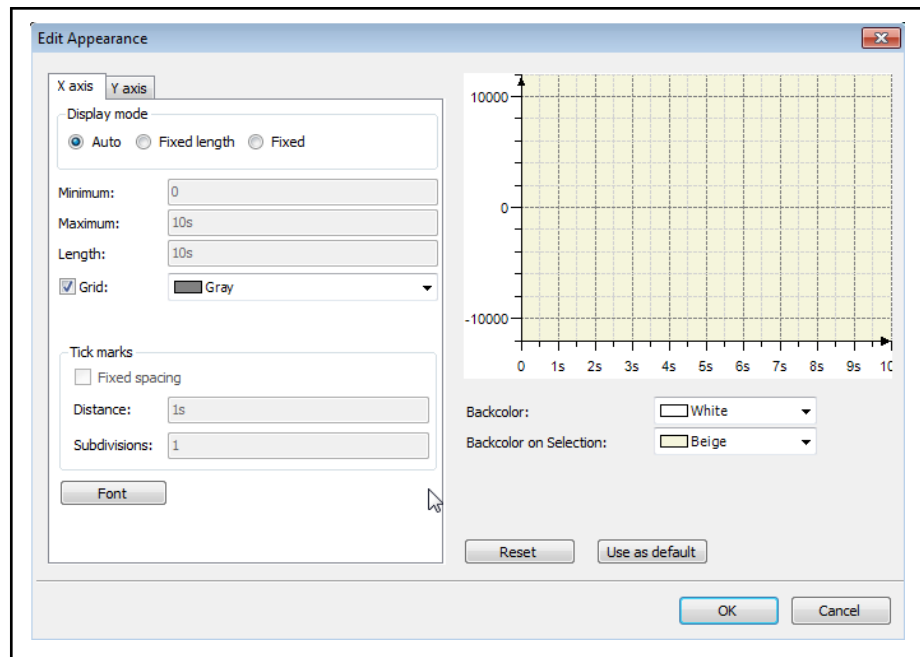


Fig. 3-43: Dialog: Display settings

Select **Auto**. A window with a recording **length** of 10 s appears over the curve. The **scale** is divided in a 1 s grid with a **scale division** of 1 s.

Selecting new variables

Now, name the variables required and set their appearance.

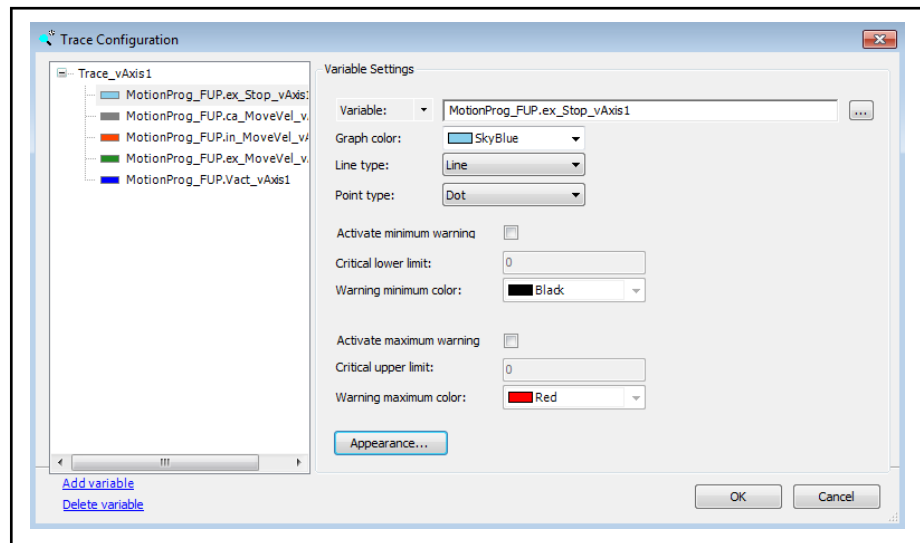


Fig. 3-44: Dialog: Creating new variables and specifying the appearance

Graph color and **Graph type** relate to the **Variables** curve.

In addition, if the respective limit is enabled, the parts of the curve that fall below a minimum value or exceed a maximum value can be offset in color against the remaining curve.

Menu to control the trace recording:

To display the context menu, right-click into the left section of the "Trace" window.

First steps - Motion

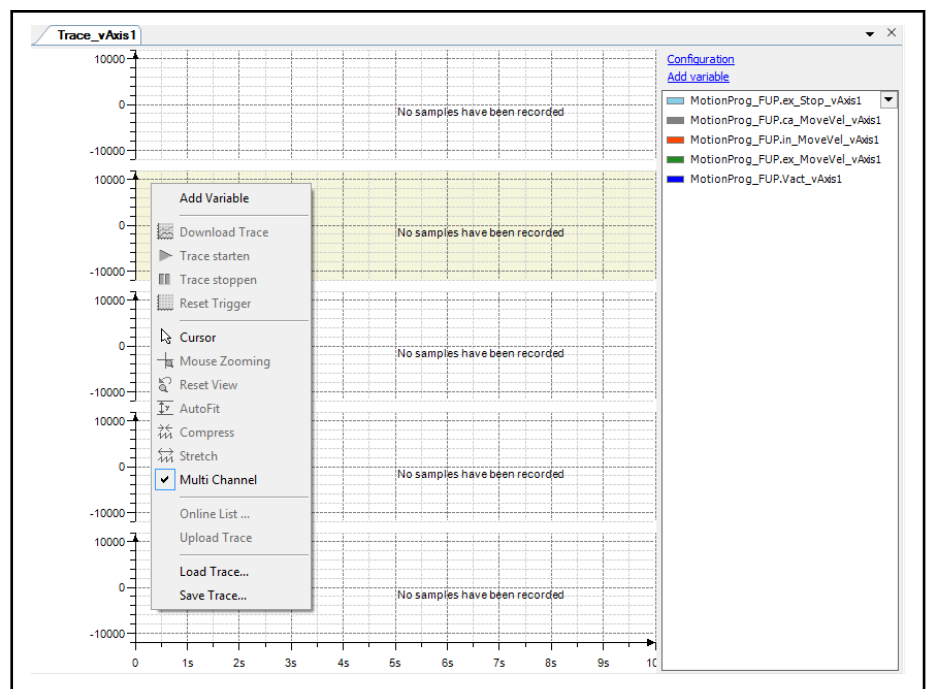


Fig. 3-45: Context menu to control the trace recording

Trace download: Before a configured trace recording on the control can be enabled for the application running there, the "Trace application" has to be loaded explicitly to the control.

Start/stop trace: This command starts or stops the display of the trace recording.

If the recording is stopped, the ► icon is positioned in front to the command.

If the recording is running, the ■ icon is positioned in front of the command and executing the command means that the recording is stopped, i.e. in the trace window, the section of the variable value curve(s) currently displayed is "frozen".

Multi-channel: This command allows to change the view of the trace curve. By default, the curves of all variables are shown in the same chart. In "Multi channel" view, the curve of each variable is visualized in its own diagram, but with an identical x-axis. Zooming and scrolling simultaneously affect the x-axis of all diagrams.

Cursor: When this command is enabled (default), the current position of the cursor in the coordinate system of the trace window is always displayed in the status bar of the programming system.

Scroll: In this mode, the mouse cursor is briefly displayed as a double arrow and by moving the cursor in the "Trace" window, the time axis of the recording currently displayed can be moved to the left or right.

Reset view: Use this command to reset the display of the recording(s) to the default values after it was changed by zooming for example.

- Performing trace:**
1. Log into the control.
Connect to the control.
 2. Start the control.
Control in BB/RUN.

First steps - Motion

3. Trace download.

When the application is running, the data required is transmitted to the control via the "Trace download" context menu.

The trace recording then starts automatically.

4. Write values.

The following values are set as in ["Write values of input variables" on page 100](#):

- "ex_MoveVel_vAxis1" to "TRUE".
- "ex_Stop_vAxis1" to "TRUE".

5. End the recording.

Stop the trace via the "Trace" context menu item "Start/stop trace".

6. Evaluate the trace recording.

Set window data.

Shortcuts	Action in the trace diagram	Mouse control
<Alt>+<←/→> arrow keys	Horizontal scrolling of x-axis (time axis) in all diagrams	Enable scrolling with mouse, click into the diagram and scroll the x-axis with the mouse keeping the left mouse button pressed
<↑/↓> arrow keys with or without pressed <Alt> key Select a diagram with <tab>	Horizontal scrolling of the selected diagram	Enable scrolling with mouse, click into the diagram and scroll the x-axis with the mouse keeping the left mouse button and <Ctrl> pressed
<↑/↓> arrow keys, with or without pressed <Alt> key Select a diagram with <tab> Keep <Ctrl> pressed	The graph is moved in bigger steps	
<+/-> keys	Horizontal scrolling of all diagrams along the x-axis (time axis is expanded/compressed for all diagrams)	• Select "Traces" or "Compress": •  or  • or use the mouse wheel
<Ctrl>+<+/-> keys Select a diagram with <tab>	Vertical scrolling of all diagrams along the y-axis (y-axis is expanded/compressed for the selected graph)	

Tab. 3-1: Trace Editor shortcut with activated multi-channel

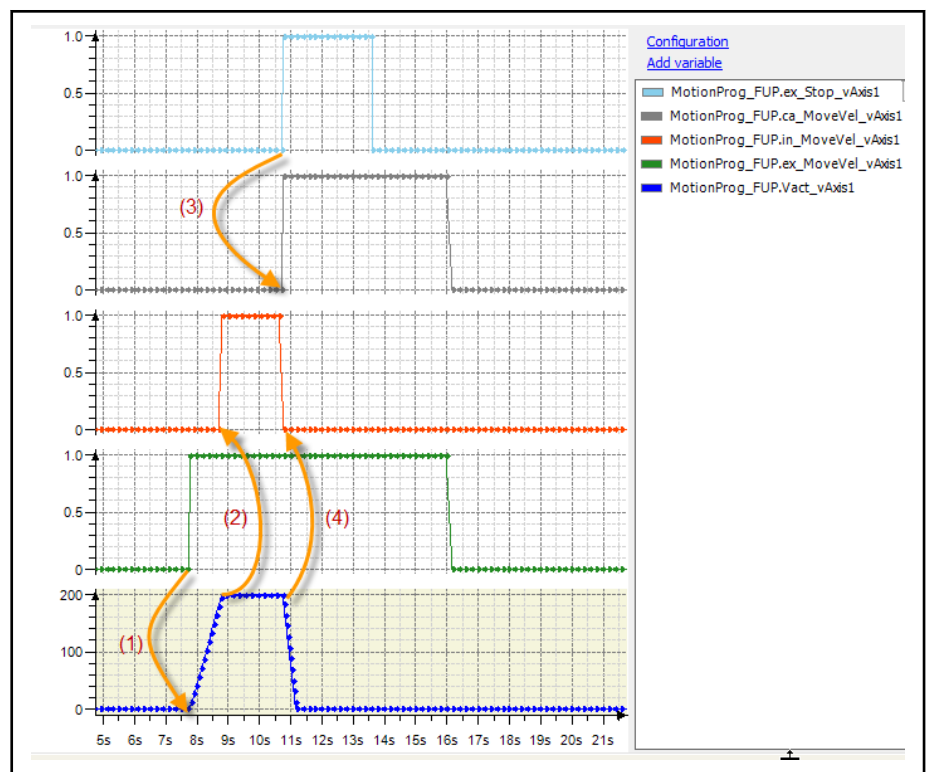


Fig. 3-46: Trace recording for the virtual axis "vAxis1"

Evaluate the trace recording

- (1) - Start of virtual axis startup
- (2) - Command speed reached
- (3) - Stop the virtual axis
- (4) - Virtual axis is decelerated

Note that the behavior of the virtual axis is as of a real axis with the same parameterization (compare the velocity characteristics, acceleration at 20 rad/s² and deceleration at 50 rad/s²).

3.4.5 Programming the real axis Drive1 (FBD)



From here onwards, the "real" drive technology is required
or
Select the second virtual axis **without** "MC_Power" as an alternative!

Before using a **real** axis, an instance of the PLCopen function block "MC_Power" is required.

Subsequently, one instance of "MC_GearIn" and "MC_GearOut" each are required.

The real axis is stopped with an "MC_Stop" instance.

The programming is executed as in section [chapter 3.5.2 "Programming the virtual axis vAxis1, programming language ST" on page 115.](#)

First steps - Motion

MC_Power

To add a new network, right-click on network 3

Insert network (below)

Insert "MC_Power" and declare an "MC_Power_Drive1" instance

Assign inputs and declare input variables if necessary:

"Enable" → "en_MC_Power_Drive1" of type "BOOL"

"Axis" → "MlcVarGlobal.Drive1" of type "AXIS_REF"

Outputs:

"Status" → "st_Power_Drive1" of type "BOOL"

"Error" → "er_Power_Drive1" of type "BOOL"

"ErrorID" → "ID_Power_Drive1" of type "ERROR_CODE"

"ErrorIdent" → "Ident_Power_Drive1" of type "ERROR_STRUCT"

MC_GearIn

To add a new network, right-click on network 4

Insert network (below)

Insert "MC_GearIn" and declaring the "MC_GearIn_Drive1" instance

Assign inputs and declare input variables if necessary:

"Execute" → "ex_GearIn_Drive1" of type "BOOL"

"RatioNumerator" → 5

"RatioDenominator" → 10

"MasterFineadjust" → 20 (%)

"Master" → "MlcVarGlobal.vAxis1" of type "AXIS_REF"

"Slave" → "MlcVarGlobal.Drive1" of type "AXIS_REF"

Outputs:

"InGear" → "ig_GearIn_Drive1" of type "BOOL"

"Active" → "ac_GearIn_Drive1" of type "BOOL"

"CommandAborted" → "ca_GearIn_Drive1" of type "BOOL"

"Error" → "er_GearIn_Drive1" of type "BOOL"

"ErrorID" → "ID_GearIn_Drive1" of type "ERROR_CODE"

"ErrorIdent" → "Ident_GearIn_Drive1" of type "ERROR_STRUCT"

MC_GearOut

To add a new network, right-click on network 5

Insert network (below)

Insert "MC_GearOut" and declare an "MC_GearOut_Drive1" instance

Assign inputs and declare input variables if necessary:

"Execute" → "ex_GearOut_Drive1" of type "BOOL"

"Slave" → "MlcVarGlobal.Drive1" of type "AXIS_REF"

Outputs:

"Done" → "do_GearOut_Drive1" of type "BOOL"

"Active" → "ac_GearOut_Drive1" of type "BOOL"

"Error" → "er_GearOut_Drive1" of type "BOOL"

"ErrorID" → "ID_GearOut_Drive1" of type "ERROR_CODE"

"ErrorIdent" → "Ident_GearOut_Drive1" of type "ERROR_STRUCT"

MC_Stop

To add a new network, right-click on network 6

Insert network (below)

Insert "MC_Stop" and declare an "MC_Stop_Drive1" instance

Assign inputs and declare input variables if necessary:

"Execute" → "ex_Stop_Drive1" of type "BOOL"

"Deceleration" → 50 (rad/s²)

"Axis" → "MlcVarGlobal.Drive1" of type "AXIS_REF"

Outputs:

"Done" → "do_Stop_Drive1" of type "BOOL"

"Active" → "ac_Stop_Drive1" of type "BOOL"

"CommandAborted" → "ca_Stop_Drive1" of type "BOOL"

"Error" → "er_Stop_Drive1" of type "BOOL"

"ErrorID" → "ID_Stop_Drive1" of type "ERROR_CODE"

"ErrorIdent" → "Ident_Stop_Drive1" of type "ERROR_STRUCT"

Actual velocity "Vact_Drive1"

The final step is the implementation of a network in which the current velocity of the axis ""Drive1"" "Vact_Drive1" (data type "REAL") can be read.

The current velocity is read from the data set of the axis (shared memory).

Data of all axes that can control an IndraMotion MLC, is included in the array

GVL_Base.AxisData: ARRAY [1..xx²] OF ML_AXISDATA_SM

The axis number has to be used as index.

The variable is provided with the name

GVL_Base.AxisData[1].rActualVelocity_i

for the real axis, axis number 1, ("Drive1" is provided with axis number 1, as "Drive1" was scanned before "vAxis1" was declared).

- To insert a new network, right-click on network 7

Insert network (below)

- Insert assignment

- Insert "Vact_Drive1" on the right ??? and press <Enter>, select the type "REAL" in the declaration window and press <Enter>.

- Go to the left ??? and click on  using the input assistant to navigate to **Variables ► ML_BASE ► Global Variable List ► GVL_Base ► AxisData:**

2) The number of controllable axes depends on the control type and its firmware, currently for the L65: 64 axes

First steps - Motion

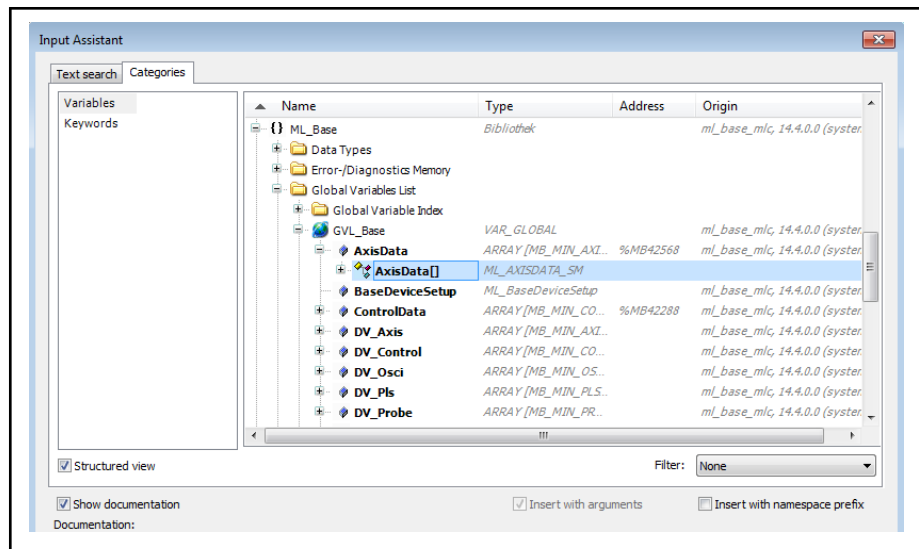


Fig. 3-47: Actual velocity value "vAxis1" from the Global Variable List of the "ML_Base.library"

Further entries are supported by the Intellisense function (automatic completion), i.e. after entering "." at "AxisData[1].", all available elements are shown and can be selected.

Confirm "rActualVelocity_i" with <Enter>.

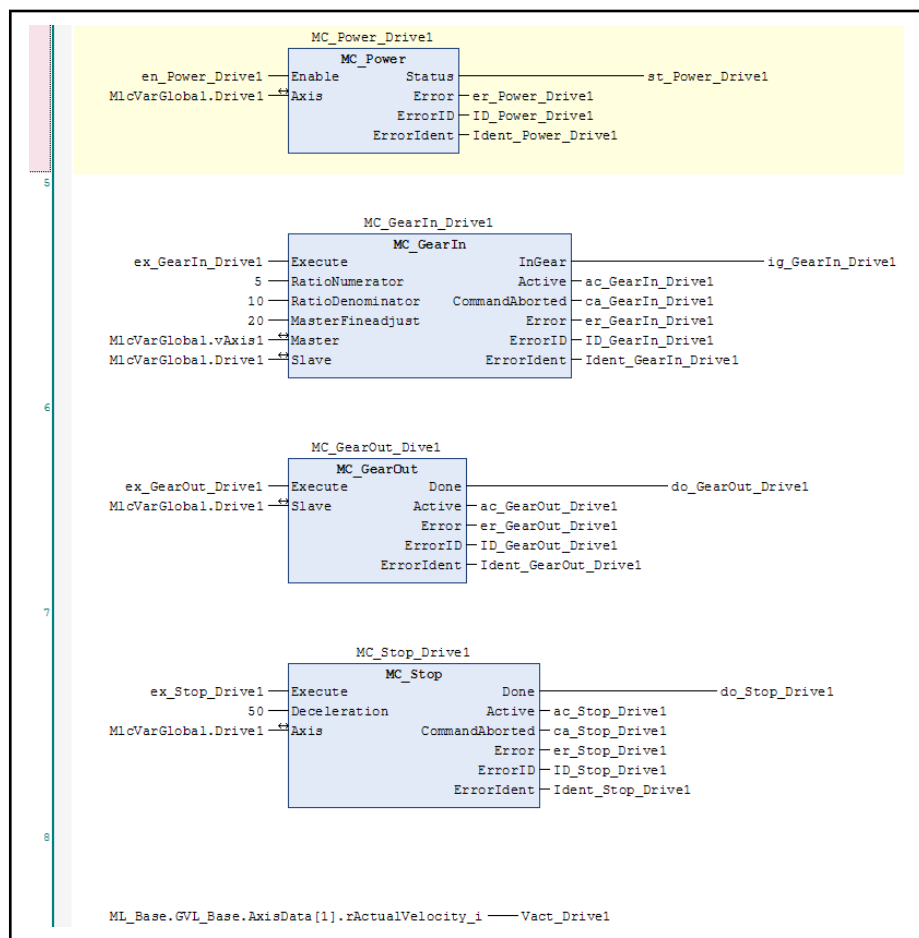


Fig. 3-48: Implementing the real axis



AxisData[].rActualVelocity_i; and
GVL_Base.AxisData[].rActualVelocity_i;
Are equally used by the compiler.

3.4.6 Downloading the complete PLC project (FBD)

The project is downloaded as described in section [chapter 3.4.3 "Downloading the PLC subproject "virtual axis" vAxis1 \(FBD\)" on page 99.](#)

The new project overwrites the old project after sending a prompt.

3.4.7 Workflow "Virtual axis → Electronic gear → Real axis "

The following procedure shows how to start and stop the axes and the gear.

1. Power on for the real axis.
Enable "en_Power_Drive1" (MC_Power) by "Write values".
st_Power_Drive1= TRUE (MC_Power).
2. Start the virtual axis.
Enable "ex_MoveVel_vAxis" (MC_MoveVelocity) by "Write values".
Startup ends at Vact_vAxis1 = 200 rpm.
3. Follow the real axis.
Enable "ex_GearIn_Drive1" (MC_GearIn) by "Write values".
The expected gear transmission ratio of the gear is:
 $Tr = (\text{RatioNumerator} / \text{RatioDenominator}) * (1 + \text{MasterFineadjust})$
for Vact_vAxis1 = 200 rpm results Vact_Drive1 = 120 rpm.



The MasterFineadjust value is specified in "%".

4. If the MC_GearIn is "InGear", the velocity "Vcom_vAxis1" of the virtual axis can be changed by "Write values" (double-click on the variable, enter the desired value). The real axis follows the virtual axis.
5. Disable "ex_GearIn_Drive1" (MC_GearIn) by "Write values".
in_GearIn_Drive1= FALSE, ac_GearIn_Drive1= TRUE.
The real axis continues to follow the virtual axis.
6. Terminate synchronicity.
Enable the "ex_GearOut_Drive1" (MC_GearOut) by "Write values".
The dependency of the real axis on the virtual axis is suspended. However, both axes continue to rotate with the same velocity.
7. Change the command velocity "Vcom_vAxis1".
The changed command velocity only affects the virtual axis. "Vact_Drive1" remains unchanged.
8. Disable "ex_GearOut_Drive1" by "Write values".
do_GearOut_Drive1= FALSE.
9. Stop the virtual axis.
Enable "ex_Stop_Drive1" (MC_Stop).

First steps - Motion

The virtual axis is decelerated to $Vact_vAxis1 = 0$ rpm.

10. Disable "ex_Stop_Drive1" (MC_Stop).

11. Stop the real axis.

Enable "ex_Stop_Drive1" (MC_Stop).

The real axis is decelerated to $Vact_Drive1 = 0$ rpm. After it stopped, it is actively held in this position.

12. Disable "ex_Stop_Drive1" (MC_Stop).

The active holding of the real axis finished.

13. Power off for the real axis.

Disable "en_Power_Drive1" (MC_Power) by "Write values".



Only possible in the "standstill" state after executing the stop at "Drive1".

3.4.8 Recording the motion of both axes (trace), FBD

The "Trace" tool was used as in section [chapter 3.4.4 "Recording the motion of the axis vAxis1 \(trace\), FBD" on page 102](#) to visualize the startup of the virtual axis and the synchronization.

The following workflow shows how to start and stop the axes and the gear for the trace recording.

1. Power on for the real axis.

Enable "en_Power_Drive1" (MC_Power).

2. Start the virtual axis.

Enable "ex_MoveVel_vAxis1" (MC_MoveVelocity).

3. Follow the real axis.

Enable "ex_GearIn_Drive1" (MC_GearIn).

4. Speed command value jump of the virtual axis.

Increase the speed command value to "250".

Is applied by "ex_MoveVel_vAxis1" TRUE $\Rightarrow$ FALSE $\Rightarrow$ TRUE.

The following figure shows the trace result:

First steps - Motion

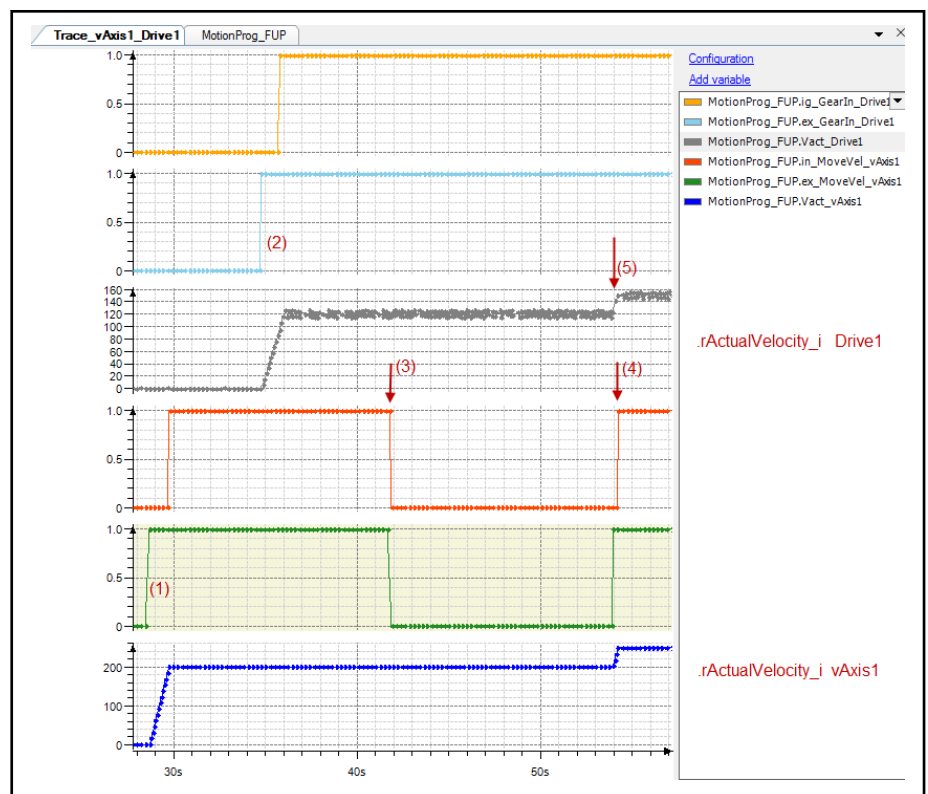


Fig. 3-49: Trace recording for both axes

Evaluate the trace recording

- (1) - Start of virtual axis startup
- (2) - Enable gear function, real axis startup
- (3) - Increase command speed of the virtual axis, "ex_MoveVel_vAxis1" TRUE ⇒ FALSE
- (4) - Command speed of the virtual axis is applied, "ex_MoveVel_vAxis1" FALSE ⇒ TRUE
- (5) - Jump response at the real axis

3.5 Programming the motion sequence in Structured Text (ST)

3.5.1 General information

The motion sequence is programmed in a PLC program using the PLCopen function blocks (programming system IndraLogic 2G acc. to EN 61131-3).

When creating a project using the "Mlc1" control, the real axis "Drive1" and the virtual axis "vAxis1", the first actions to program the motion sequence were started below the "Logic" folder:

First steps - Motion

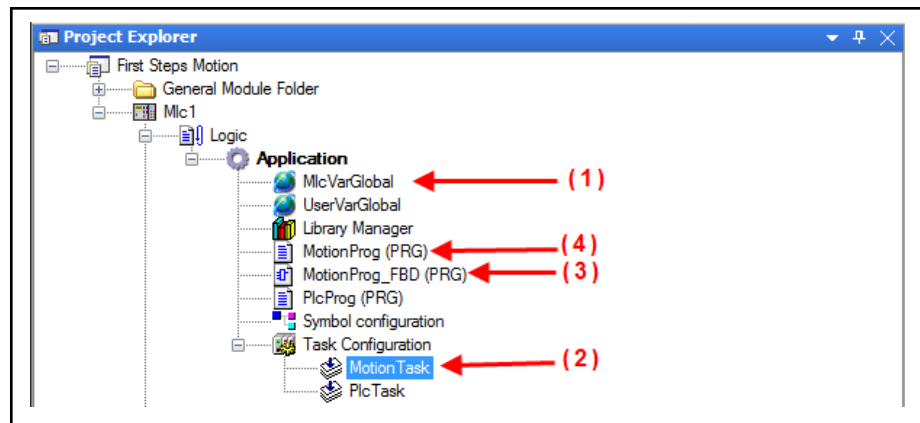


Fig. 3-50: Project Explorer after expanding the "Logic" node

The following programs were created:

- **MotionProg (3)** (preselection of ST, Structured Text)
- **PlcProg** (preselection of ST, Structured Text), not used

The third program, **MotionProg_FBD (4)**, was manually supplemented: **Application**, right-click, **Add ► POU**, enter the name and select FBD as programming language.

The programs are still empty.

The following tasks were created under "Task configuration":

- **MotionTask (2)**, (priority 2, externally event-triggered by the Sercos task, watchdog active) for "MotionProg".
The program(s) running under this task, has (have) to be processed completely within the specified period of the Sercos cycle. If they are not within this part, the control is stopped
- **PlcTask**, (low-priority cyclic task) for PlcProg (not used).
In the remaining time, the program is processed at least piece by piece, that means during several cycles, if required

Global variables/global constants were created:

- **UserVarGlobal**, reserved for the user, not used in "First steps - Motion"
- **MlcVarGlobal (1)** includes the address constants for the axes Drive1 and vAxis1

Program:

```
(* Automated generated code by MLC. *)
(* Please don't edit, the code will be overwritten...*)
VAR_GLOBAL CONSTANT
(*virtual axis "vAxis1" with axis number 2*)
vAxis1: AXIS_REF := (CntlNo:=LOCAL_CNTRL,AxisNo:=AXIS_2);
(*real axis "Drive1" with axis number 1*)
Drive1: AXIS_REF := (CntlNo:=LOCAL_CNTRL,AxisNo:=AXIS_1);
(*...*)
END_VAR
```



Only the MotionTask is used for the "First steps Motion" project.

According to the programming language required, either **MotionProg** or **MotionProg_FBD** is assigned.

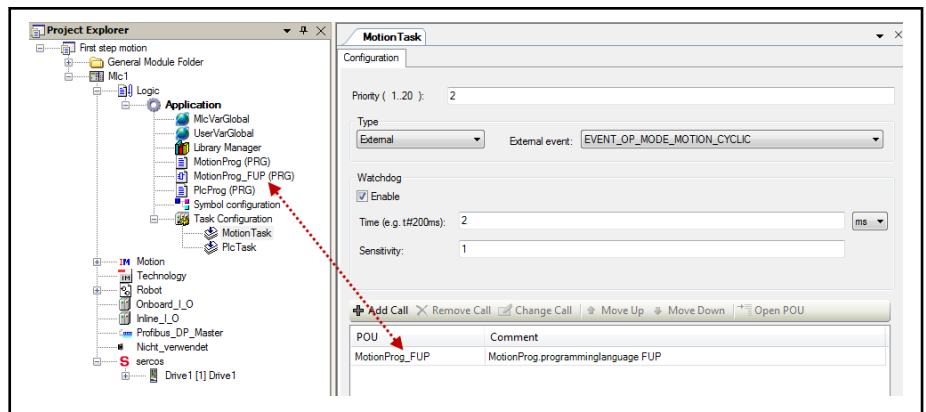


Fig. 3-51: Deleting call: MotionProg_FBD; add call: MotionProg

3.5.2 Programming the virtual axis "vAxis1", programming language ST



This section uses ST (Structured Text) as programming language. From the functional point of view, the same result as in the sections with the programming language FBD.

Similarly to [chapter 3.1 "Project goal First steps - Motion" on page 77](#), instances of the PLCopen blocks "MC_MoveVelocity" and "MC_Stop" are called using "vAxis1".



The "MC_Power" instance is not required for virtual axes.

Proceed as follows:

1. Assign the MotionTask to the "MotionProg" program.
2. Left double-click to open the "MotionProg" program.

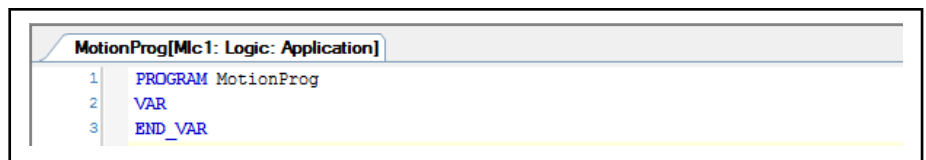


Fig. 3-52: New program in ST (Structured Text)

Unlike when programming in FBD, the instances of the function blocks in the declaration section, range VAR ... END_VAR have to be declared first for the programming language ST (Structured Text).

Only then, they can be used in the implementation.

First enter the instance name of the MC_MoveVelocity function block "MC_MoveVel_vAxis1:" into the range VAR ... END_VAR. Right-click to open the context menu and select the input assistant (alternatively <F2> window).

First steps - Motion

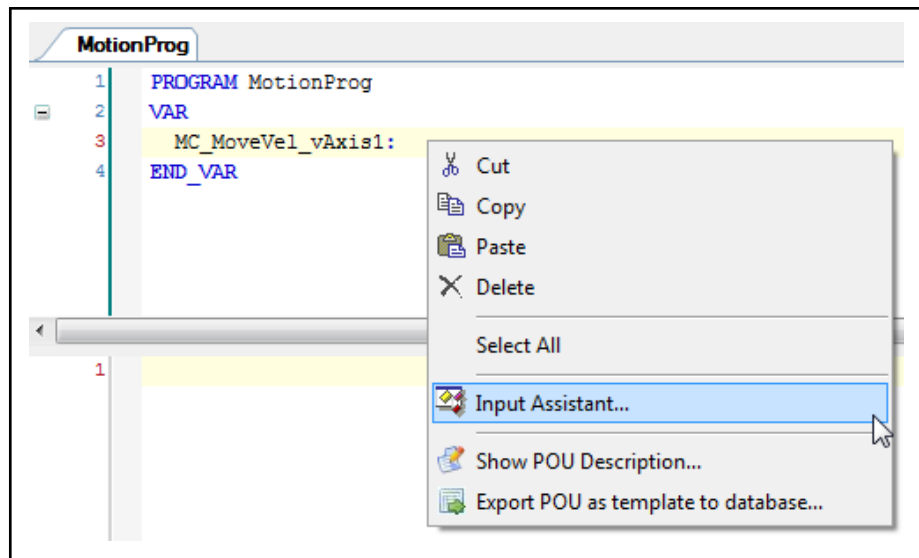


Fig. 3-53: Implementing the instance "MoveVel_vAxis1" of type "MC_MoveVelocity"

Therefore, select the following in the open selection window:

User-defined types

► ML_PLCopen ► POU ► Motion ► MC_MoveVelocity.

Deselect the option "Insert with arguments".

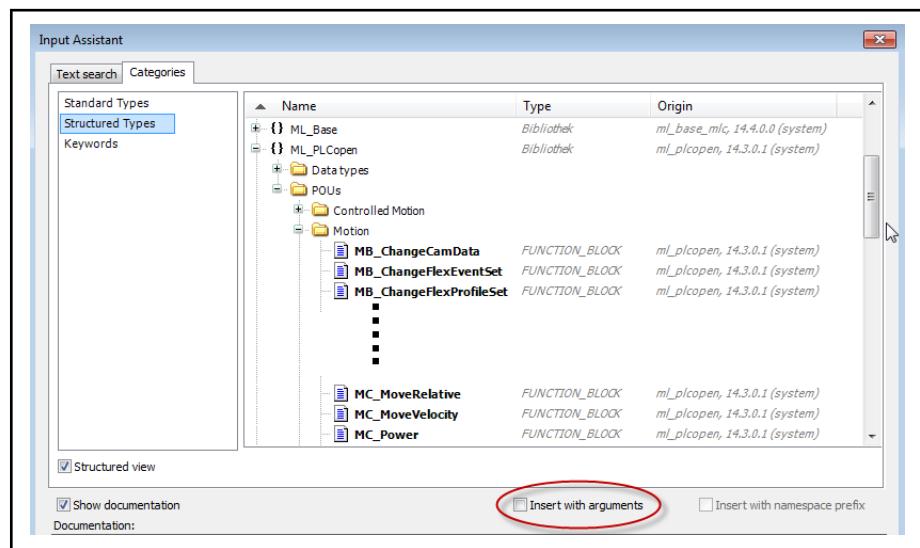


Fig. 3-54: Selection window "Function blocks"

Press **OK** and complete the line with a ";".

Go to line 1 in the implementation section of the editor and press <F2> to open the input assistant.

Select **Instance calls** ► **MoveVel_vAxis1** in the open selection window.

Select the option "Insert with arguments".

First steps - Motion

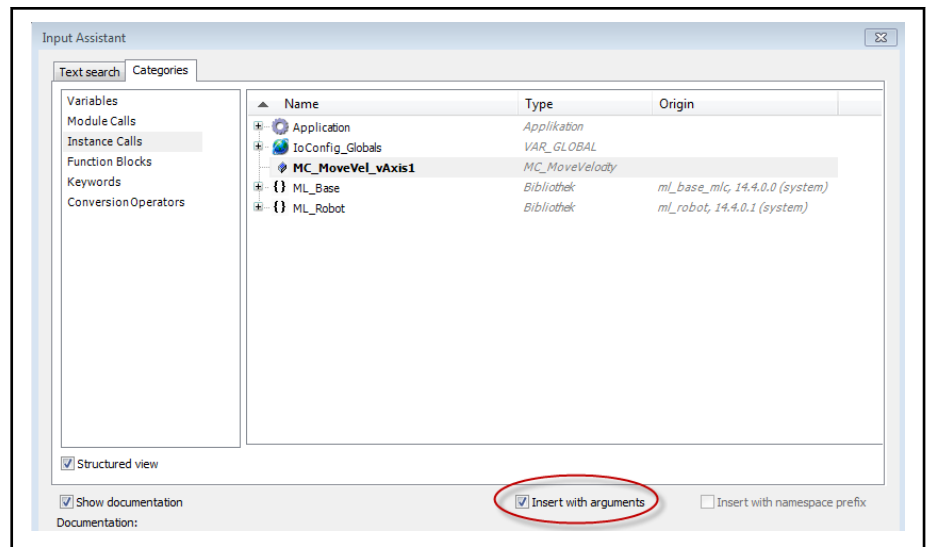


Fig. 3-55: Applying the instance into the implementation

Press **OK** to confirm the insertion of the instance.

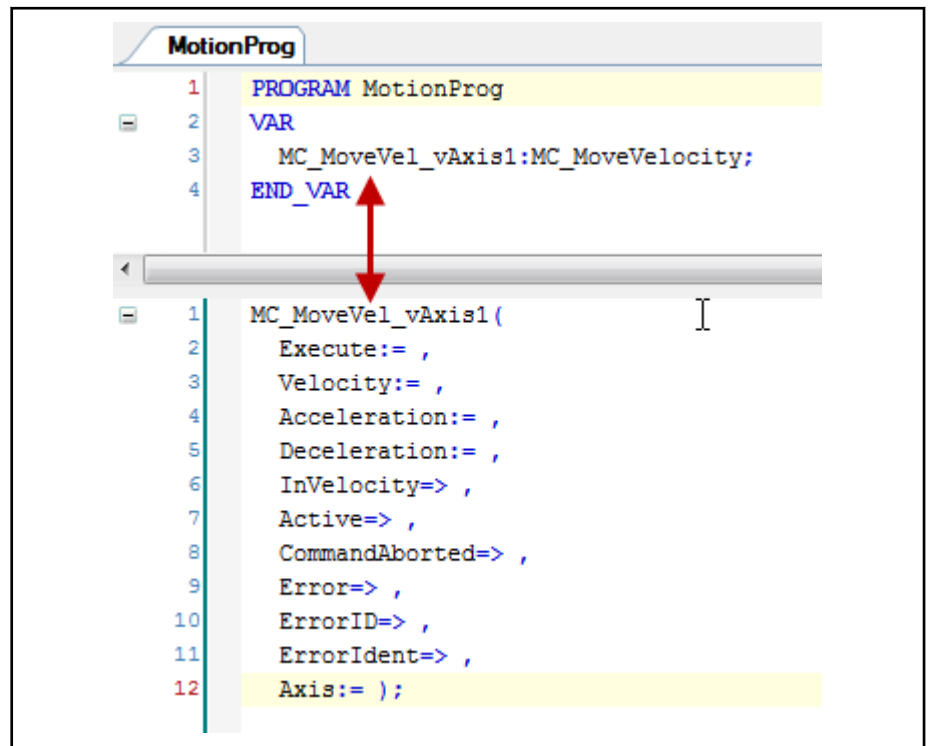


Fig. 3-56: Body of a function block instance

Assign the function block inputs.

Start with the "Execute" input:

- Enter the name "ex_MoveVel_vAxis1" and click on the next line.
- Confirm the open "Auto declare" window with <Enter>

To assign the "Velocity" input, proceed as follows:

- Enter the name "Vcom_vAxis1" and click on the next line.
- Enter REAL as type and 200 (rpm) as initial value and press <Enter>
- Enter 20 (rad/s²) for the inputs "Acceleration" and "Deceleration".

First steps - Motion



Enter the values directly at the function block inputs.

Alternatively, a variable with the respective value can be used.

For feedback whether the function block operated correctly, its outputs are assigned.

- Assign the "InVelocity" output of the function block as usual using the "iv_MoveVel_vAxis1" variable, type "BOOL"
- Enter the name "iv_MoveVel_vAxis1" and click on the next line.
- Confirm the open "Auto declare" window with <Enter>
- Assign the "Active" output with "ac_MoveVel_vAxis1"
- Assign "ca_MoveVel_vAxis1" to the "CommandAborted" output
- Assign the "Error" output with "er_MoveVel_vAxis1"
- Assign the "ErrorID" output with "ID_MoveVel_vAxis1"
- Assign the "ErrorIdent" output with "Ident_MoveVel_vAxis1"

To assign the "Axis" input, proceed as follows:

- Right-click to open the input assistant:
Select **Variables** ► **Application** ► **MlcVarGlobal** ► **vAxis1**.

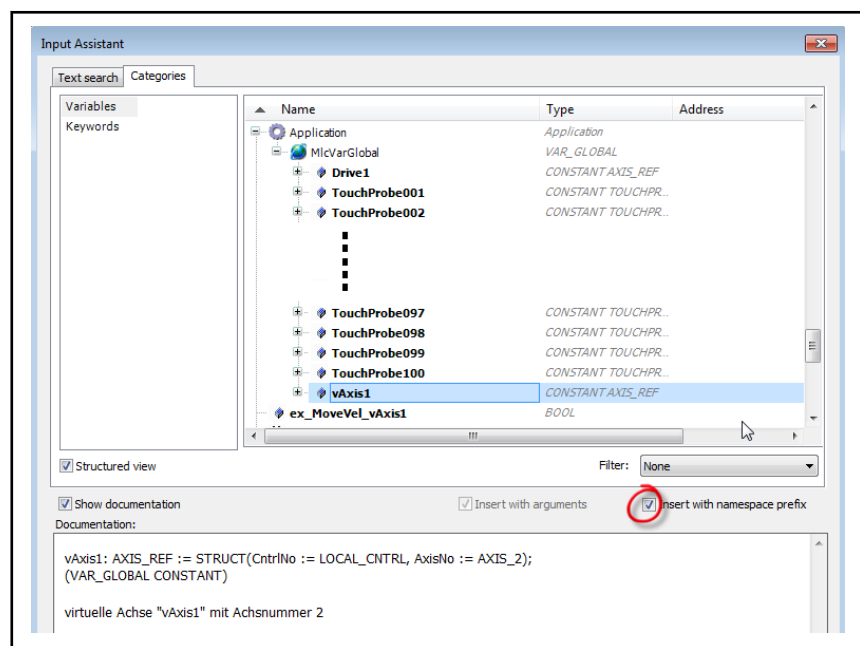


Fig. 3-57: *Selecting the axis instance when assigning a function block of the type "MC_MoveVelocity"*

- Confirm with **OK**.



The name of a global constant consists of the folder name and the element name:

`MlcVarGlobal.vAxis1`

The PLCopen function block of type "MC_MoveVelocity" is thus assigned according to its task. The declaration section was automatically completed.

```

MotionProg
2  VAR
3      MC_MoveVel_vAxis1:MC_MoveVelocity;
4      ex_MoveVel_vAxis1: BOOL;
5      Vcom_vAxis1: REAL := 200;
6      iv_MoveVel_vAxis1: BOOL;
7      ac_MoveVel_vAxis1: BOOL;
8      ca_MoveVel_vAxis1: BOOL;
9      er_MoveVel_vAxis1: BOOL;
10     ID_MoveVel_vAxis1: ERROR_CODE;
11     Ident_MoveVel_vAxis1: ERROR_STRUCT;
12 END_VAR

1  MC_MoveVel_vAxis1(
2      Execute:= ex_MoveVel_vAxis1,
3      Velocity:= Vcom_vAxis1,
4      Acceleration:= 20,
5      Deceleration:= 20,
6      InVelocity=> iv_MoveVel_vAxis1,
7      Active=> ac_MoveVel_vAxis1,
8      CommandAborted=> ca_MoveVel_vAxis1,
9      Error=> er_MoveVel_vAxis1,
10     ErrorID=> ID_MoveVel_vAxis1,
11     ErrorIdent=> Ident_MoveVel_vAxis1 ,
12     Axis:= MlcVarGlobal.vAxis1 );
    
```

Fig. 3-58: PLCopen FB, "MC_MoveVelocity" type is inserted and assigned



Whenever a function block is completed, run a test compilation using <Shift>+<F11>. Clear any errors immediately.

Generate an instance of the PLCopen function block "MC_Stop".

MC_Stop

Declare and insert instance "MC_Stop_vAxis1" of "MC_Stop"

Assign inputs and declare input variables if necessary:

"Execute" → "ex_Stop_vAxis1" of type "BOOL"

"Deceleration" → 50

"Axis" → "MlcVarGlobal.vAxis1" of type "AXIS_REF"

Outputs:

"Done" → "do_Stop_vAxis1" of type "BOOL"

"Active" → "ac_Stop_vAxis1" of type "BOOL"

"CommandAborted" → "ca_Stop_vAxis1" of type "BOOL"

"Error" → "er_Stop_vAxis1" of type "BOOL"

"ErrorID" → "ID_Stop_vAxis1" of type "ERROR_CODE"

"ErrorIdent" → "Ident_Stop_vAxis1" of type "ERROR_STRUCT"

First steps - Motion

Actual velocity "Vact_vAxis1"

The final step is the implementation of a network in which the current velocity of the axis "vAxis1" "Vact_vAxis1" (data type REAL) can be read.

The current velocity is read from the data set of the axis (shared memory).

Data of all axes, which can control a control, is located in the array

GVL_Base.AxisData: ARRAY [1..xx³⁾] OF ML_AXISDATA_SM

The axis number has to be used as index.

The variable is provided with the name

GVL_Base.AxisData[2].rActualVelocity_i

for the virtual axis, axis number 2, ("Drive1" is provided with axis number 1, as "Drive1" was scanned before "vAxis1" was declared).

- Insert new line
- Insert `Vact_vAxis1:=` , complete declaration
- using the input assistant to navigate to **Variables ► ML_BASE ► Global Variable List ► GVL_Base ► AxisData:**

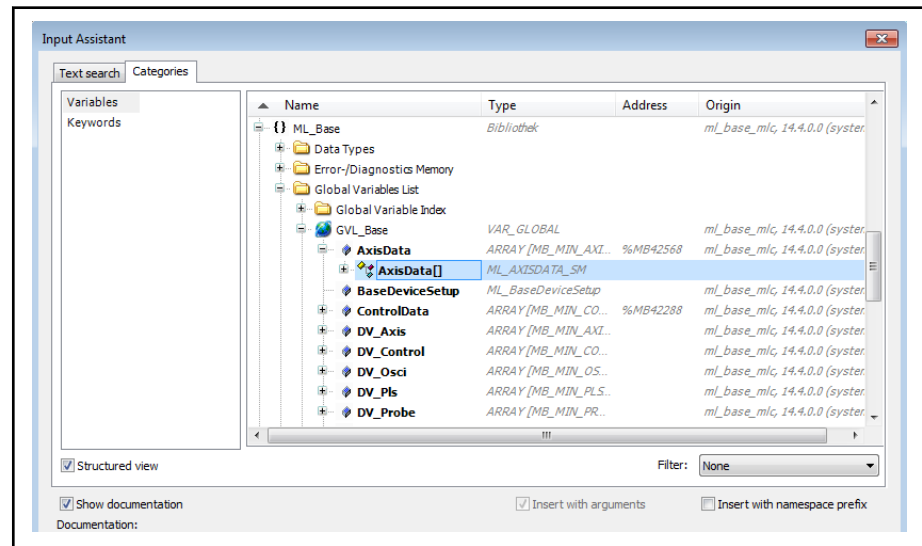


Fig. 3-59: Actual velocity value "vAxis1" from the Global Variable List of the ML_Base.library

Further entries are supported by the Intellisense function (automatic completion), i.e. after entering "." at "AxisData[2].", all available elements are shown and can be selected.

Confirm "rActualVelocity_i" with <Enter>.

3) The number of controllable axes depends on the control type and its firmware, currently for the L65: 64 axes

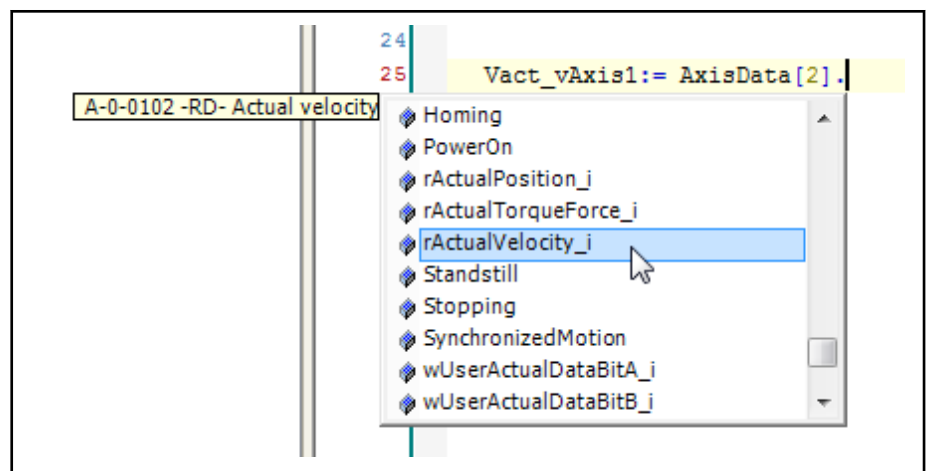
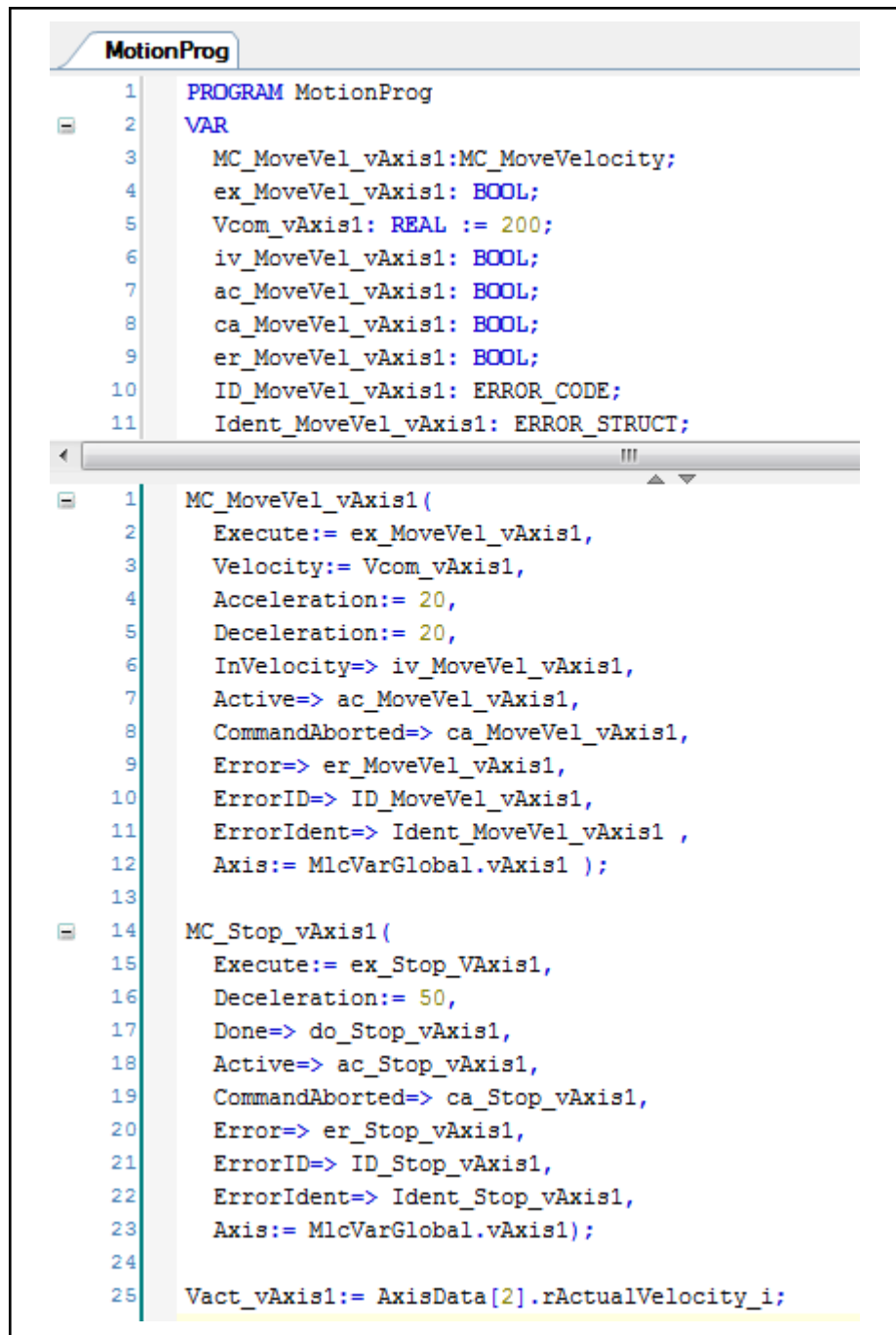


Fig. 3-60: Selecting the variables of the actual velocity

- Do not forget to complete the line with ";".

The implementation of the motion sequence according to the task assignment is thus completed:

First steps - Motion



```

1  PROGRAM MotionProg
2  VAR
3      MC_MoveVel_vAxis1:MC_MoveVelocity;
4      ex_MoveVel_vAxis1: BOOL;
5      Vcom_vAxis1: REAL := 200;
6      iv_MoveVel_vAxis1: BOOL;
7      ac_MoveVel_vAxis1: BOOL;
8      ca_MoveVel_vAxis1: BOOL;
9      er_MoveVel_vAxis1: BOOL;
10     ID_MoveVel_vAxis1: ERROR_CODE;
11     Ident_MoveVel_vAxis1: ERROR_STRUCT;
12
13
14 MC_MoveVel_vAxis1(
15     Execute:= ex_MoveVel_vAxis1,
16     Velocity:= Vcom_vAxis1,
17     Acceleration:= 20,
18     Deceleration:= 20,
19     InVelocity=> iv_MoveVel_vAxis1,
20     Active=> ac_MoveVel_vAxis1,
21     CommandAborted=> ca_MoveVel_vAxis1,
22     Error=> er_MoveVel_vAxis1,
23     ErrorID=> ID_MoveVel_vAxis1,
24     ErrorIdent=> Ident_MoveVel_vAxis1 ,
25     Axis:= MlcVarGlobal.vAxis1 );
26
27 MC_Stop_vAxis1(
28     Execute:= ex_Stop_vAxis1,
29     Deceleration:= 50,
30     Done=> do_Stop_vAxis1,
31     Active=> ac_Stop_vAxis1,
32     CommandAborted=> ca_Stop_vAxis1,
33     Error=> er_Stop_vAxis1,
34     ErrorID=> ID_Stop_vAxis1,
35     ErrorIdent=> Ident_Stop_vAxis1,
36     Axis:= MlcVarGlobal.vAxis1);
37
38 Vact_vAxis1:= AxisData[2].rActualVelocity_i;

```

Fig. 3-61: Implementing the virtual axis "vAxis1"



AxisData[].rActualVelocity_i; and
 GVL_Base.AxisData[].rActualVelocity_i;
 Are equally used by the compiler.

3.5.3 Downloading PLC subproject "virtual axis" "vAxis1" (ST)

The previously implemented motion sequence still has to be compiled, applied to the control and started.

For the compilation, go to the IndraWorks menu bar

- **Build ► Build**
Compilation of the added part of the application
- **Build ► Generate code**
The command allows to generate a code and to check compilation errors

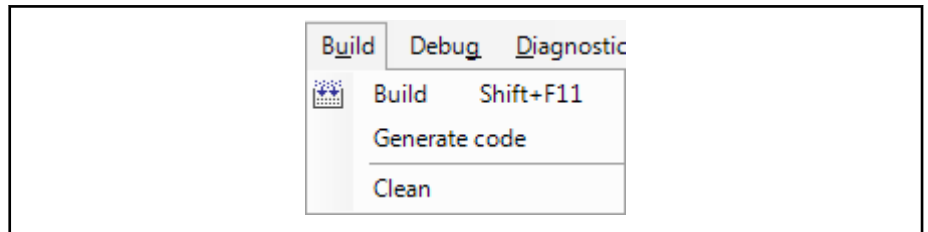


Fig. 3-62: Build application

To transfer the compiled data to the target control, select **Debug ► Login** from the IndraWorks menu bar (refer to the figure below).

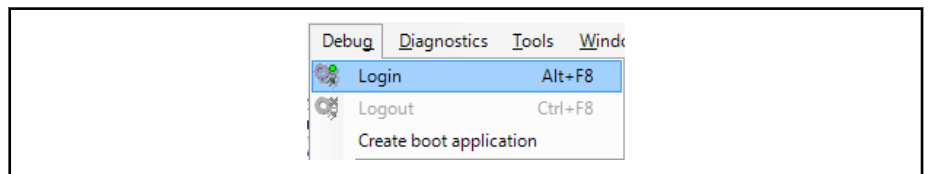


Fig. 3-63: Transferring compiled data to the control

A prompt asking for agreement to download is displayed depending on whether an application is already loaded and whether this application is a previous version of the current application.

The application is still stopped after a successful transmission (red in the Project Explorer and in the status bar) and the variables in declaration and implementation are prepared to display the values. However, the values are invalid.

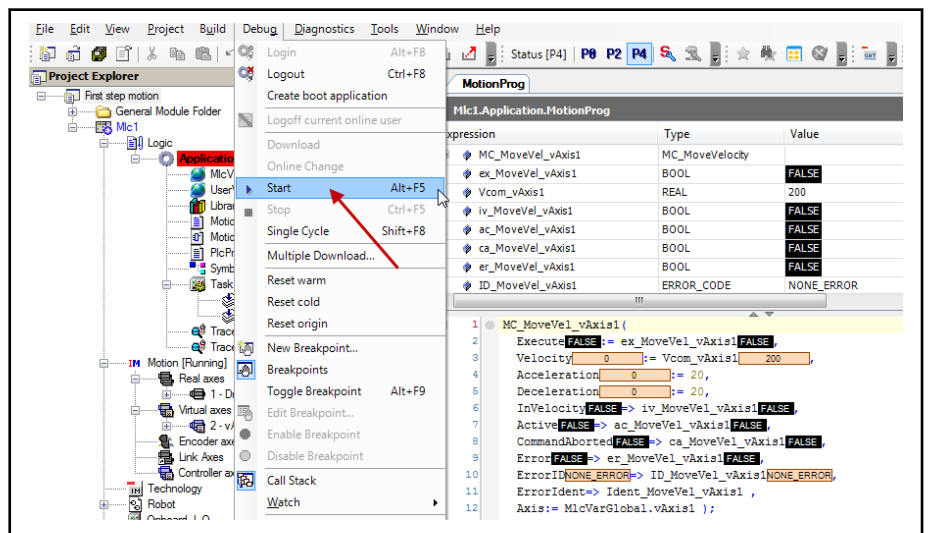


Fig. 3-64: Starting the program processing in the control (1)

Start the processing in the control via **Debug ► Start**.

First steps - Motion

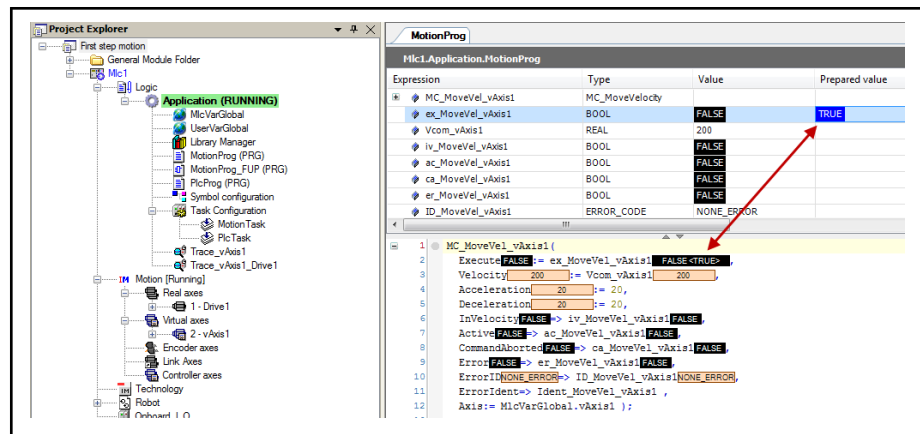


Fig. 3-65: Starting the program processing in the control (2)

The motion process starts automatically, since the processing of the programmed function blocks is subsequently enabled.

"Write values", input variables

To enable the processing of "MoveVel_vAxis1" of type "MC_MoveVelocity", there has to be a positive edge at the "Execute" input of the function block.

Left-click on "FALSE" behind the input variable "ex_MoveVel_vAxis1". The "Prepare value" window opens. Enter "TRUE" and confirm it.

Now, a selected "FALSE <TRUE>" can be seen behind "ex_MoveVel_vAxis1" and "TRUE" is displayed in the declaration line.

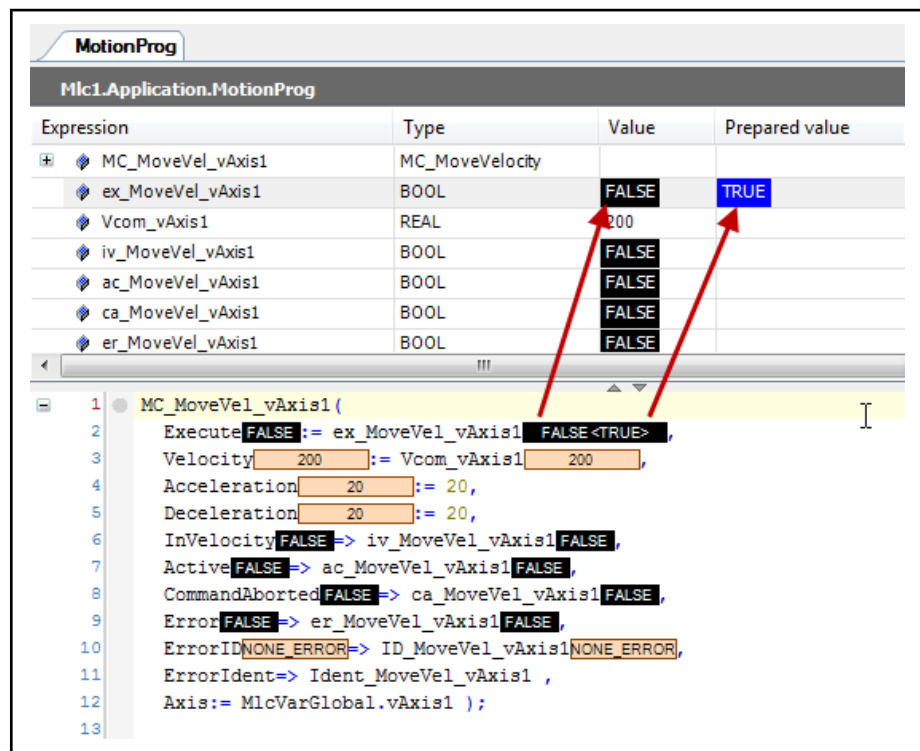


Fig. 3-66: Preparing "Write values"

Go to **Debug ► Write values** <Ctrl>+<F7> to set "ex_MoveVel_vAxis1" to "TRUE" in the control.



- "Write values" means that the variable is set to the desired value **once** at the beginning of the next cycle
- "Force values" means that the variable is **permanently** set to the desired value at the beginning of the following cycles

The resulting positive edge sets the outputs "InVelocity" and "Active" of the "MoveVel_vAxis1" function block to "TRUE" (1) (refer to the following figure).

The axis travels at 200 rpm (2).

Expression	Type	Value	Prepared value	Ad
MC_MoveVel_vAxis1	MC_MoveVelocity			
ex_MoveVel_vAxis1	BOOL	TRUE		
Vcom_vAxis1	REAL	200		
iv_MoveVel_vAxis1	BOOL	TRUE		
ac_MoveVel_vAxis1	BOOL	TRUE		
ca_MoveVel_vAxis1	BOOL	FALSE		
er_MoveVel_vAxis1	BOOL	FALSE		
ID_MoveVel_vAxis1	ERROR_CODE	NONE_ERR...		

```
1 MC_MoveVel_vAxis1(  
2   Execute TRUE := ex_MoveVel_vAxis1 TRUE ,  
3   Velocity 200 := Vcom_vAxis1 200 ,  
4   Acceleration 20 := 20 ,  
5   Deceleration 20 := 20 ,  
6   InVelocity TRUE => iv_MoveVel_vAxis1 TRUE , (1)  
7   Active TRUE => ac_MoveVel_vAxis1 TRUE , (1)  
8   CommandAborted FALSE => ca_MoveVel_vAxis1 FALSE ,  
9   Error FALSE => er_MoveVel_vAxis1 FALSE ,  
10  ErrorID NONE_ERROR => ID_MoveVel_vAxis1 NONE_ERROR ,  
11  ErrorIdent => Ident_MoveVel_vAxis1 ,  
12  Axis := MlcVarGlobal.vAxis1 );  
13  
14 MC_Stop_vAxis1(  
15   Execute FALSE := ex_Stop_vAxis1 FALSE ,  
16   Deceleration 50 := 50 ,  
17   Done FALSE => do_Stop_vAxis1 FALSE ,  
18   Active FALSE => ac_Stop_vAxis1 FALSE ,  
19   CommandAborted FALSE => ca_Stop_vAxis1 FALSE ,  
20   Error FALSE => er_Stop_vAxis1 FALSE ,  
21   ErrorID NONE_ERROR => ID_Stop_vAxis1 NONE_ERROR ,  
22   ErrorIdent => Ident_Stop_vAxis1 ,  
23   Axis := MlcVarGlobal.vAxis1 );  
24  
25 Vact_vAxis1 200 := AxisData[2].rActualVelocity_i 200 ; RETURN
```

Fig. 3-67: After "Write values" at the "Execute" input

Switch "ex_MoveVel_vAxis1" again to "FALSE". The axis continues traveling at 200 rpm.

To stop the axis motion, set the variable "ex_Stop_vAxis1" at the "Execute" input of the "Stop_vAxis1" function block of type "MC_Stop".

Stop the PLC and log out.

3.5.4 Recording the motion of the axis "vAxis1" (trace), ST

IndraLogic provides a trace tool to trace motion processes. Use this tool to record the following:

- the axis velocity "Vact_vAxis1"

First steps - Motion

- for "MC_MoveVel_vAxis1" (MC_MoveVelocity)
 - "ex_MoveVel_vAxis1" (Execute)
 - "in_MoveVel_vAxis1" (InVelocity)
 - "ca_MoveVel_vAxis1" (CommandAborted)

and

- for "MC_Stop_vAxis1" (MC_Stop)
 - "ex_Stop_vAxis1" (Execute)

Trace recording has to be configured first.

Insert the object "Trace_vAxis1" via the context menu of "Application" **Add ► Trace....** Double-click to open the object.

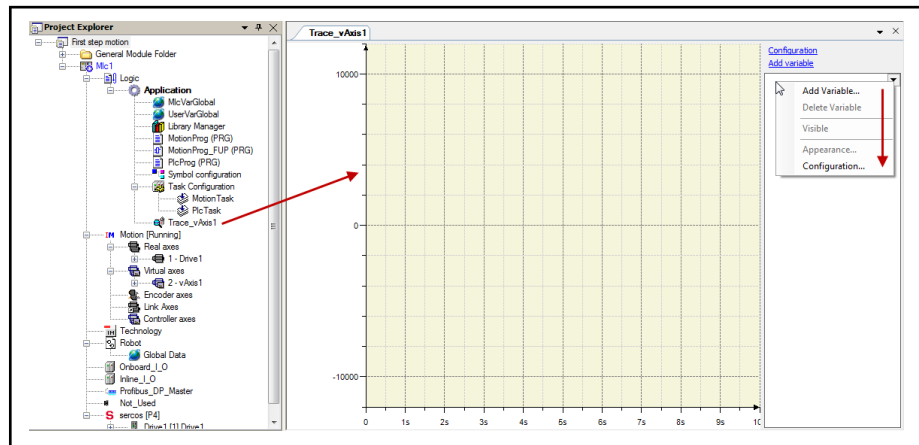


Fig. 3-68: Dialog: New trace object "Trace_vAxis1" with context menu

The following settings can be made for the "Trace" object via the context menu on the right:

- Configuration
- Display settings
- Add new variable

Configuration... Configure the new trace object first:

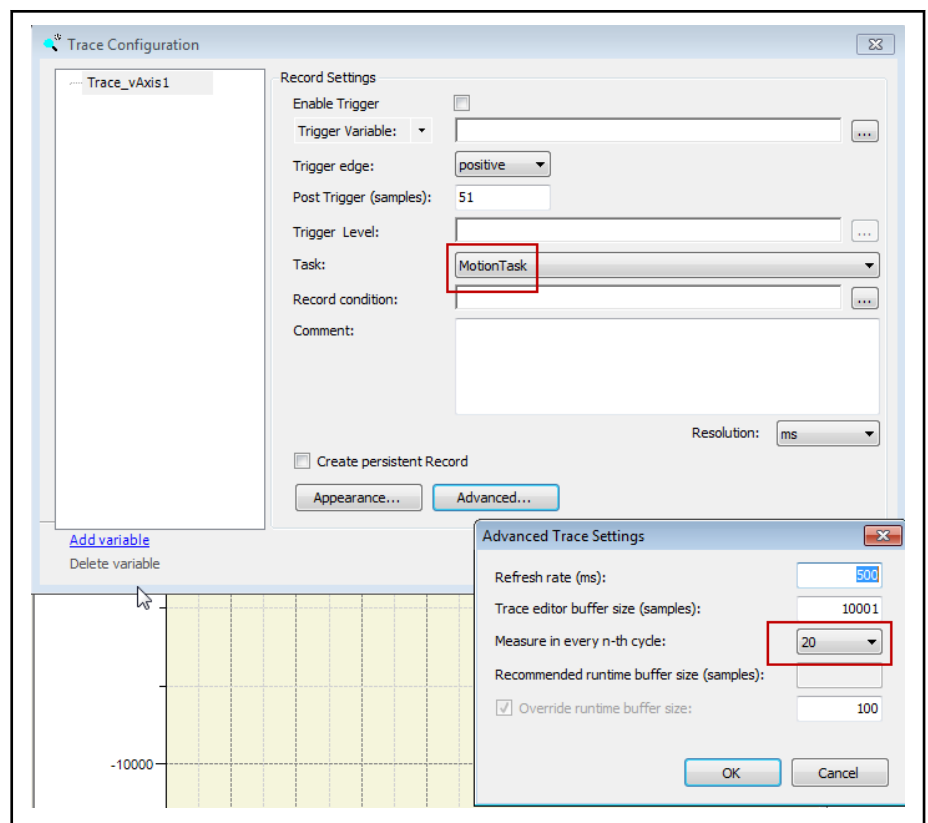


Fig. 3-69: Dialog: Trace configuration with extended "Trace" setting

The required settings are

- Task: **MotionTask**
- Recording in every **20** th cycle, i.e. $20 \times 2 \text{ ms} = 40 \text{ ms}$

Task: Select the task - for which the values of trace variables are to be read after processing - from the list of available tasks. Default: First task in the task configuration tree.

Recording condition: A Boolean variable, a value or a Boolean expression can be entered here. When this condition is met (value is "TRUE"), value recording is started. If nothing is entered, value recording starts when the trace configuration is loaded to the control and the application is running.

Measure in every: Enter the number of cycles after which the trace variable value is to be read again. Use the formula (Buffer size $\times$ Record in every x-th cycle $\times$ Task interval) to estimate the minimum recording time span.

Recommended runtime buffer size: Size of the ring buffer in bytes, which is currently provided in the runtime system to collect trace variable values in the currently configured record. The buffer content is then displayed as value curve(s) in the trace window. Default: 100.



If the configuration changes, the Trace object has to be logged in again.

Display settings

Settings for the display can now be made. These settings are also applied without logging in again.

First steps - Motion

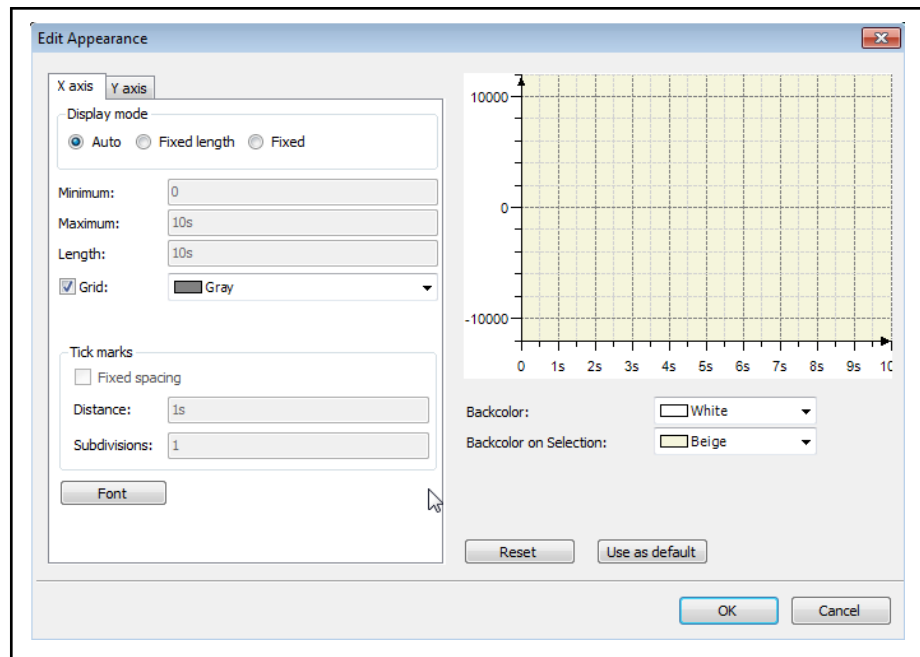


Fig. 3-70: Dialog: Display settings

Select **Auto**. A window with a recording **length** of 10 s appears over the curve. The **scale** is divided in a 1 s grid with a **scale division** of 1 s.

Selecting new variables

Now, name the variables required and set their appearance.

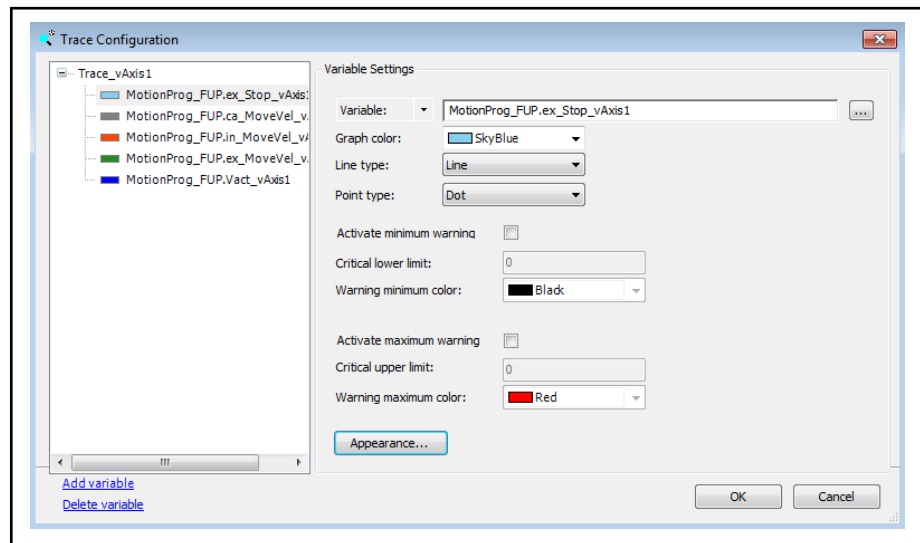


Fig. 3-71: Dialog: Creating new variables and specifying the appearance

Graph color and **Graph type** relate to the **Variables** curve.

In addition, if the respective limit is enabled, the parts of the curve that fall below a minimum value or exceed a maximum value can be offset in color against the remaining curve.

Menu to control the trace recording:

To display the context menu, right-click into the left section of the "Trace" window.

First steps - Motion

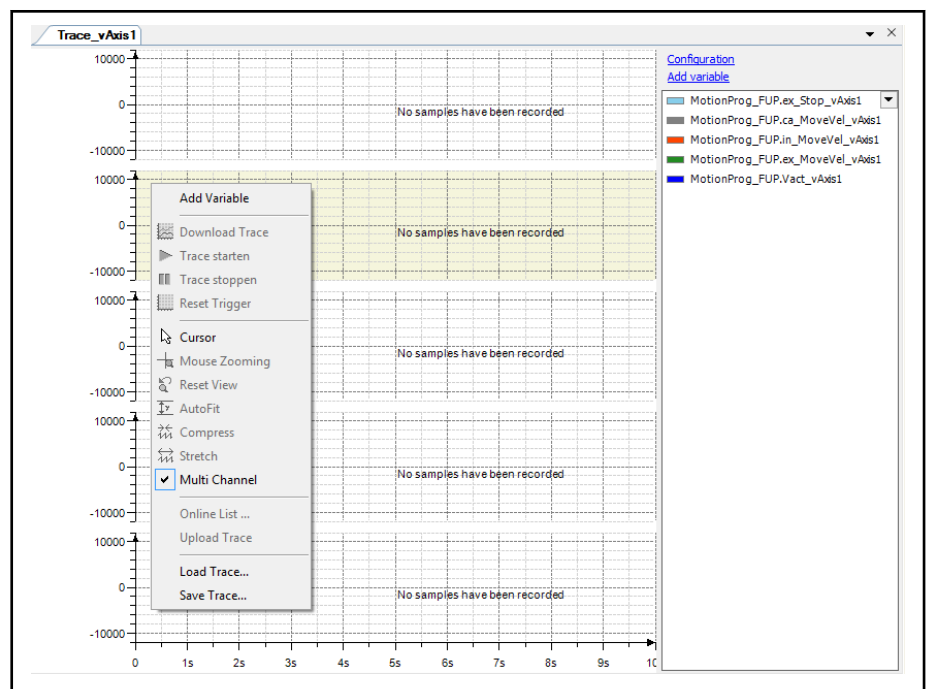


Fig. 3-72: Context menu to control the trace recording

Trace download: Before a configured trace recording on the control can be enabled for the application running there, the "Trace application" has to be loaded explicitly to the control.

Start/stop trace: This command starts or stops the display of the trace recording.

If the recording is stopped, the ► icon is positioned in front to the command.

If the recording is running, the ■ icon is positioned in front of the command and executing the command means that the recording is stopped, i.e. in the trace window, the section of the variable value curve(s) currently displayed is "frozen".

Multi-channel: This command allows to change the view of the trace curve. By default, the curves of all variables are shown in the same chart. In "Multi channel" view, the curve of each variable is visualized in its own diagram, but with an identical x-axis. Zooming and scrolling simultaneously affect the x-axis of all diagrams.

Cursor: When this command is enabled (default), the current position of the cursor in the coordinate system of the trace window is always displayed in the status bar of the programming system.

Scroll: In this mode, the mouse cursor is briefly displayed as a double arrow and by moving the cursor in the "Trace" window, the time axis of the recording currently displayed can be moved to the left or right.

Reset view: Use this command to reset the display of the recording(s) to the default values after it was changed by zooming for example.

- Performing trace:**
1. Log into the control.
Connect to the control.
 2. Trace download.

When the application is running, the data required is transmitted to the control via the "Trace download" context menu.

First steps - Motion

The trace recording then starts automatically.

3. Write values.

The following values are set as in ["Write values, input variables" on page 124](#):

- "ex_MoveVel_vAxis1" to "TRUE"
- "ex_Stop_vAxis1" to "TRUE"

4. End the recording.

Stop the trace via the "Trace" context menu item "Start/stop trace".

5. Evaluate the trace recording.

Set window data.

Shortcuts	Action in the trace diagram	Mouse control
<Alt>+<←/→> arrow keys	Horizontal scrolling of x-axis (time axis) in all diagrams	Enable scrolling with mouse, click into the diagram and scroll the x-axis with the mouse keeping the left mouse button pressed
<↑/↓> arrow keys with or without pressed <Alt> key Select a diagram with <tab>	Horizontal scrolling of the selected diagram	Enable scrolling with mouse, click into the diagram and scroll the x-axis with the mouse keeping the left mouse button and <Ctrl> pressed
<↑/↓> arrow keys, with or without pressed <Alt> key Select a diagram with <tab> Keep <Ctrl> pressed	The graph is moved in bigger steps	
<+/-> keys	Horizontal scrolling of all diagrams along the x-axis (time axis is expanded/compressed for all diagrams)	• Select "Traces" or "Compress": •  or  • or use the mouse wheel
<Ctrl>+<+/-> keys Select a diagram with <tab>	Vertical scrolling of all diagrams along the y-axis (y-axis is expanded/compressed for the selected graph)	

Tab. 3-2: Trace Editor shortcut with activated multi-channel

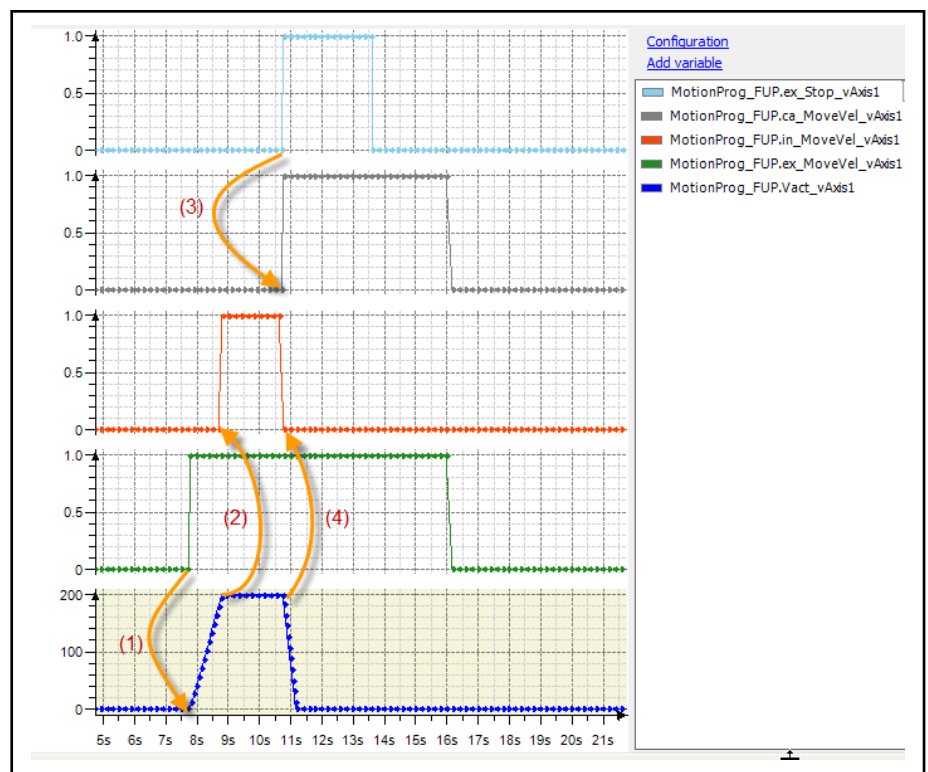


Fig. 3-73: Trace recording for the virtual axis vAxis1

Evaluate the trace recording

- (1) - Start of virtual axis startup
- (2) - Command speed reached
- (3) - Stop the virtual axis
- (4) - Virtual axis is decelerated

Note that the behavior of the virtual axis is as of a real axis with the same parameterization (compare the velocity characteristics, acceleration at 20 rad/s² and deceleration at 50 rad/s²).

3.5.5 Programming the real axis "Drive1" (ST)



From here onwards, the "real" drive technology is required
or
Select the second virtual axis **without** "MC_Power" as an alternative!

Before using a **real** axis, an instance of the PLCopen function block "MC_Power" is required.

Subsequently, one instance of "MC_GearIn" and "MC_GearOut" each are required.

The real axis is stopped with an "MC_Stop" instance.

Programming is executed as in [chapter 3.4.2 "Programming the virtual axis vAxis1, programming language FBD" on page 92.](#)

First steps - Motion

MC_Power

Add new line

Insert "MC_Power" and declare an "MC_Power_Drive1" instance

Assign inputs and declare input variables if necessary:

"Enable" → "en_Power_Drive1" of type "BOOL"

"Axis" → "MlcVarGlobal.Drive1" of type "AXIS_REF"

Outputs:

"Status" → "st_Power_Drive1" of type "BOOL"

"Error" → "er_Power_Drive1" of type "BOOL"

"ErrorID" → "ID_Power_Drive1" of type "ERROR_CODE"

"ErrorIdent" → "Ident_Power_Drive1" of type "ERROR_STRUCT"

MC_GearIn

Add new line

Insert "MC_GearIn" and declaring the "MC_GearIn_Drive1" instance

Assign inputs and declare input variables if necessary:

"Execute" → "ex_GearIn_Drive1" of type "BOOL"

"RatioNumerator" → 5

"RatioDenominator" → 10

"MasterFineadjust" → 20 (%)

"Master" → "MlcVarGlobal.vAxis1" of type "AXIS_REF"

"Slave" → "MlcVarGlobal.Drive1" of type "AXIS_REF"

Outputs:

"InGear" → "ig_GearIn_Drive1" of type "BOOL"

"Active" → "ac_GearIn_Drive1" of type "BOOL"

"CommandAborted" → "ca_GearIn_Drive1" of type "BOOL"

"Error" → "er_GearIn_Drive1" of type "BOOL"

"ErrorID" → "ID_GearIn_Drive1" of type "ERROR_CODE"

"ErrorIdent" → "Ident_GearIn_Drive1" of type "ERROR_STRUCT"

MC_GearOut

Add new line

Insert "MC_GearOut" and declare an "MC_GearOut_Drive1" instance

Assign inputs and declare input variables if necessary:

"Execute" → "ex_GearOut_Drive1" of type "BOOL"

"Slave" → "MlcVarGlobal.Drive1" of type "AXIS_REF"

Outputs:

"Done" → "do_GearOut_Drive1" of type "BOOL"

"Active" → "ac_GearOut_Drive1" of type "BOOL"

"Error" → "er_GearOut_Drive1" of type "BOOL"

"ErrorID" → "ID_GearOut_Drive1" of type "ERROR_CODE"

"ErrorIdent" → "Ident_GearOut_Drive1" of type "ERROR_STRUCT"

MC_Stop

Add new line

Insert "MC_Stop" and declare an "MC_Stop_Drive1" instance

Assign inputs and declare input variables if necessary:

"Execute" → "ex_Stop_Drive1" of type "BOOL"

"Deceleration" → 50 (rad/s²)

"Axis" → "MlcVarGlobal.Drive1" of type "AXIS_REF"

Outputs:

"Done" → "do_Stop_Drive1" of type "BOOL"

"Active" → "ac_Stop_Drive1" of type "BOOL"

"CommandAborted" → "ca_Stop_Drive1" of type "BOOL"

"Error" → "er_Stop_Drive1" of type "BOOL"

"ErrorID" → "ID_Stop_Drive1" of type "ERROR_CODE"

"ErrorIdent" → "Ident_Stop_Drive1" of type "ERROR_STRUCT"

Actual velocity "Vact_Drive1"

The final step is the implementation of a network in which the current velocity of the axis "Drive1" "Vact_Drive1" (data type REAL) can be read.

The current velocity is read from the data set of the axis (shared memory).

Data of all axes that can control an IndraMotion MLC, is included in the array

GVL_Base.AxisData: ARRAY [1..xx⁴] OF ML_AXISDATA_SM

The axis number has to be used as index.

The variable is provided with the name

GVL_Base.AxisData[1].rActualVelocity_i

for the real axis, axis number 1, ("Drive1" is provided with axis number 1, as "Drive1" was scanned before "vAxis1" was declared).

- Insert new line
- Insert `Vact_Drive1:=` , complete declaration
- using the input assistant to navigate to **Global Variables ▶ ML_BASE ▶ Global Variable List ▶ GVL_Base ▶ AxisData:**

⁴⁾ The number of controllable axes depends on the control type and its firmware, currently for the L65: 64 axes

First steps - Motion

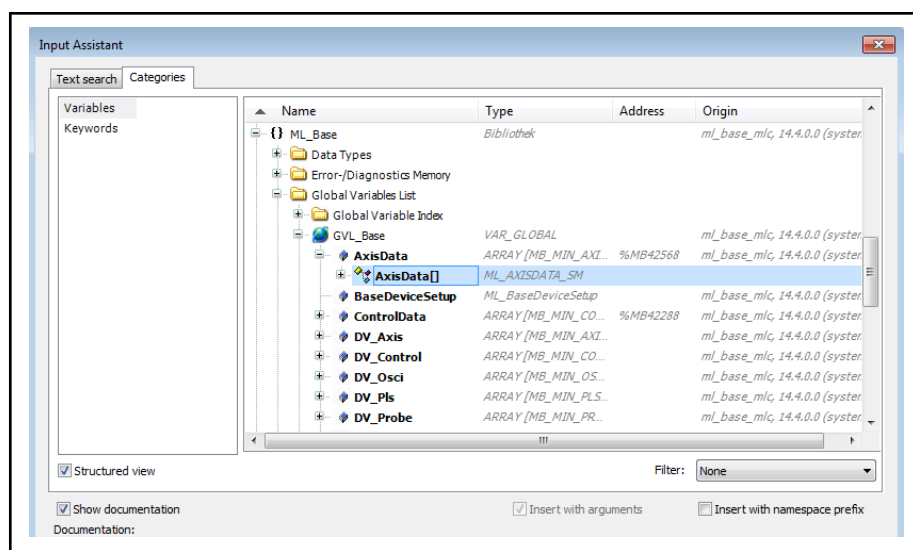


Fig. 3-74: Actual velocity value "vAxis1" from the Global Variable List of the "ML_Base.library"

Further entries are supported by the Intellisense function (automatic completion), i.e. after entering "." at "GVL_Base.AxisData[1].", all available elements are shown and can be selected.

Confirm "rActualVelocity_i" with <Enter>.

- Do not forget to complete the line with ";".

```
MC_Power_Drive1(  
    Enable:= en_Power_Drive1,  
    Status=> st_Power_Drive1,  
    Error=> er_Power_Drive1,  
    ErrorID=> ID_Power_Drive1,  
    ErrorIdent=> Ident_Power_Drive1,  
    Axis:= MlcVarGlobal.Drive1);  
  
MC_GearIn_Drive1(  
    Execute:=ex_GearIn_Drive1,  
    RatioNumerator:= 5,  
    RatioDenominator:= 10 ,  
    MasterFineadjust:= 20,  
    InGear=> ig_GearIn_Drive1,  
    Active=> ac_GearIn_Drive1,  
    CommandAborted=> ca_GearIn_Drive1,  
    Error=> er_GearIn_Drive1,  
    ErrorID=> ID_GearIn_Drive1,  
    ErrorIdent=> Ident_GearIn_Drive1,  
    Master:= MlcVarGlobal.vAxis1,  
    Slave:= MlcVarGlobal.Drive1);  
  
MC_GearOut_Drive1(  
    Execute:= ex_GearOut_Drive1,  
    Done=> do_GearOut_Drive1,  
    Active=> ac_GearOut_Drive1,  
    Error=> er_GearOut_Drive1,  
    ErrorID=> ID_GearOut_Drive1,  
    ErrorIdent=> Ident_GearOut_Drive1,  
    Slave:= MlcVarGlobal.Drive1);  
  
MC_Stop_Drive1(  
    Execute:= ex_Stop_Drive1,  
    Deceleration:= 50,  
    Done=> do_Stop_Drive1,  
    Active=> ac_Stop_Drive1,  
    CommandAborted=> ca_Stop_Drive1,  
    Error=> er_Stop_Drive1,  
    ErrorID=> ID_Stop_Drive1,  
    ErrorIdent=> Ident_Stop_Drive1,  
    Axis:=MlcVarGlobal.Drive1 );  
  
Vact_Drive1:=GVL_Base.AxisData[1].rActualVelocity_i;
```

Fig. 3-75: Implementing the real axis



AxisData[].rActualVelocity_i; and
GVL_Base.AxisData[].rActualVelocity_i;
Are equally used by the compiler.

First steps - Motion

3.5.6 Downloading the complete PLC project (ST)

The project is downloaded as described in section [chapter 3.5.3 "Downloading PLC subproject "virtual axis" vAxis1 \(ST\)" on page 122.](#)

The new project overwrites the old after issuing a prompt.

3.5.7 Workflow "Virtual axis → Electronic gear → Real axis "

The following procedure shows how to start and stop the axes and the gear.

1. Power on for the real axis.

Enable "en_Power_Drive1" (MC_Power) by "Write values".

st_Power_Drive1= TRUE (MC_Power).

2. Start the virtual axis.

Enable "ex_MoveVel_vAxis1" (MC_MoveVelocity) by "Write values".

Startup ends at Vact_vAxis1 = 200 rpm.

3. Follow the real axis.

Enable "ex_GearIn_Drive1" (MC_GearIn) by "Write values".

The expected gear transmission ratio of the gear is:

$Tr = (\text{RatioNumerator} / \text{RatioDenominator}) * (1 + \text{MasterFineadjust})$

for Vact_vAxis1 = 200 rpm results Vact_Drive1 = 120 rpm.



The MasterFineadjust value is specified in "%".

4. If the MC_GearIn is "InGear", the velocity "Vcom_vAxis1" of the virtual axis can be changed by "Write values" (double-click on the variable, enter the desired value). The real axis follows the virtual axis.

5. Disable "ex_GearIn_Drive1" (MC_GearIn) by "Write values".

in_GearIn_Drive1= FALSE, ac_GearIn_Drive1= TRUE.

The real axis continues to follow the virtual axis.

6. Terminate synchronicity.

Enable the "ex_GearOut_Drive1" (MC_GearOut) by "Write values".

The dependency of the real axis on the virtual axis is suspended. However, both axes continue to rotate with the same velocity.

7. Change the command velocity "Vcom_vAxis1".

The changed command velocity only affects the virtual axis. "Vact_Drive1" remains unchanged.

8. Disable "ex_GearOut_Drive1" by "Write values".

do_GearOut_Drive1= FALSE

9. Stop the virtual axis.

Enable "ex_Stop_vAxis1" (MC_Stop).

The virtual axis is decelerated to Vact_vAxis1 = 0 rpm.

10. Disable "ex_Stop_vAxis1" (MC_Stop).

11. Stop the real axis.

Enable "ex_Stop_Drive1" (MC_Stop).

The real axis is decelerated to Vact_Drive1= 0 rpm. After it stopped, it is actively held in this position.

12. Disable "ex_Stop_Drive1" (MC_Stop).
The active holding of the real axis finished.
13. Power off for the real axis.
Disable "en_Power_Drive1" (MC_Power) by "Write values".



Only possible in the "standstill" state after executing the stop at "Drive1".

3.5.8 Recording the motion of both axes (trace), ST

The "Trace" tool was used as in section [chapter 3.5.4 "Recording the motion of the axis vAxis1 \(trace\), ST" on page 125](#) to visualize the startup of the virtual axis and the synchronization.

The following workflow shows how to start and stop the axes and the gear for the trace recording.

1. Power on for the real axis.
Enable "en_Power_Drive1" (MC_Power).
2. Start the virtual axis.
Enable "ex_MoveVel_vAxis1" (MC_MoveVelocity).
3. Follow the real axis.
Enable "ex_GearIn_Drive1" (MC_GearIn).
4. Speed command value jump of the virtual axis.
Increase the speed command value to 250.
Is applied by "ex_MoveVel_vAxis1" TRUE ⇒ FALSE ⇒ TRUE.

The following figure shows the trace result:

First steps - Motion

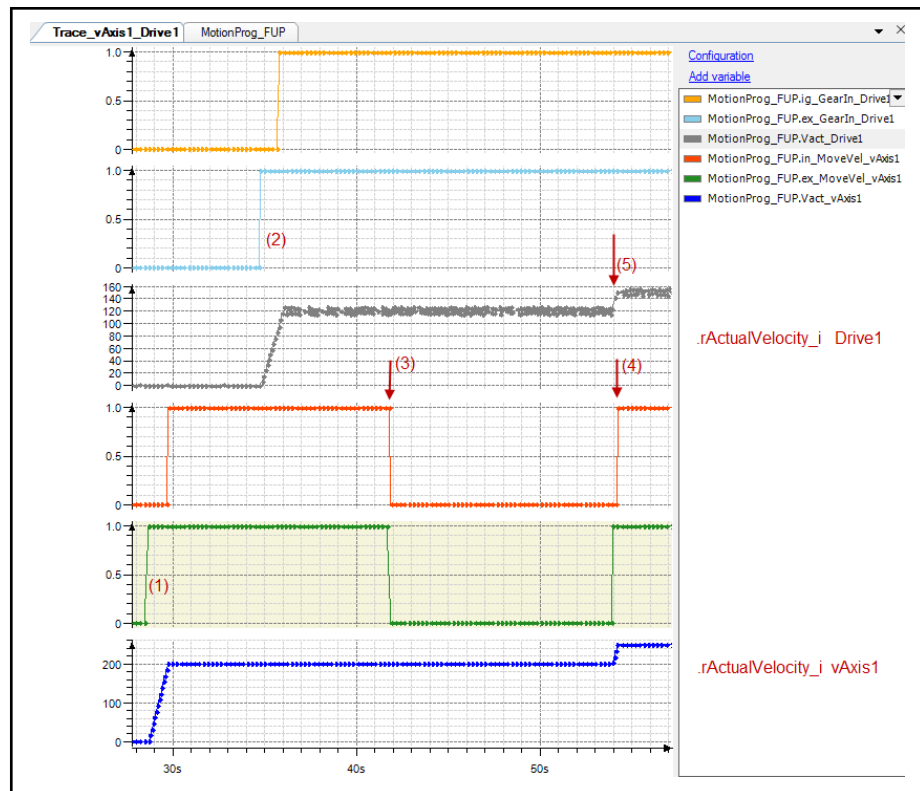


Fig. 3-76: Trace recording for both axes

Evaluate the trace recording

- (1) - Start of virtual axis startup
- (2) - Enable gear function, real axis startup
- (3) - Increase command speed of the virtual axis, "ex_MoveVel_vAxis1" TRUE ⇒ FALSE
- (4) - Command speed of the virtual axis is applied, "ex_MoveVel_vAxis1" FALSE ⇒ TRUE
- (5) - Jump response at the real axis

3.6 Restoring the project

Storage location An export file was generated from the created project that allows to restore parts of the project irrespective of the control hardware (IndraMotion MLC L25, L45, L65, L75, XM2, VPx) and the drive engineering.

The export file was created for the "Application" node and includes the following components:

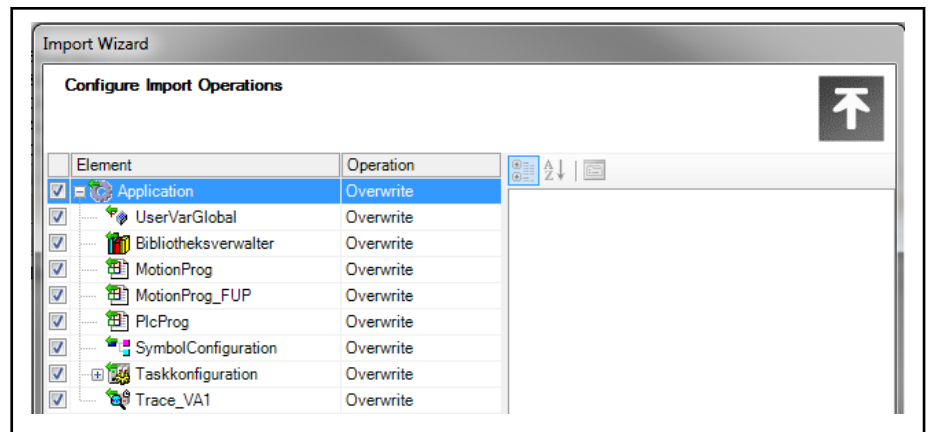


Fig. 3-77: Components of the export file

The specified components are overwritten during import.



The export file can be used as basis for "First steps - FlexProfile".

Restoring

To make the project run on the control (IndraMotion MLC L25, L45, L65, L75, XM2, VPx), proceed as follows:

1. Clear old data from the control.
 Clear the old memory content of the control.
 For details, refer to [chapter 3.3.1 "Resetting the control \(CMLx5\)" on page 78](#).
 The control memory is cleared and any possible Motion component present and the PLC component are ready for operation.
2. Create a new project.
 Create a new empty project (**File ► New ► Project**). Any project name can be selected, although "First steps Motion" is recommended.
 A new empty project has been created.
3. Create the control.
 Drag a control (IndraMotion MLC L25, L45, L65, L75, XM2, VPx) from the device library and drop it onto the empty project. Any control name can be selected. However, "Mlc1" is recommended.
 Complete device type and IP address.
 Perform a connection test.
 Complete the insertion as usual.
 The control is in the project now.
4. Create the Motion subproject.
 Scan the Sercos bus from the Sercos III node.
 The axis number "1" is assigned to the drive. The drive has to be re-named in "Drive1".
5. Declare a virtual axis. The axis obtains the axis number "2" and the name "vAxis1".
 In the Global Variable List "MlcVarGlobal", the axis numbers and axis names are provided as axis references of the "AXIS_REF" type.

First steps - Motion

The Motion subproject can be downloaded (refer to [chapter 3.3.8 "Downloading the Motion subproject" on page 88](#)).

6. Import the export file "Application_FirstSteps_IndraLogic_Motion".
In the Project Explorer, expand the control node up to **Logic ► Application**.
Select **Import...** from the control context menu of the "Application" node.
Select the export file "Application_FirstSteps_IndraLogic_Motion.iwxx" in "<Drive>:All Programs\Rexroth\IndraWorks\Projects". The "Configure import operations" dialog opens. Select the "Overwrite" operation for each element in this import dialog. To start the import, press **Finish**.
The required files were copied. The project can now run on the control.

4 First steps - FlexProfile

4.1 Project goal "First steps – FlexProfile"

A virtual axis to run at a constant velocity is specified. The axis functions as a master axis for a real axis coupled via a "FlexProfile" MotionProfile.

The real axis contains, for instance, an "IndraDrive M" drive connected to the control via a Sercos III cable.

The MotionProfile consists of a motion step that produces a simple rest-in-rest motion. A slave axis distance of 180° is traversed on a master axis revolution of 360°. The following figure shows the respective motion diagram.

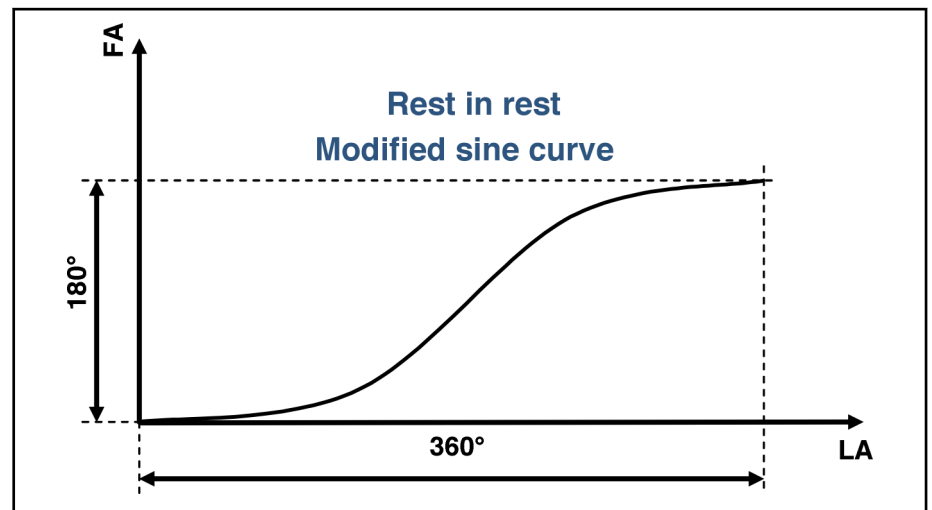


Fig. 4-1: Coupling profile between master axis (vAxis1) and slave axis (Drive1)

The slave axis can be provided with up to four MotionProfiles, each with a maximum of 16 (64) steps.¹⁾

The FlexProfile 0 in "Drive1" is used for "First steps – FlexProfile".

There are two options to create a FlexProfile:

- IndraWorks, "FlexProfile" dialog of the slave axis, refer to [chapter 4.4.2 "Creating the FlexProfile via IndraWorks seadogs" on page 144](#)
- or
- IndraLogic, PLC program code with a function block "MB_ChangeFlexProfileSet" to transfer data from the PLC program to the slave axis, see [chapter 4.4.3 "Creating the FlexProfile via PLC" on page 146](#)
 - For programming in ST, refer to ["FlexProfile, data transfer in ST" on page 147](#)
 - For programming in FBD, refer to ["FlexProfile, data transfer in in FBD" on page 148](#)

Actual processing is started via the "ML_FlexProfile" function block ("ML_FlexProfile", undertakes the task (connecting a slave axis to the master axis) which "MC_GearIn" performed in "First steps – Motion".

The "ML_FlexProfile" function block is disabled by the "MC_GearOut" function block.

- For programming in ST see ["ML_FlexProfile in ST" on page 150](#)

¹⁾ The number of steps within a FlexProfile depends on the firmware version and the control type.

First steps - FlexProfile

- For programming in FBD see ["ML_FlexProfile in FBD" on page 152](#)

4.2 Technical requirements

- Software:
 - PC with installed IndraWorks Engineering for IndraMotion MLC 14VRS
- Control:
 - IndraMotion MLC L25
 - IndraMotion MLC L45
 - IndraMotion MLC L65
 - IndraMotion MLC L75
 - IndraMotion MLC XM2
 - IndraMotion MLC VPx

A control, e.g. of type **"IndraMotion MLC L65"** has to be available with the current firmware 14VRS. Additionally, the following requirements have to be met:

- The IP address is known
- The control is connected to the programming device (PC) via Ethernet

Drive controller, e.g. type **"IndraDrive M"** with current firmware:

- The drive controller is connected to the control via Sercos III
- The address of the drive controller at the Sercos bus is scanned
or
- a bridged Sercos bus if there is no drive



For further information on the FlexProfile, refer to the documentation

IndraMotion MLC 14VRS Functional Description, chapter "Flex-Profile".

The FlexProfile functionality is supported by the following axis types (slave axis):

- Virtual axis
- SercosDrive
- IndraDrive with interpolation in the control

Use the virtual axis "vAxis1" as master axis and the real axis "Drive1", IndraDrive **with interpolation in the control**, as slave axis.

4.3 Creating the Motion subproject

To make the project run on the control (IndraMotion MLC L25, L45, L65, L75, XM2, VPx), proceed as follows:

1. Clear old data from the control.

Clear the old memory content of the control.

For details, refer to [chapter 3.3.1 "Resetting the control \(CMLx5\)" on page 78](#).

The control memory is cleared and any possible Motion component present and the PLC component are ready for operation.

2. Create a new project.

Create a new empty project (**File ► New ► Project**). Any project name can be selected, although "First steps FlexProfile" is recommended.

A new empty project has been created.

3. Create the control.

Drag a control (IndraMotion MLC L25, L45, L65, L75, XM2, VPx) from the device library and drop it onto the empty project. Any control name can be selected. However, "Mlc1" is recommended.

Complete device type and IP address.

Perform a connection test.

Complete the insertion as usual.

The control is in the project now.

4. Create the Motion subproject.

Scan the Sercos bus (via the context menu "Sercos" node **Scan bus configuration**).



Check that the box in the "Ipo drive" column is not selected. Remove possible tick in checkbox!

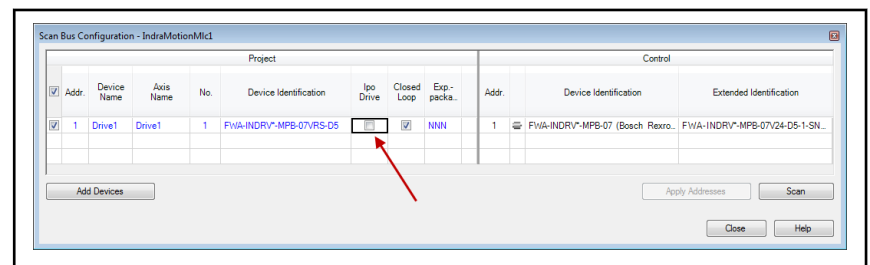


Fig. 4-2: Scanning bus configuration

Axis number "1" is assigned to the drive.

5. Declare a virtual axis. Assign axis number "2" to the axis.

In the Global Variable List "MlcVarGlobal", the axis numbers and axis names are provided as axis references of the "AXIS_REF" type.

Download can take place (see [chapter 3.3.8 "Downloading the Motion subproject" on page 88](#)), the Motion component is then in the "Ready for operation (BB)" state.

6. Import the export file "Application_FirstSteps_IndraLogic_Motion.iwx".

In the Project Explorer, expand the control node up to **Logic ► Application**.

Select **Import...** from the control context menu of the "Application" node.

Select the export file "Application_FirstSteps_IndraLogic_Motion.iwx" in "<Drive>:All Programs\Rexroth\IndraWorks\Projects". Select the "Overwrite" operation for each element in this import dialog. To start the import, press **Finish**.

The required files were copied. The imported project can now run on the control.

First steps - FlexProfile

4.4 Creating a FlexProfile

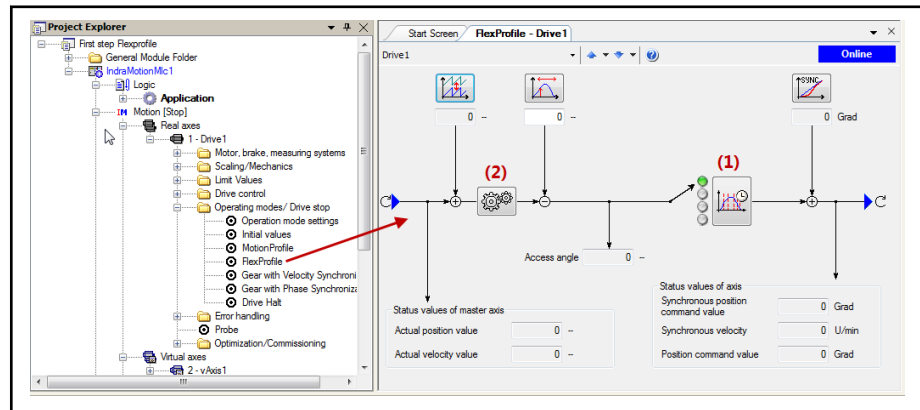
4.4.1 Creating a FlexProfile, general information



Creating or modifying a FlexProfile requires the control and PC to be coupled **online**.

4.4.2 Creating the FlexProfile via IndraWorks seadogs

Click on the "FlexProfile" entry in the Project Explorer under "Real axes" → "Drive1" to open a dialog to configure the general profile.



(1) Transition to "FlexProfile 0"... "FlexProfile 3" dialogs

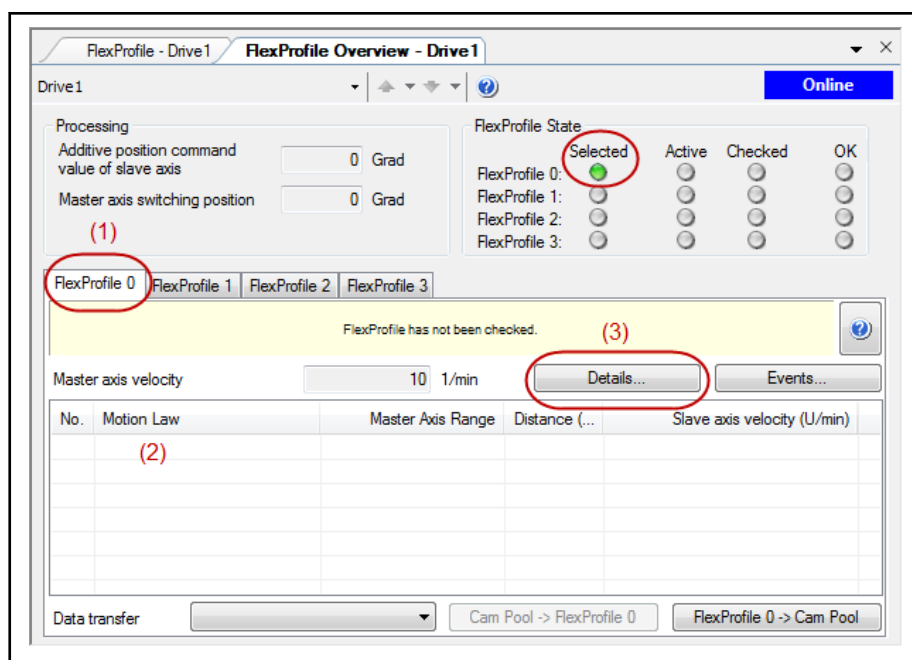
(2) Transition to the "Gear settings" dialog

Fig. 4-3: "FlexProfile" dialog with action chain between master and slave axis

To open the "FlexProfile overview", click on (1).

The following dialog shows the processing status for the four FlexProfiles assigned to the axis (tabs "FlexProfile 0" to "FlexProfile 3").

First steps - FlexProfile



- (1) Tabs "FlexProfile 0" to "FlexProfile 3". "FlexProfile 0" is selected
- (2) Overview on the motion steps 0...15 (max. 64 motion steps) of the selected FlexProfile (write-protected)
- (3) Transition to the details of the selected FlexProfile

Fig. 4-4: Dialog: "FlexProfile" - Overview window; FlexProfile 0 is selected

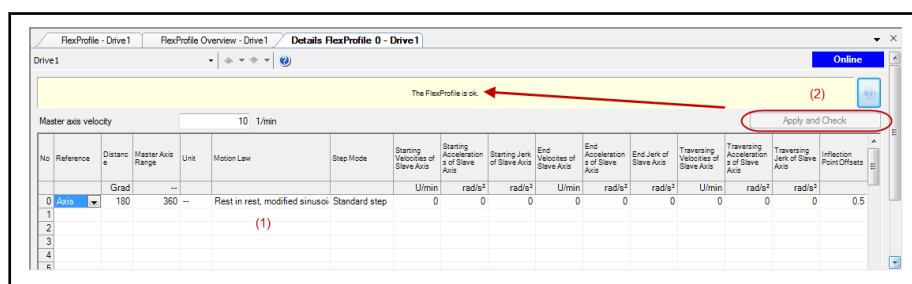
A new MotionProfile can be created via "Details" (1).

The FlexProfile editor opens automatically.

An empty configuration table is shown in which each line defines one profile step [0...15 (max. 64 motion steps)].

The master axis sources, motion laws and motion step types required can be selected from a drop down menu.

Enter the data as shown in the figure below and configure the motion sequence.



- (1) Configuration of the individual motion steps 0...15 (max. 64 motion steps) of the selected FlexProfile
- (2) Applying MotionProfile data and checking whether the profile is correct

Fig. 4-5: Dialog: "FlexProfile" - Configuration window

For FlexProfile 0, the following data was entered into the dialog above:

Reference	Axis
Distance	180 degrees

First steps - FlexProfile

Master axis section	360
Unit	-
Motion law	Rest-in-rest, modified sine curve
Step type	Standard step
...more...	0
Inflection point offsets	0.5

After the FlexProfile configuration has been completed, the changes have to be applied to the control. To trigger the transfer, press **Apply and Check** (2).



A MotionProfile can only be started, when it is checked and marked as valid.

A modified configuration resets the status "Checked".

A currently active MotionProfile cannot be modified.

In the "FlexProfile overview" window, the state of the MotionProfile is indicated as "Checked" and "OK" after successful application.

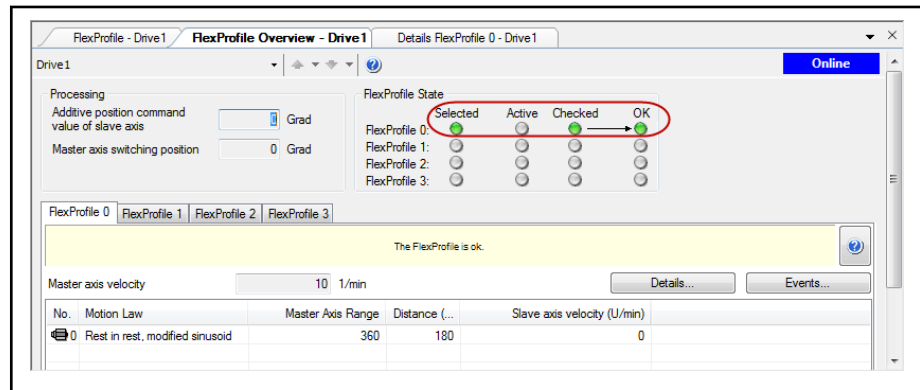


Fig. 4-6: Dialog: "FlexProfile" – FlexProfile overview window, FlexProfile 0, checked → ok

The configured MotionProfile "FlexProfile 0" can now be started with the PLC function block "ML_FlexProfile".

4.4.3 Creating the FlexProfile via PLC

A FlexProfile should now be configured via the PLC.



Please note that the execution time of the "MB_ChangeFlexProfileSet" function block can be longer depending on the size and the complexity of a profile to be written.

Therefore, only use the function block in freely running PLC processes.

Thus, for "First steps - FlexProfile", programming takes place in the programs ["FlexProfile, data transfer in ST" on page 147](#) or ["FlexProfile, data transfer in in FBD" on page 148](#). Both programs are alternatively controlled via the PlcTask.

Usage in a Sercos-synchronous task (e.g. "MotionTask") is not recommended for real programs.

Tasks to be solved

1. Preparing data for the motion steps 0...15 (max. 64 motion steps) of the four FlexProfiles "FlexProfile 0"... "FlexProfile 3".

```
profileAxis: ARRAY[0..15] OF MB_PROFILE_STEP;
```

The ARRAY "profileAxis" thus contains the data of a FlexProfile.

For "First steps - FlexProfile", the data set "0" of type "MB_PROFILE_STEP" is thus assigned in our program.

This data set corresponds to [fig. 4-5 "Dialog: FlexProfile – Configuration window" on page 145](#).

2. Transferring FlexProfile data from the PLC program to the control (axis Drive1).

This task is executed by the instance "ChangeFlexProfileSetDrive1)" of type "MB_ChangeFlexProfileSet".

This instance corresponds to [fig. 4-6 "Dialog: FlexProfile – FlexProfile overview window, FlexProfile 0, checked → ok" on page 146](#).

FlexProfile, data transfer in ST

When providing the data set, the variable "profileAxis" is declared first.

Subsequently, the desired values are assigned to its structure elements.

The structure elements are listed in the library manager (**ML_Base ► Data types ► Flex structs ► MB_PROFILE_STEP**).

The library name is automatically supplemented according to the control created in the Project Explorer, "ml_base_mlc" in the following figure.

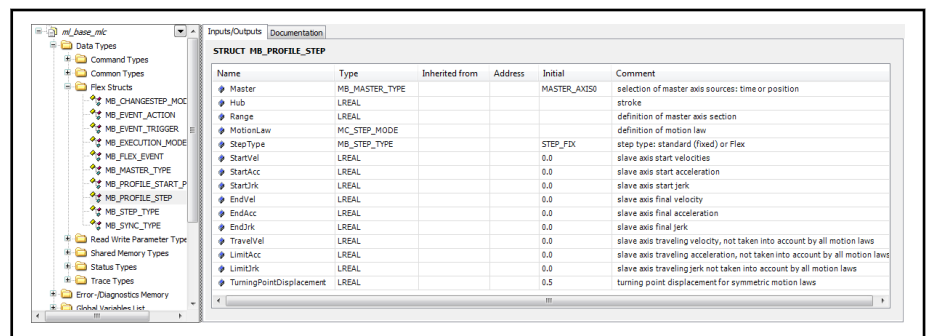


Fig. 4-7: Description of the structure "MB_PROFILE_STEP" in the library manager

Use the "." operator with the following Intellisense function for the input (auto completion).

First steps - FlexProfile

```

1  PROGRAM PlcProg
2  VAR
3    profileAxis: ARRAY[0..15] OF MB_PROFILE_STEP;
4  END_VAR

1  profileAxis[0].Master:= MB_MASTER_TYPE.MASTER_AXIS0;
2  profileAxis[0].Hub:=180;
3  profileAxis[0].Range:=360;
4  profileAxis[0].MotionLaw:=MC_STEP_MODE.REST_IN_REST_MOD_SINE;
5  profileAxis[0].StepType:=MB_STEP_TYPE.STEP_FIX;
6  profileAxis[0].StartVel:=0;
7  profileAxis[0].StartAcc:=0;
8  profileAxis[0].StartJrk:=0;
9  profileAxis[0].EndVel:=0;
10 profileAxis[0].EndAcc:=0;
11 profileAxis[0].EndJrk:=0;
12 profileAxis[0].TravelVel:=0;
13 profileAxis[0].LimitAcc:=0;
14 profileAxis[0].LimitJrk:=0;
15 profileAxis[0].TurningPointDisplacement:=0.5;
16

```

Fig. 4-8: "profileAxis" data set with value assignment in ST

From line 17 onwards, the instance "ChangeFlexProfileSetDrive1" of the function block "MB_ChangeFlexProfileSet" is usually entered to transfer the data array to the control.

```

13  profileAxis[0].LimitAcc:=0;
14  profileAxis[0].LimitJrk:=0;
15  profileAxis[0].TurningPointDisplacement:=0.5;
16
17  ChangeFlexProfileSetDrive1(
18    Execute:= ex_ChangeFlexProfileSetDrive1,
19    SetNumber:= SetNum,
20    NumberOfSteps:= 1,
21    MasterAxisVelocity:= 10,
22    ProfileStepAdr:= ADR(ProfileAxis),
23    Done=> do_ChangeFlexProfileSetDrive1,
24    Active=> ac_ChangeFlexProfileSetDrive1,
25    Error=> er_ChangeFlexProfileSetDrive1,
26    ErrorID=> ID_ChangeFlexProfileSetDrive1,
27    ErrorIdent=> Ident_ChangeFlexProfileSetDrive1,
28    Axis:=Drive1);

```

Fig. 4-9: Function block to transfer the data array to the control (SetNum = 0 for FlexProfile 0)

The data set is transferred by "Write values" at the input "ex_ChangeFlexProfileSetDrive1", TRUE/FALSE.

The state of the FlexProfile is then checked/ok.



Use SetNum= 2 to write to "FlexProfile 2" and compare the result with "FlexProfile 0", input via the IndraWorks dialog, in the Flex-Profile editor.

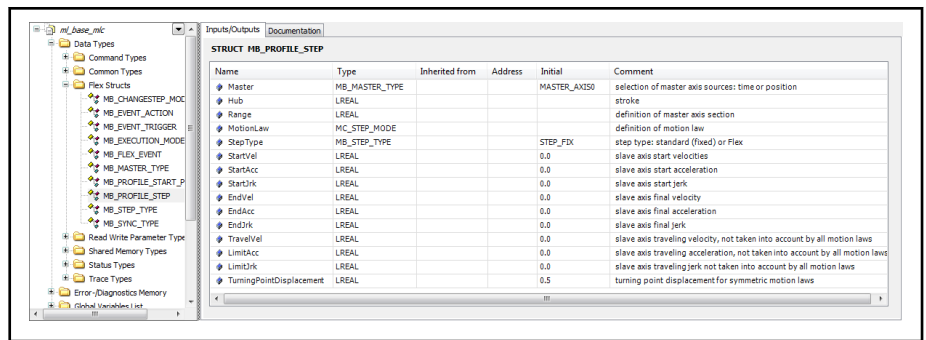
FlexProfile, data transfer in in FBD

When providing the data set, the variable "profileAxis" is declared first.

Subsequently, the desired values are assigned to its structure elements.

The structure elements are listed in the library manager (ML_Base ► Data types ► Flex structs ► MB_PROFILE_STEP).

First steps - FlexProfile



The screenshot shows the 'STRUCT MB_PROFILE_STEP' in the library manager. The table below lists the fields and their types.

Name	Type	Inherited from	Address	Initial	Comment
Master	MB_MASTER_TYPE			MASTER_AXIS0	selection of master axis sources: time or position
Hub	LREAL				stroke
Range	LREAL				definition of master axis section
MotionLaw	MC_STEP_MODE				definition of motion law
StepType	MB_STEP_TYPE			STEP_FIX	step type: standard (fixed) or Flex
StartVel	LREAL			0.0	slave axis start velocities
StartAcc	LREAL			0.0	slave axis start acceleration
StartJrk	LREAL			0.0	slave axis start jerk
EndVel	LREAL			0.0	slave axis final velocity
EndAcc	LREAL			0.0	slave axis final acceleration
EndJrk	LREAL			0.0	slave axis final jerk
TravelVel	LREAL			0.0	slave axis traveling velocity, not taken into account by all motion laws
LimitAcc	LREAL			0.0	slave axis traveling acceleration, not taken into account by all motion laws
LimitJrk	LREAL			0.0	slave axis traveling jerk, not taken into account by all motion laws
TurningPointDisplacement	LREAL			0.5	turning point displacement for symmetric motion laws

Fig. 4-10: Description of the structure "MB_PROFILE_STEP" in the library manager

Use the "." operator with the following Intellisense function to input the assignments (auto completion).

```

1 PROGRAM PlcProg_FUP
2 VAR
3   profileAxis: ARRAY[0..15] OF MB_PROFILE_STEP;
4 END_VAR

```

Below the code, the Intellisense function shows the following assignments for the 'profileAxis' array:

- 1 MB_MASTER_TYPE.MASTER_AXIS0 — profileAxis[0].Master
- 2 180 — profileAxis[0].Hub
- 3 360 — profileAxis[0].Range
- 4 MC_STEP_MODE.REST_IN_REST_... — profileAxis[0].MotionLaw
- 5 MB_STEP_TYPE.STEP_FIX — profileAxis[0].StepType
- 6 0 — profileAxis[0].StartVel
- 7 0 — profileAxis[0].StartAcc
- 8 0 — profileAxis[0].StartJrk
- 9 0 — profileAxis[0].EndVel
- 10 0 — profileAxis[0].EndAcc
- 11 0 — profileAxis[0].EndJrk
- 12 0 — profileAxis[0].TravelVel
- 13 0 — profileAxis[0].LimitAcc
- 14 0 — profileAxis[0].LimitJrk
- 15 0.5 — profileAxis[0].TurningPoin...

Fig. 4-11: "profileAxis" data set with value assignment in FBD

The instance "ChangeFlexProfileSetDrive1" of the function block "MB_ChangeFlexProfileSet" is usually input as network number 16 to transfer the data array to the control.

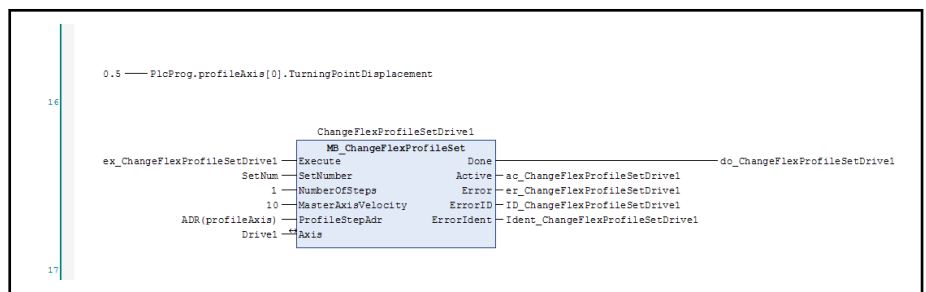


Fig. 4-12: Function block to transfer the data array to the control (SetNum = 0 for FlexProfile 0)

The data set is transferred by "Write values" at the input "ex_ChangeFlexProfileSetDrive1", TRUE/FALSE.

The state of the FlexProfile is then checked/ok.



Use SetNum= 1 to write to "FlexProfile 1" and compare the result with "FlexProfile 0", input via the IndraWorks dialog, in the Flex-Profile editor.

4.5 Completing the PLC subproject, executing a FlexProfile motion step

When the MotionProfile is written to the axis, it can be started by executing the "ML_FlexProfile" function block. This function block creates a coupling between the master axis and the slave axis using the MotionProfile previously specified and selected and synchronizes the slave axis.

Starting from moment, a slave axis motion follows the master axis motion. In the simplest case, the master axis rotates at constant velocity, for example by calling the PLCopen function block "MC_MoveVelocity". The slave axis travels along the desired MotionProfile.

Import the export file "Application_FirstSteps_IndraLogic_Motion.iwx" from the directory "<Drive>:\All Programs\Rexroth\IndraWorks\Projects".

For the import in the Project Navigator, open the control node **Logic ▶ Application** and select **Import...** in the context menu of the "Application" node.

This file contains the programs already tested in "First Steps Motion" in Structured Text (MotionProg) and in FBD (MotionProg_FBD).

The following modifications are required:

1. Change **VcomvAxis1** from 200 rpm to 10 rpm.
2. **MoveVelvAxis1:**
 "Acceleration" → 100 [rad/s²]
 "Deceleration" → 100 [rad/s²]
3. Delete the "MC_GearIn_Drive1" instance of "MC_GearIn". This instance is replaced by the "FlexProfileDrive1" instance of the "ML_FlexProfile".

"ML_FlexProfile" in ST

As described above, add a function block of type "ML_FlexProfile" and assign it as described in the following.

Add "ML_FlexProfile" and declare a "FlexProfileDrive1" instance:

Declaration section of "MotionProg":

1. Insert an empty line in front of END_VAR.
2. Enter `FlexProfileDrive1:`, press <F2> to open the input assistant and
 deselect the option "Insert with arguments" under **User-defined types**
 ML_PLCOpen ▶ POUs ▶ Motion ▶ ML_FlexProfile.
3. Complete line:

```
FlexProfileDrive1: ML_FlexProfile;
```

Add empty line instead of "GearInDrive1"

To open the input assistant, press <F2> and select the option "Insert with arguments" under **Instance calls ▶ FlexProfileDrive1**

Assign inputs and declare input variables if necessary:

"Execute" → "ex_FlexProfileDrive1" of type "BOOL"

"RatioNumerator" → 1

First steps - FlexProfile

"RatioDenominator" → 1
"MasterFineadjust" → 0
"SetSelection" → 0
"UseSwitchingPosition" → "FALSE"
"SwitchingPosition" → 0
"ProfileEntry" → of type "MB_PRO-
FILE_START_POINT.SLAVE_CURRENT_MASTER_CURRENT",
"ML_Base.library", data types, flex structs, "MB_PROFILE_START_POINT"
"MasterOffset" → 0
"SlaveOffset" → 0
"SyncType" → of type "MB_SYNC_TYPE"
Initial value: "SYNC_RAMPIN_CATCH_UP", "ML_Base.library"
"SyncVelocity" → 100
"SyncAcceleration" → 100
"ExecutionMode" → of type "MB_EXECUTION_MODE"
Initial value: "EXECUTE_CYCLIC", "ML_Base.library"
"Master" → "vAxis1" of type "AXIS_REF"
(Already declared variable from the range "Variables", Application, MlcVarGlobal)
"Slave" → "Drive1" of type "AXIS_REF"
(Already declared variable from the range "Variables", Application, MlcVarGlobal)

Outputs:

"InSync" → "is_FlexProfileDrive1" of type "BOOL"
"Active" → "ac_FlexProfileDrive1" of type "BOOL"
"CommandAborted" → "ca_FlexProfileDrive1" of type "BOOL"
"Error" → "er_FlexProfileDrive1" of type "BOOL"
"ErrorID" → "ID_FlexProfileDrive1" of type "ERROR_CODE"
"ErrorIdent" → "Ident_FlexProfileDrive1" of type "ERROR_STRUCT"

First steps - FlexProfile

```

33   Axis:= MlcVarGlobal.Drive1);
34
35   FlexProfileDrive1(
36     Execute:= ex_FlexProfileDrive1,
37     RatioNumerator:= 1,
38     RatioDenominator:= 1,
39     MasterFineadjust:= 0,
40     SetSelection:= 0,
41     UseSwitchingPosition:= FALSE,
42     SwitchingPosition:= 0,
43     ProfileEntry:= MB_PROFILE_START_POINT.SLAVE_CURRENT_MASTER_CURRENT,
44     MasterOffset:= 0,
45     SlaveOffset:= 0,
46     SyncType:= MB_SYNC_TYPE.SYNC_RAMPIN_CATCH_UP,
47     SyncVelocity:= 100,
48     SyncAcceleration:= 100,
49     ExecutionMode:= MB_EXECUTION_MODE.EXECUTE_CYCLIC,
50     InSync=> is_FlexProfileDrive1,
51     Active=> ac_FlexProfileDrive1,
52     CommandAborted=> ca_FlexProfileDrive1,
53     Error=> er_FlexProfileDrive1,
54     ErrorID=> ID_FlexProfileDrive1,
55     ErrorIdent=> Ident_FlexProfileDrive1,
56     Master:=vAxis1,
57     Slave:= Drive1);
58
59   MC_GearOut_Drive1(
60     Execute:= ex_GearOut_Drive1,
61     Done=> do_GearOut_Drive1,

```

Fig. 4-13: "ML_FlexProfile" is inserted and assigned (ST)

"ML_FlexProfile" in FBD

As described above, add a function block of type "ML_FlexProfile" and assign it as described in the following.

Add "ML_FlexProfile" and declare a "FlexProfileDrive1" instance:

Assign inputs and declare input variables if necessary:

"Execute" → "ex_FlexProfileDrive1" of type "BOOL"

"RatioNumerator" → 1

"RatioDenominator" → 1

"MasterFineadjust" → 0

"SetSelection" → 0

"UseSwitchingPosition" → FALSE

"SwitchingPosition" → 0

"ProfileEntry" → of type "MB_PROFILE_START_POINT.SLAVE_CURRENT_MASTER_CURRENT",

"ML_Base.library", data types, flex structs, "MB_PROFILE_START_POINT"

"MasterOffset" → 0

"SlaveOffset" → 0

"SyncType" → of type "MB_SYNC_TYPE"

Initial value: SYNC_RAMPIN_CATCH_UP, ML_Base.library

"SyncVelocity" → 100

"SyncAcceleration" → 100

"ExecutionMode" → of type "MB_EXECUTION_MODE"

Initial value: EXECUTE_CYCLIC, ML_Base.library

"Master" → "vAxis1" of type "AXIS_REF"

(Already declared variable from the range "Variables", Application, MlcVarGlobal)

"Slave" → "Drive1" of type "AXIS_REF"

(Already declared variable from the range "Variables", Application, MlcVarGlobal)

Outputs:

"InSync" → "is_FlexProfileDrive1" of type "BOOL"

"Active" → "ac_FlexProfileDrive1" of type "BOOL"

"CommandAborted" → "ca_FlexProfileDrive1" of type "BOOL"

"Error" → "er_FlexProfileDrive1" of type "BOOL"

"ErrorID" → "ID_FlexProfileDrive1" of type "ERROR_CODE"

"ErrorIdent" → "Ident_FlexProfileDrive1" of type "ERROR_STRUCT"

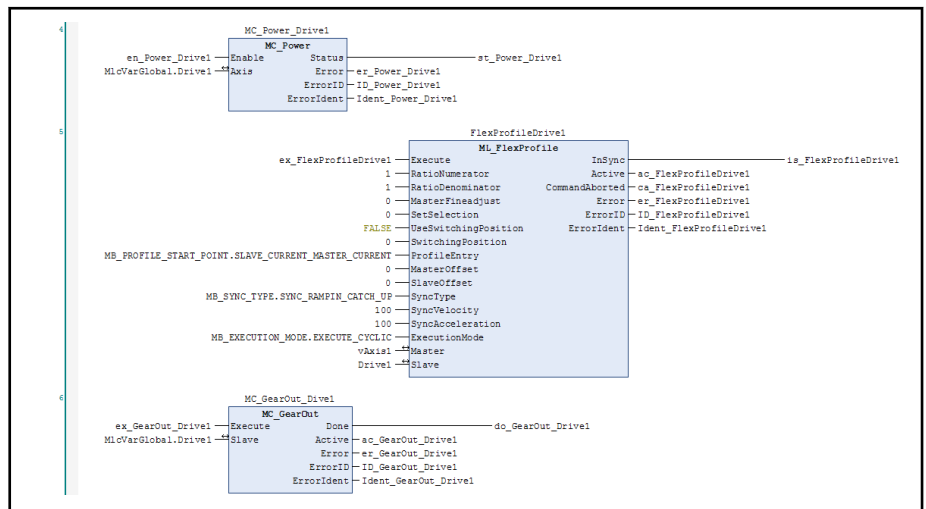


Fig. 4-14: "ML_FlexProfile" is inserted and assigned (FBD)

4.6 Downloading and testing the PLC subproject

The previously implemented motion sequence still has to be compiled, applied to the control and started.

For the compilation, go to the via IndraWorks menu item

- **Build ► Build**, compilation of the added part of the application
- **Build ► Generate code**, the command allows to generate a code and to check compilation errors.

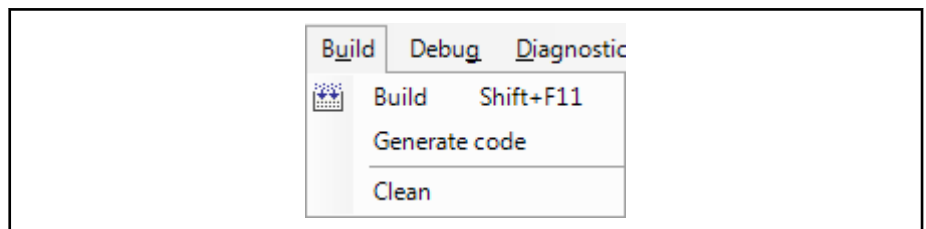


Fig. 4-15: Build application

First steps - FlexProfile



The menu items **Create ► Recompile** and **Create ► Clean all** are not available by default anymore as in 13VRS. However, these commands can be added again via **Tools ► Customize ► Commands ► Create**.

To transfer the compiled data to the target control, select **Debug ► Login** from the IndraWorks menu bar (refer to the figure below).

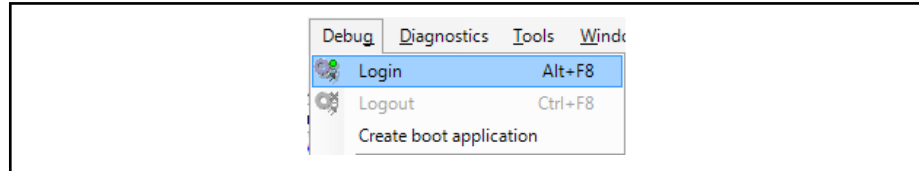


Fig. 4-16: Transferring compiled data to the control

A prompt asking for agreement to download is displayed depending on whether an application is already loaded and whether this application is a previous version of the current application.

The application is still stopped after a successful transmission (red in the Project Explorer and in the status bar) and the variables in declaration and implementation are prepared to display the values. However, the values are invalid.

Start the processing in the control via **Debug ► Start**.

4.7 Workflow "Virtual axis → FlexProfile 0 → Real axis "

The following workflow shows how to start and stop the axes and the Flex-Profile:

1. Power on for the real axis.
 Enable "en_Power_Drive1" (MC_Power) by "Write values".
 st_Power_Drive1= TRUE (MC_Power).
2. Start the virtual axis.
 Enable "ex_MoveVel_vAxis11" (MC_MoveVelocity) by "Write values".
 Startup ends at Vact_vAxis1 = 10 rpm.
3. Follow the real axis.
 Enable "ex_FlexProfile_Drive1" (ML_FlexProfile) by "Write values".
 The expected coupling behavior is:

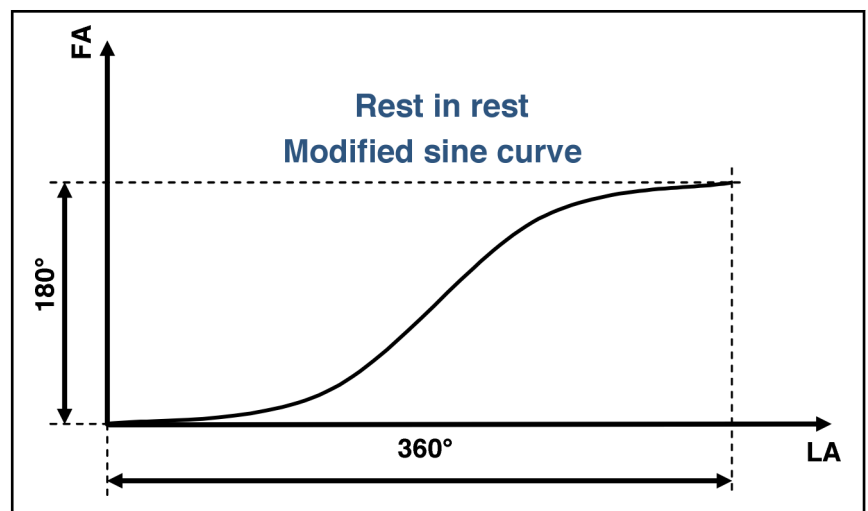


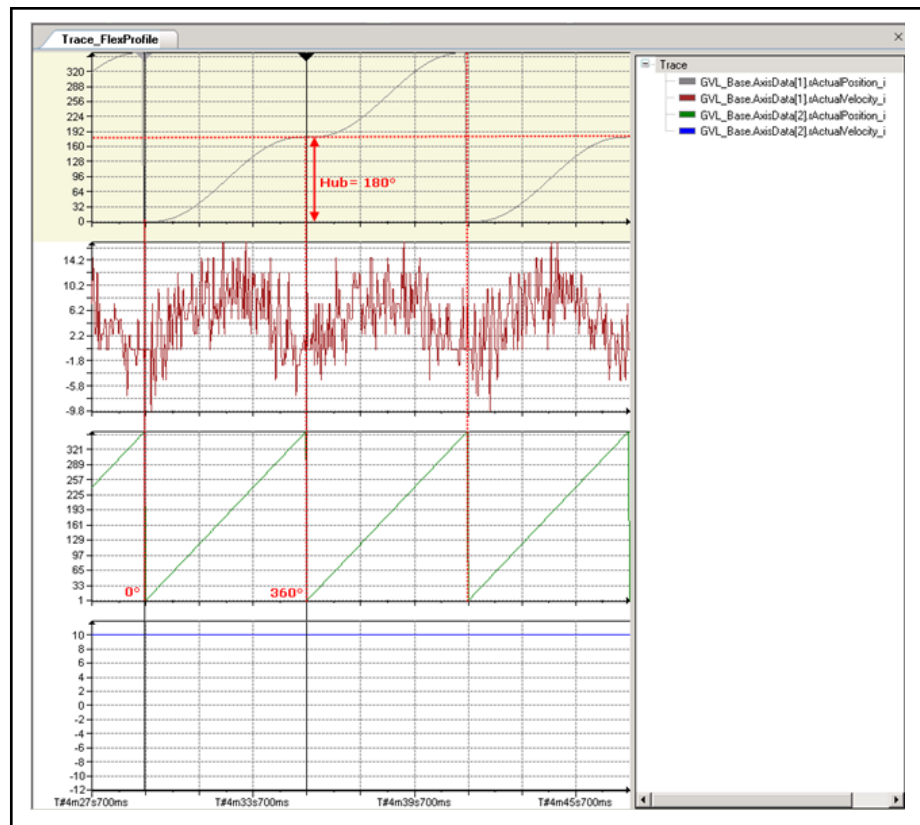
Fig. 4-17: Coupling profile between master axis (vAxis1) and slave axis (Drive1)

4. Disable "ex_FlexProfileDrive1" (ML_FlexProfile) by "Write Values".
is_FlexProfileDrive1= FALSE, ac_FlexProfileDrive1= TRUE.
The real axis continues to follow the virtual axis.
5. Terminate synchronicity.
Enable the "ex_GearOut_Drive1" (MC_GearOut) by "Write values".
The dependency of the real axis on the virtual axis is suspended. However, both axes continue to rotate with the same velocity.
6. Change the command velocity "Vcom_vAxis1".
The changed command velocity only affects the virtual axis.
"Vact_Drive1" remains unchanged.
7. Disable "ex_GearOut_Drive1" by "Write values" TRUE/FALSE.
do_GearOut_Drive1= FALSE
The real axis stops. The axis is actively held in this position after reaching the standstill.
8. Stop the virtual axis.
Enable "ex_Stop_vAxis1" (MC_Stop)
The virtual axis is decelerated to Vact_vAxis1 = 0 rpm.
9. Power off for the real axis.
Disable "en_Power_Drive1" (MC_Power) by "Write values".

4.8 Recording the motion of both axes (trace)

The motions of master axis and slave axis can be displayed in the trace recording. The following characteristic results:

First steps - FlexProfile



Grey Slave axis (Drive1), position
Red Slave axis (Drive1), velocity
Green Master axis (vAxis1), position
Blue Master axis (vAxis1), velocity

Fig. 4-18: Recording the motion of both axes

The trace recording below shows the velocity characteristics of the master axis.

The master axis rotates at a constant velocity of 10 [rpm].

The position of the master axis is located above.

The uppermost signal shows the position of the slave axis.

It can be seen that the parameterized previously FlexProfile is executed correctly.

4.9 Restoring the project

An export file was generated from the created project allowing to restore the project irrespective of the control hardware (IndraMotion MLC L25, L45, L65 L75, XM2, VPx). The restored project can run as simulation or by using your control.

The export file was created for the "Application" node and includes the following components:

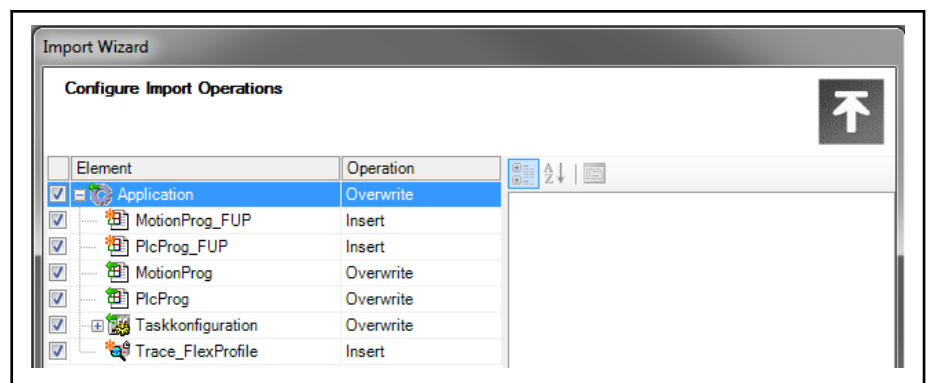


Fig. 4-19: Components of the export file

The specified components are overwritten during import.

The export file "Application_FirstSteps_IndraLogic_FlexProfile.iwx" is located in the directory "<Drive>:\All Programs\Rexroth\IndraWorks\Projects".

5 First steps - Robot Control V2

5.1 Project goal "First steps - Robot Control V2"

A simple example explains the general process of a Robot Control V2 project creation using IndraMotion MLC. Starting with the creation of an example project up to the execution of the project, information is provided on the minimum required configuration in IndraWorks and the programming in the PLC.

In the example project "RoCoTestV14", it is assumed that only virtual axes are available. At the end of the description, information on how to integrate real axes is provided.



The user can either load the example project from the IndraWorks installation DVD from the directory "\\AddOns\\Mlc\\Example-Project\\RobotControl"; the example project that contains a .zip file with the following content:

- the parameter export file (*.par)
- the PLC program
- the visualization data

- or -

the user can create the project step by step using the following description. The detailed data provided in the appendix is to be used.

5.2 Required hardware and software

Required hardware *Minimum:*

- Control, type "IndraMotion MLC Lx5", "IndraMotion MLC XM2" or "IndraMotion MLC VPx"
- Current firmware is loaded
- IP address is known
- Programming device (PC / laptop)
- Control is connected to the programming device via Ethernet

Optional:

- Three real axes (e.g. IndraDrive and EcoDriveCS)
- Current firmware (from MPx07) is loaded

Required software IndraWorks installation 14VRS using IndraMotion MLC as target system.

5.3 Procedure when creating an RoCo project

Execute the following steps for the step-by-step creation of the example project:

General approach

1. Start IndraWorks and create a new project.

After starting IndraWorks, create the new project via **File ► New ► Project.....**

First steps - Robot Control V2

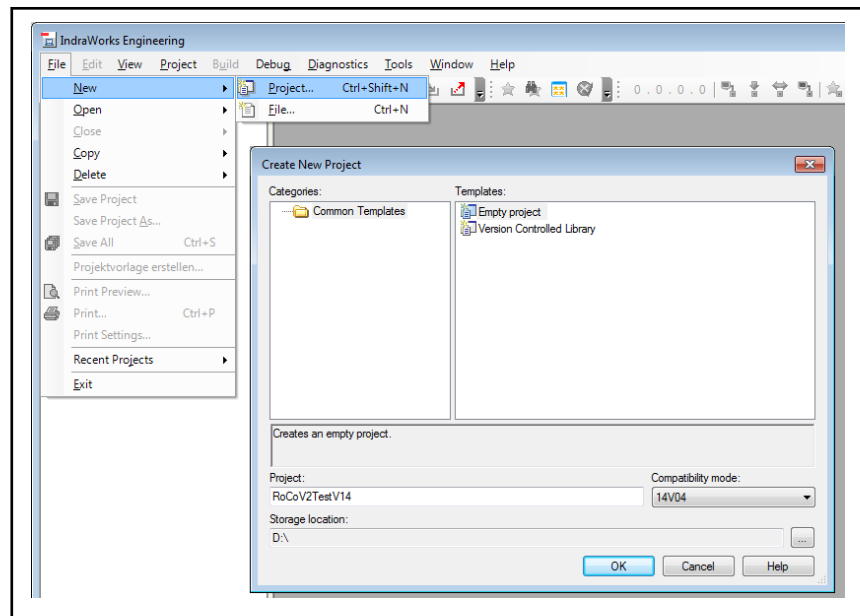


Fig. 5-1: Creating a new IndraWorks project

Enter any project name (e.g. "RocoV2TestV14").

Select a storage location.

When selecting the project language, the language of the user texts is determined.

Press **OK**.

The newly created project is displayed in the Project Explorer with the previously defined project name.

2. Create and configure the IndraMotion MLC control.

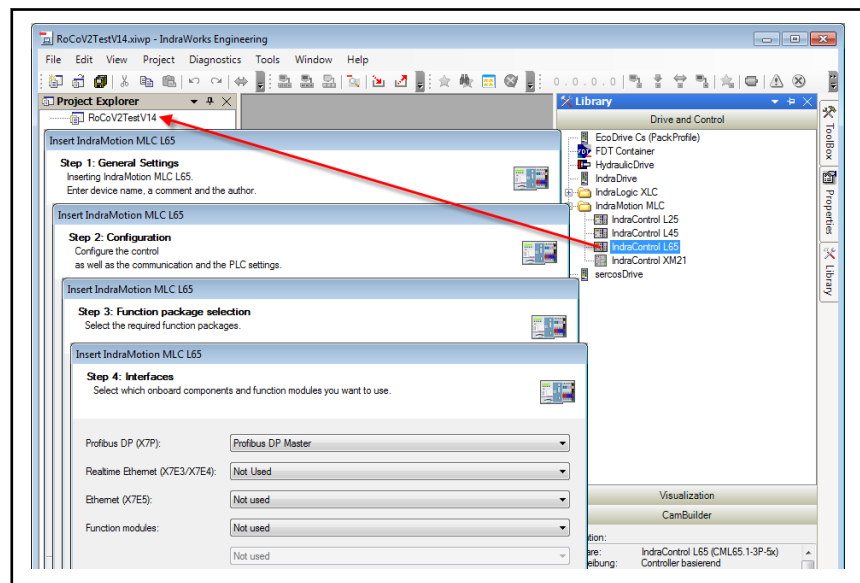


Fig. 5-2: Creating and configuring an IndraMotion MLC

Drag the device (e.g. "IndraMotion MLC L65") from the "Drive and Control" library and drop it onto the newly created project in the Project Explorer.

First steps - Robot Control V2

Subsequently, the wizard to specify the basic settings is automatically started.

Enter any device name (e.g. "mlc_L65") in the "General settings" dialog and press **Next**.

In the "Configuration" dialog, define the settings for the firmware and communication between IndraWorks and the device.

The presettings for the "device type", the "firmware version" and the "firmware release" do usually not have to be changed.

Enter the IP address of the control, to which the application is to be loaded, under "IP address".

Test the connection and check whether the communication between PC and control is established successfully.

Press **Next**.

If no function modules are connected to the control, no other settings have to be selected in the "Interfaces" dialog.

Press **Finish**.

3. Create and configure three virtual axes.

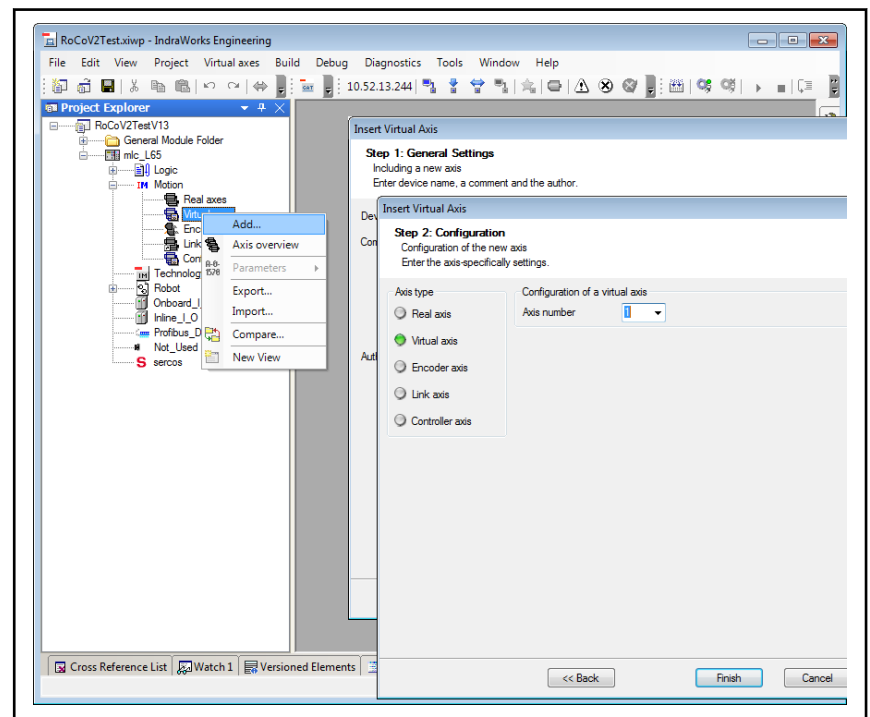


Fig. 5-3: Creating virtual axes

Right-click on "Virtual axes" and select "Add" from the context menu.

Enter the name "Axis1" in the "General settings" dialog under "Device name" and press **Next**.

Enter axis number "1" under the "Configuration" dialog and press **Finish**.

First steps - Robot Control V2

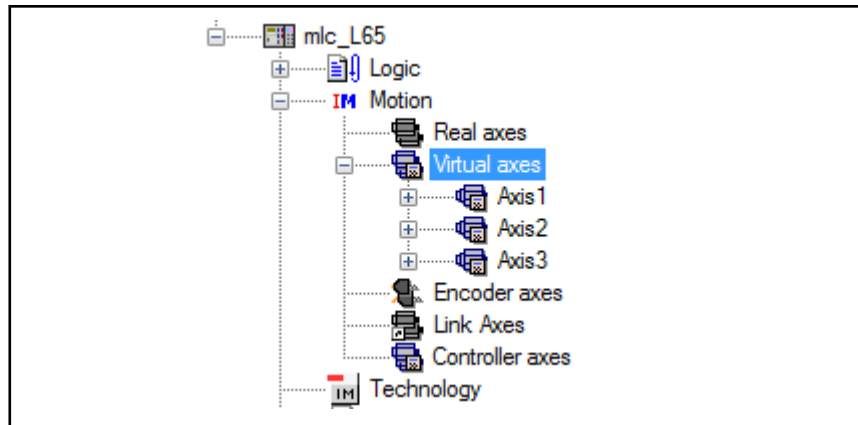


Fig. 5-4: Virtual axis created

Perform this analogously to create two more virtual axes, so that three virtual axes are displayed at the "Virtual axes" node.

4. Add and configure a kinematics.

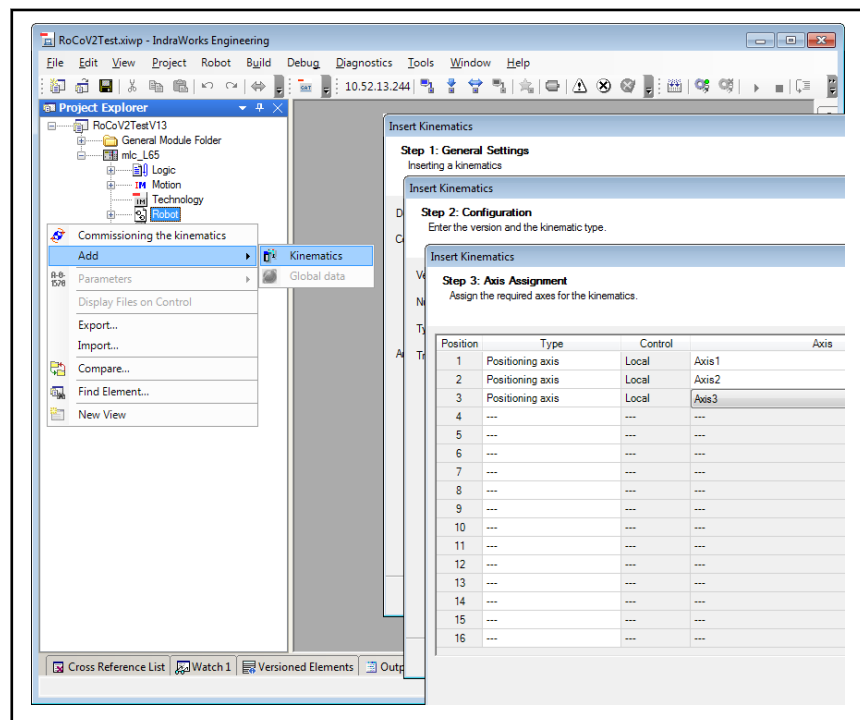


Fig. 5-5: Adding and configuring kinematics

Right-click on the "Robot" node and select "Add" and "Kinematics" from the context menu.

Enter any "device name" (e.g. "Kin01") in the "General settings" dialog and press **Next**.

Select "Cartesian" under the kinematics type in the dialog "Configuration" and press **Next**.

Define three positioning axes in the "Axis assignment" dialog and assign the virtual axes.

To close the procedure, press **OK**.

First steps - Robot Control V2

- Download the Motion configuration to the device.

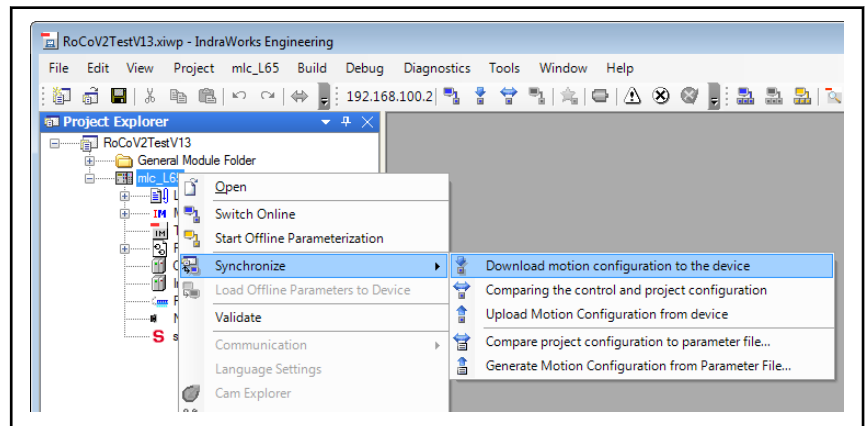


Fig. 5-6: Download the Motion configuration to the device

Right-click on the "Control" node and select "Synchronize" and "Download motion configuration to the device" from the context menu. The settings made up to now are applied to the control.

- Import parameters.

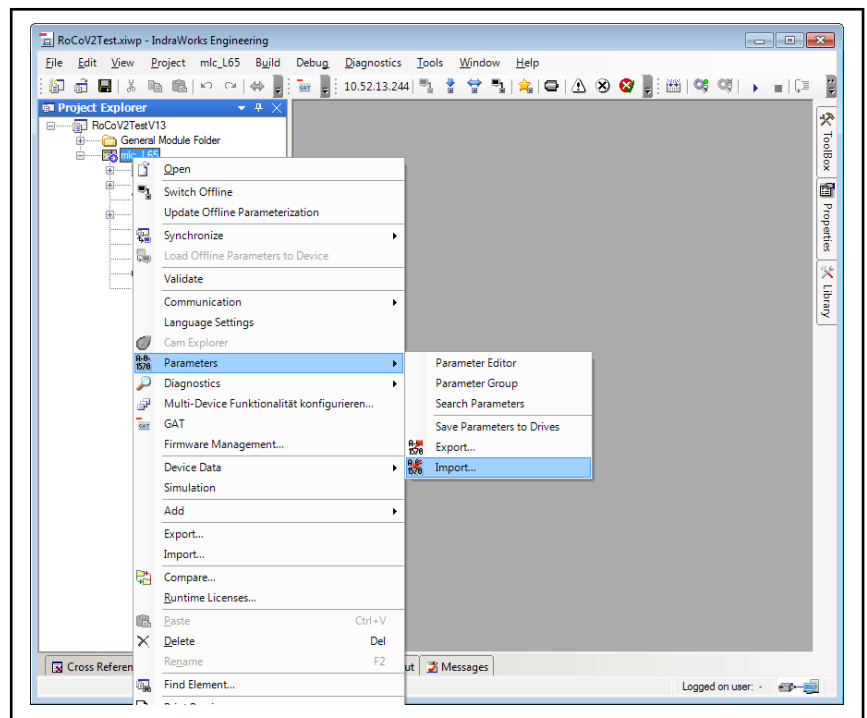


Fig. 5-7: Importing parameters

Right-click on the "Control" node and select "Parameters" and "Import" from the context menu.



For the parameter file, go to the project directory on the installation DVD: "AddOns\MLC\Example-Project\RobotControl\RoCoV2TestV14\RoCoV2TestV14virtuell.par".

First steps - Robot Control V2

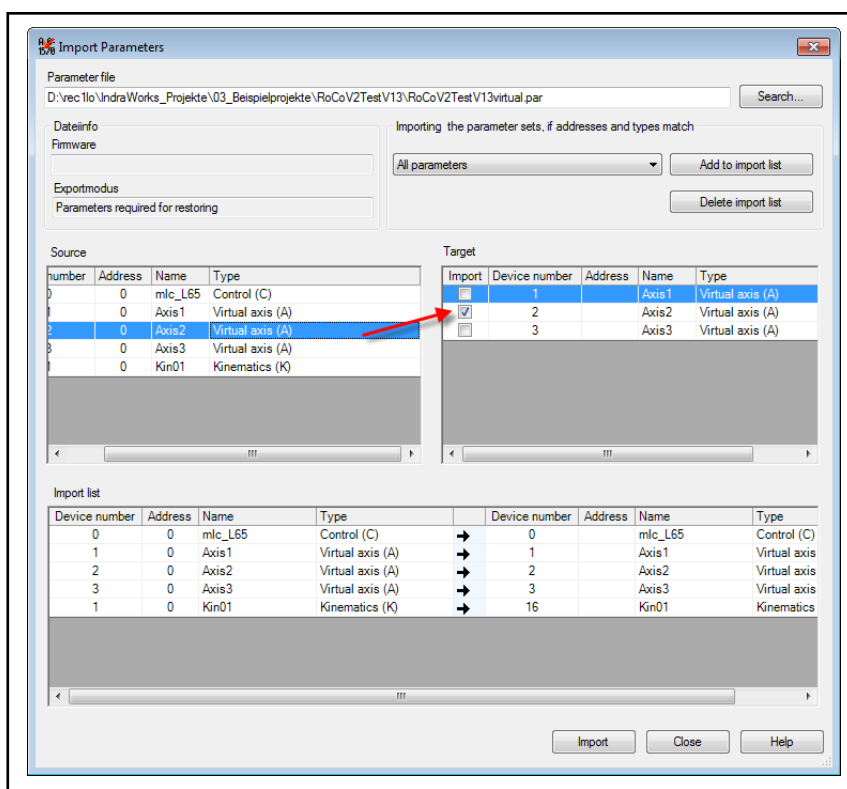


Fig. 5-8: Selecting parameters

The import wizard opens automatically. Assign one target to each source. Thus, control, axis and kinematic parameters can be imported. To confirm the entry, press **Import**.

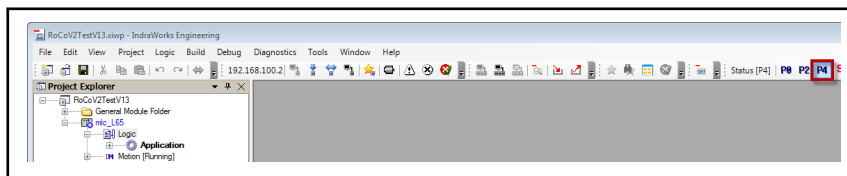


Fig. 5-9: Operational state

After a successful parameter import, it should be possible to switch the control to the operational state BB (P4).

7. Create a PLC program.

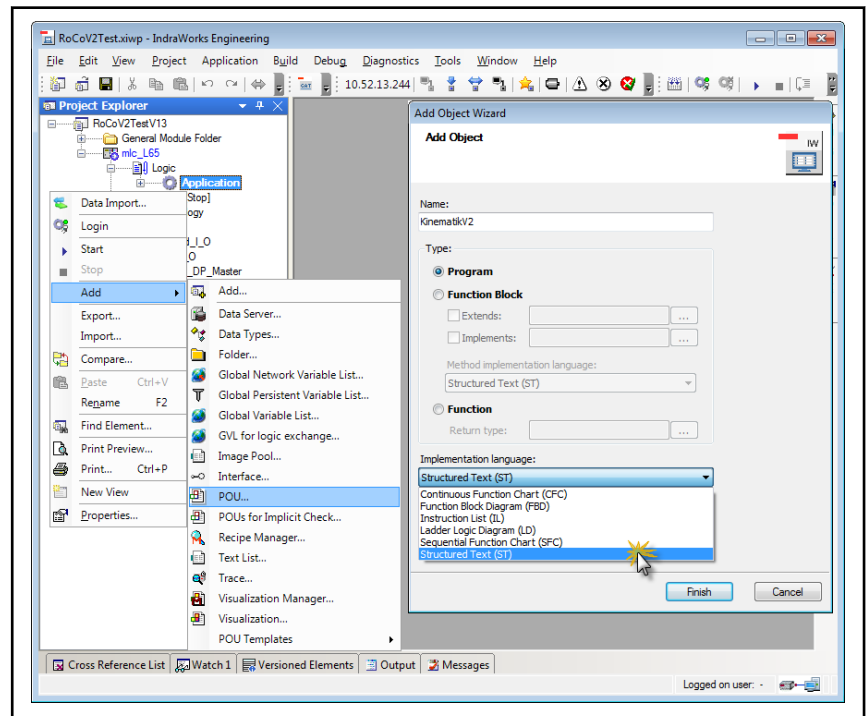


Fig. 5-10: Creating PLC program

Right-click on the "Application" node and select "Add" and then "POU" from the context menu.

Specify any program name (e.g. "KinematikV2") in the "Add object" dialog. Select "Program" as object type and "Structured Text (ST)" as implementation language.

Add the program stored with the declaration section to the "Details on the project RoCoV2TestV14 - Listing of the PLC program (KinematikV2)".

Also add the program with the declaration section stored in the "Details on the project RoCoV2TestV14 - Listing of the PLC program (PlcProg)" to the program "PlcProg".

All programs have to be compiled without errors here via .

8. Add the program call to the MotionTask.

To go to the task configuration, double-click on the MotionTask.

First steps - Robot Control V2

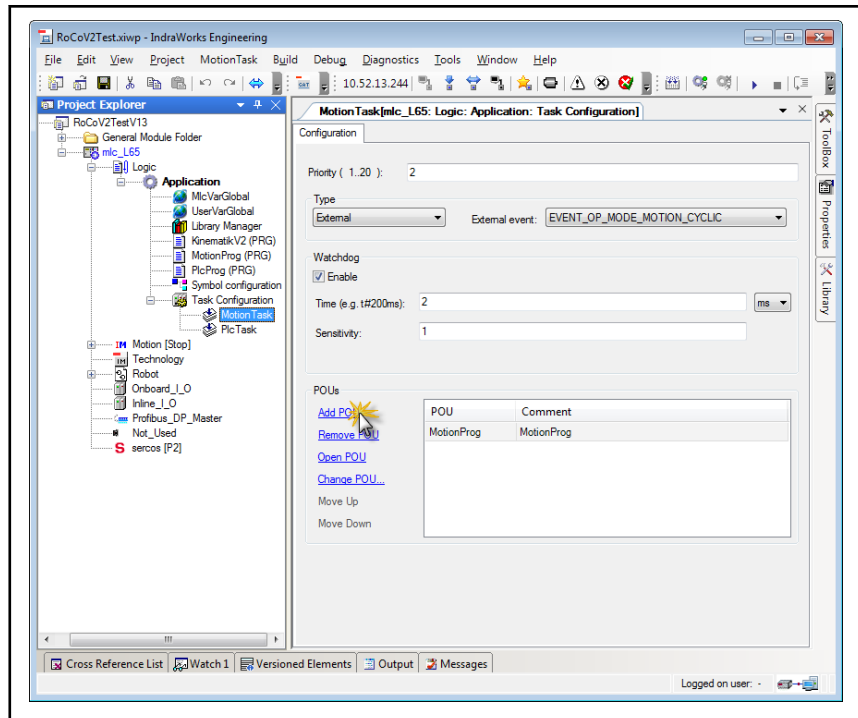


Fig. 5-11: Configuring task

Add the PLC program call. Select the previously created PLC program via "Add call" and press **OK**.

9. Implement visualization.

To operate the PLC program, go to a visualization in the project directory on the installation DVD ("AddOns\MLC\Example-Project\RobotControl\RoCoV2TestV14\Visualization.iwx"). This can be used for all required operating actions.

First steps - Robot Control V2

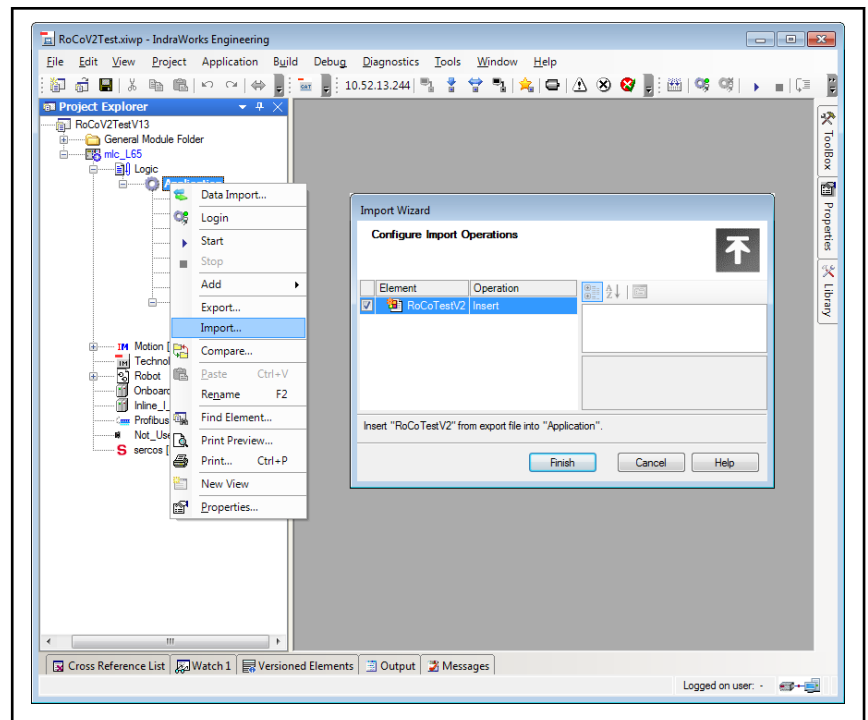


Fig. 5-12: Importing visualization

Right-click on "Application" and select "Import" from the context menu. Select the "Visualization" file from the project directory and press **Open**. To confirm the import, press **Finish**.

10. Download the PLC program.
To download the PLC program, log into the control.

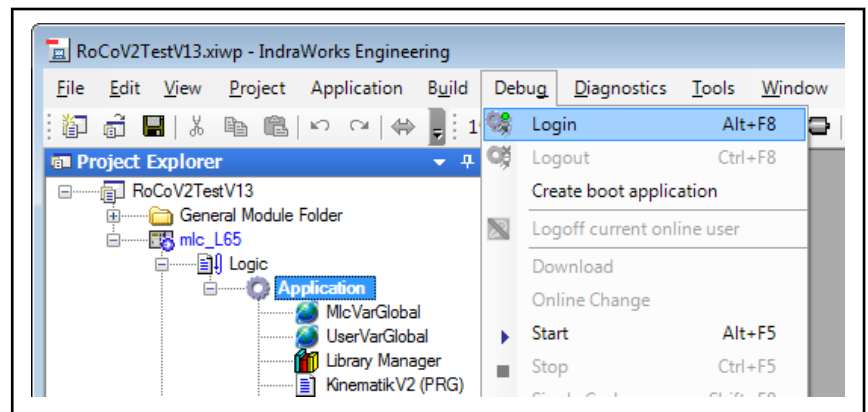


Fig. 5-13: Login

Therefore, select **Debug** ► **Login**. Then, start the control.

First steps - Robot Control V2

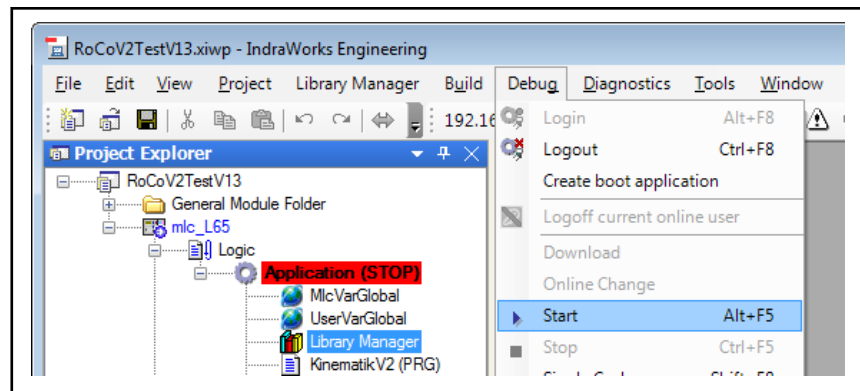


Fig. 5-14: Starting application

Therefore, select **Debug ► Start**.

11. Start the motion via visualization.

Use the imported visualization to prepare, start and stop the motion sequence.

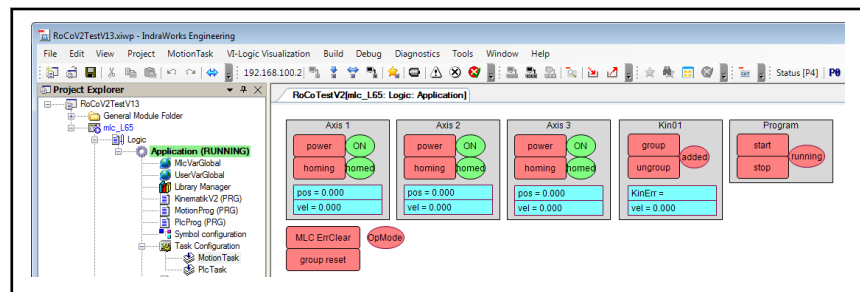


Fig. 5-15: Starting motion via visualization

To group the axes, press **group**. The field "added" becomes green, once the axes are grouped successfully. All axes belonging to the defined kinematics are grouped in this step.

To start the program, press **Start**. The program was successfully started when the "running" field becomes green.

The program can be stopped (**Stop** button) and the kinematics can be ungrouped (**Ungroup**) analogously.

In case of error, the kinematics can be reset via the **Group reset** button and all control and drive errors can be deleted via **MLC ErrClear** (C-0-1030).

Integrating real axes

Real axes can also be used instead of virtual axes. Connect three real drives to the control via the Sercos bus and perform the following steps:

First steps - Robot Control V2

1. Delete the virtual axes.

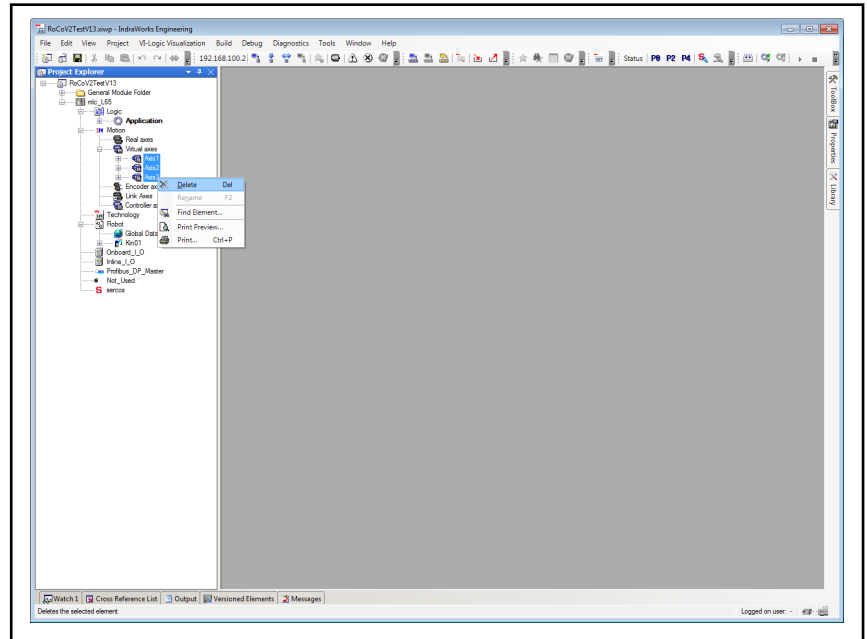


Fig. 5-16: Deleting virtual axes

Select all three virtual axes. Right-click on the highlighted axes and select "Delete" from the context menu.

2. Create and configure three real drives.

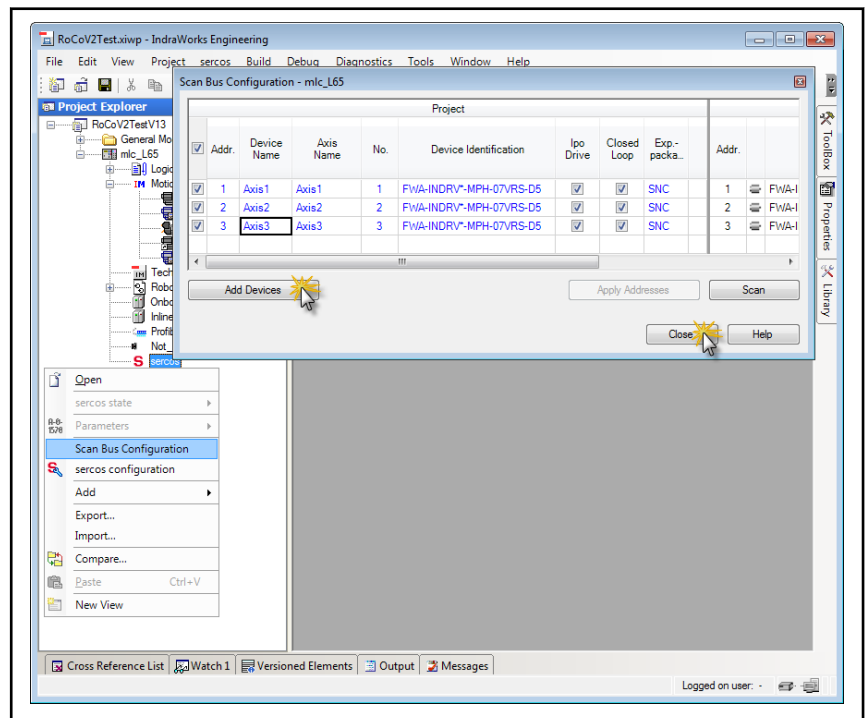


Fig. 5-17: Creating real axes

Right-click on the "Sercos" node and select "Scan bus configuration" from the context menu.

First steps - Robot Control V2

As device names, select the names of the deleted virtual axes ("Axis1", "Axis2", "Axis3"). This ensures that the PLC program can run without any adaptations.

Press **Add devices** to add the axes and press **Close** to close the window.

3. Adjust the kinematic configuration.

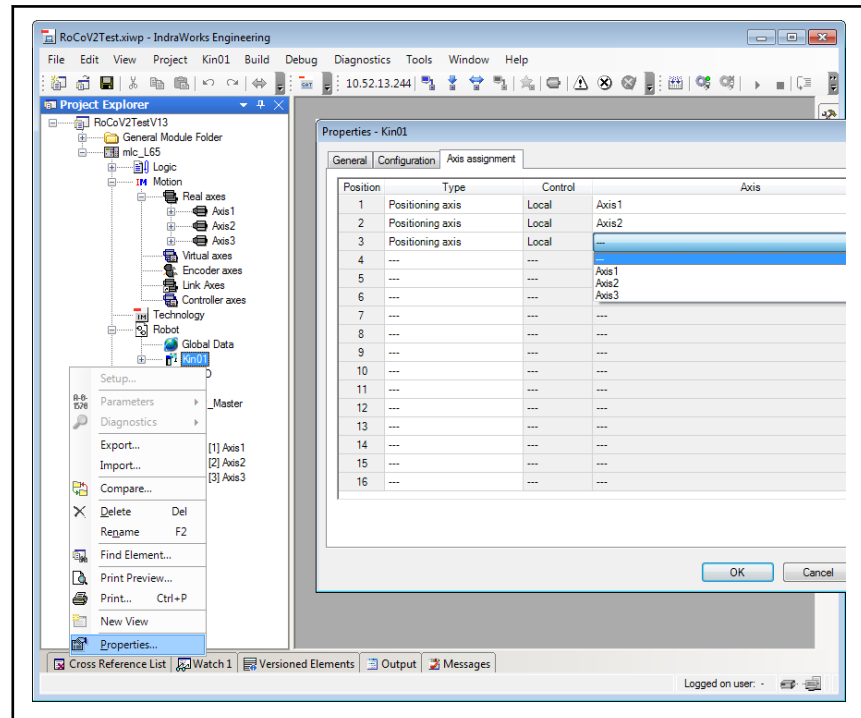


Fig. 5-18: Adjusting the kinematic configuration

Open the "Properties" dialog of the kinematics and adjust the axis assignments to the new real axes.

To confirm your entries, press **OK**.

4. Download Motion configuration and import parameters.

Download the Motion configuration to the control and import the parameters for the real axes. Control and kinematic parameters do not have to be imported again!



For the parameter file for real axes, go to the project directory on the installation DVD: "AddOns\MLC\Example-Project\RobotControl\RoCoV2TestV14\RoCoV2TestV14real.par".

Now, commission the kinematics with the already known kinematics. Observe that in contrast to the virtual axes, power has to be added for the real axes (**Power** button) and - if no absolute value encoders are used - homing has to be executed (**Homing** button). Grouping is only possible afterwards.

5.4 Details on the "RoCoV2TestV14" project

Listing of the PLC program (kinematicsV2), declaration

```
PROGRAM KinematikV2
VAR
  (* variables *)
  bPower      : ARRAY[1..3] OF BOOL := [3(FALSE)];
  bHome       : ARRAY[1..3] OF BOOL := [3(FALSE)];
  bGroup      : BOOL := FALSE;
```

First steps - Robot Control V2

```
bUngroup      : BOOL := FALSE;
bGroupReset   : BOOL := FALSE;
bStart        : BOOL := FALSE;
bStop         : BOOL := FALSE;
bRunning      : BOOL := FALSE;
bMoveDirectAbsoluteP0 : BOOL := FALSE;
bMoveDirectAbsoluteP1 : BOOL := FALSE;
bMoveDirectAbsoluteP2 : BOOL := FALSE;
bMoveDirectAbsoluteP3 : BOOL := FALSE;
bMoveDirectAbsoluteP4 : BOOL := FALSE;
bMoveDirectAbsoluteP5 : BOOL := FALSE;
bMoveDirectAbsoluteP6 : BOOL := FALSE;
rVFac: REAL := 0.0;
rAFac: REAL := 0.0;
rDFac: REAL := 0.0;
rJFac: REAL := 0.0;
arPositionP0: ARRAY[1..16] OF REAL := [0.0,0.0,0.0,13(0)];
arPositionP1: ARRAY[1..16] OF REAL := [100.0,0.0,0.0,13(0)];
arPositionP2: ARRAY[1..16] OF REAL := [100.0,100.0,0.0,13(0)];
arPositionP3: ARRAY[1..16] OF REAL := [100.0,100.0,100.0,13(0)];
arPositionP4: ARRAY[1..16] OF REAL := [400.0,400.0,400.0,13(0)];
arPositionP5: ARRAY[1..16] OF REAL := [300.0,200.0,100.0,13(0)];
arPositionP6: ARRAY[1..16] OF REAL := [111.0,222.0,333.0,13(0)];
uiState       : UINT := 0;
uiCounter     : UINT := 0;
tGroupWait    : TIME := T#0S;

(* instances *)
fbPower       : ARRAY[1..3] OF MC_Power;
fbHome        : ARRAY[1..3] OF MB_Home;
fbGroupAllAxes : ML_GroupAllAxes;
fbUngroupAllAxes: MC_UngroupAllAxes;
fbGroupReset  : MC_GroupReset;
fbMoveDirectAbsoluteP0: MC_MoveDirectAbsolute;
fbMoveDirectAbsoluteP1: MC_MoveDirectAbsolute;
fbMoveDirectAbsoluteP2: MC_MoveDirectAbsolute;
fbMoveDirectAbsoluteP3: MC_MoveDirectAbsolute;
fbMoveDirectAbsoluteP4: MC_MoveDirectAbsolute;
fbMoveDirectAbsoluteP5: MC_MoveDirectAbsolute;
fbMoveDirectAbsoluteP6: MC_MoveDirectAbsolute;
fbGroupWaitP0: ML_GroupWait;
fbGroupWaitP6: ML_GroupWait;
fbGroupStop   : MC_GroupStop;
END_VAR

VAR CONSTANT
    STATE_INIT       : UINT := 00;
    STATE_MOVE_ZERO  : UINT := 10;
    STATE_MOVEMENT   : UINT := 20;
    STATE_FINISH     : UINT := 30;
END_VAR
```

Listing of the PLC program (kinematicsV2), implementation

```
(* example program for kinematics *)

(* set power for all real kinematic axes *)
fbPower[1]( Enable := bPower[1], Axis := Axis1 );
fbPower[2]( Enable := bPower[2], Axis := Axis2 );
fbPower[3]( Enable := bPower[3], Axis := Axis3 );

(* homing for all real kinematic axes *)
fbHome[1]( Execute := bHome[1], Axis := Axis1 );
fbHome[2]( Execute := bHome[2], Axis := Axis2 );
fbHome[3]( Execute := bHome[3], Axis := Axis3 );

(* group all axes *)
fbGroupAllAxes( Execute := bGroup, AxisGroup := Kin01 );

(* ungroup all axes *)
fbUngroupAllAxes( Execute := bUngroup, AxisGroup := Kin01 );

(* reset group *)
fbGroupReset( Execute := bGroupReset, AxisGroup:= Kin01 );

(* state machine *)
CASE uiState OF
    STATE_INIT: // initialization
        rVFac := 40; // velocity [%]
        rAFac := 100; // acceleration [%]
        rDFac := 100; // deceleration [%]
        rJFac := 100; // jerk [%]
```

First steps - Robot Control V2

```

tGroupWait := T#3S;
uiCounter  := 0;
bRunning   := FALSE;

IF bStart THEN
    uiState := STATE_MOVE_ZERO;
    bStart  := FALSE;
    bRunning := TRUE;
END_IF;

STATE_MOVE_ZERO: // move to zero position
    bMoveDirectAbsoluteP0 := TRUE;

    (* movement successfully - go to next state *)
    IF fbGroupWaitP0.Done THEN
        bMoveDirectAbsoluteP0 := FALSE;
        uiState                := STATE_MOVEMENT;
    END_IF;

    (* movement stopped or finished with error - go to initialization *)
    IF fbMoveDirectAbsoluteP0.Error OR fbGroupWaitP0.Error OR bStop THEN
        bMoveDirectAbsoluteP0 := FALSE;
        uiState                := STATE_INIT;
    END_IF;

STATE_MOVEMENT: // move to several positions
    bMoveDirectAbsoluteP1 := TRUE;
    bMoveDirectAbsoluteP2 := TRUE;
    bMoveDirectAbsoluteP3 := TRUE;
    bMoveDirectAbsoluteP4 := TRUE;
    bMoveDirectAbsoluteP5 := TRUE;
    bMoveDirectAbsoluteP6 := TRUE;

    (* movements successfully - go to next state *)
    IF fbGroupWaitP6.Done THEN
        bMoveDirectAbsoluteP1 := FALSE;
        bMoveDirectAbsoluteP2 := FALSE;
        bMoveDirectAbsoluteP3 := FALSE;
        bMoveDirectAbsoluteP4 := FALSE;
        bMoveDirectAbsoluteP5 := FALSE;
        bMoveDirectAbsoluteP6 := FALSE;
        uiCounter              := uiCounter + 1;
        uiState                := STATE_FINISH;
    END_IF;

    (* movement stopped or finished with error - go to initialization *)
    IF fbGroupWaitP6.Error OR
        fbMoveDirectAbsoluteP1.Error OR
        fbMoveDirectAbsoluteP2.Error OR
        fbMoveDirectAbsoluteP3.Error OR
        fbMoveDirectAbsoluteP4.Error OR
        fbMoveDirectAbsoluteP5.Error OR
        fbMoveDirectAbsoluteP6.Error OR
        bStop THEN
        bMoveDirectAbsoluteP1 := FALSE;
        bMoveDirectAbsoluteP2 := FALSE;
        bMoveDirectAbsoluteP3 := FALSE;
        bMoveDirectAbsoluteP4 := FALSE;
        bMoveDirectAbsoluteP5 := FALSE;
        bMoveDirectAbsoluteP6 := FALSE;
        uiState                := STATE_INIT;
    END_IF;

STATE_FINISH: // movement finished
    IF uiCounter > 3 THEN
        uiState := STATE_INIT;
    ELSE
        uiState := STATE_MOVEMENT;
    END_IF;

END_CASE;

(* function block: move to PO *)
fbMoveDirectAbsoluteP0(
    Execute      := bMoveDirectAbsoluteP0,
    Position     := arPositionP0,
    Velocity     := rVFac,
    Acceleration := rAFac,
    Deceleration := rDFac,
    Jerk         := rJFac,
    CoordSystem  := CS_MACHINE_COORDINATES,
    SlopeType    := ST_BLOCK_SLOPE,
    SetMode      := SM_TO,

```

First steps - Robot Control V2

```
        BlendingRadius := 0.0,  
        AxisGroup      := Kin01);  
  
(* function block: wait 3 seconds *)  
fbGroupWaitP0(  
    Execute      := fbMoveDirectAbsoluteP0.Done,  
    WaitTime     := tGroupWait,  
    AxisGroup    := Kin01);  
  
(* function block: move to P1 *)  
fbMoveDirectAbsoluteP1(  
    Execute      := bMoveDirectAbsoluteP1,  
    Position     := arPositionP1,  
    Velocity     := rVFac,  
    Acceleration := rAFac,  
    Deceleration := rDFac,  
    Jerk         := rJFac,  
    CoordSystem  := CS_MACHINE_COORDINATES,  
    SlopeType    := ST_BLOCK_SLOPE,  
    SetMode      := SM_TO,  
    BlendingRadius := 0.0,  
    AxisGroup    := Kin01);  
  
(* function block: move to P2 *)  
fbMoveDirectAbsoluteP2(  
    Execute      := bMoveDirectAbsoluteP2,  
    Position     := arPositionP2,  
    Velocity     := rVFac,  
    Acceleration := rAFac,  
    Deceleration := rDFac,  
    Jerk         := rJFac,  
    CoordSystem  := CS_MACHINE_COORDINATES,  
    SlopeType    := ST_BLOCK_SLOPE,  
    SetMode      := SM_TO,  
    BlendingRadius := 0.0,  
    AxisGroup    := Kin01);  
  
(* function block: move to P3 *)  
fbMoveDirectAbsoluteP3(  
    Execute      := bMoveDirectAbsoluteP3,  
    Position     := arPositionP3,  
    Velocity     := rVFac,  
    Acceleration := rAFac,  
    Deceleration := rDFac,  
    Jerk         := rJFac,  
    CoordSystem  := CS_MACHINE_COORDINATES,  
    SlopeType    := ST_BLOCK_SLOPE,  
    SetMode      := SM_TO,  
    BlendingRadius := 0.0,  
    AxisGroup    := Kin01);  
  
(* function block: move via P4 *)  
fbMoveDirectAbsoluteP4(  
    Execute      := bMoveDirectAbsoluteP4,  
    Position     := arPositionP4,  
    Velocity     := rVFac,  
    Acceleration := rAFac,  
    Deceleration := rDFac,  
    Jerk         := rJFac,  
    CoordSystem  := CS_MACHINE_COORDINATES,  
    SlopeType    := ST_BLOCK_SLOPE,  
    SetMode      := SM_VIA,  
    BlendingRadius := 30,  
    AxisGroup    := Kin01);  
  
(* function block: move via P5 *)  
fbMoveDirectAbsoluteP5(  
    Execute      := bMoveDirectAbsoluteP5,  
    Position     := arPositionP5,  
    Velocity     := rVFac,  
    Acceleration := rAFac,  
    Deceleration := rDFac,  
    Jerk         := rJFac,  
    CoordSystem  := CS_MACHINE_COORDINATES,  
    SlopeType    := ST_BLOCK_SLOPE,  
    SetMode      := SM_VIA,  
    BlendingRadius := 30,  
    AxisGroup    := Kin01);  
  
(* function block: move to P6 *)  
fbMoveDirectAbsoluteP6(  
    Execute      := bMoveDirectAbsoluteP6,  
    Position     := arPositionP6,
```

First steps - Robot Control V2

```

Velocity      := rVFac,
Acceleration  := rAFac,
Deceleration  := rDFac,
Jerk          := rJFac,
CoordSystem   := CS_MACHINE_COORDINATES,
SlopeType     := ST_BLOCK_SLOPE,
SetMode       := SM_TO,
BlendingRadius := 0.0,
AxisGroup     := Kin01);

(* function block: wait 3 seconds *)
fbGroupWaitP6(
    Execute      := fbMoveDirectAbsoluteP6.Done,
    WaitTime     := tGroupWait,
    AxisGroup    := Kin01);

(* function block: stop movement *)
fbGroupStop(
    Execute      := bStop,
    Deceleration := rDFac,
    Jerk         := rJFac,
    AxisGroup    := Kin01);

```

Listing of the PLC program (PlcPrg), declaration

```

PROGRAM PlcProg
VAR
    fbReset      : MB_WriteParameter;
    bErrClear    : BOOL;
END_VAR

```

Listing of the PLC program (PlcPrg), implementation

```

(* error clear MLC *)
fbReset(
    Execute      := bErrClear,
    ParameterNumber := FP_C_0_1030,
    Value        := 3,
    Axis         := Axis1);

```

6 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 Hotline** and **Service Helpdesk** 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
- Type plate specifications of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your e-mail address)

Index

A

Abbreviations.....	14
About this documentation.....	5
Validity of the documentation.....	5
ANSI Z535.6-2006.....	6

C

Complaints.....	14
Criticism.....	14
Customer Feedback.....	14

D

Documentation structure.....	6
------------------------------	---

E

Export file	
First steps - IndraLogic.....	74
First steps - Motion.....	138
First steps - FlexProfile.....	156

F

Feedback.....	14
First steps - FlexProfile.....	141
Create Motion subproject.....	78
First steps - IndraLogic.....	15
Add and edit element.....	69
Add function block.....	29
Add new device to project.....	17
Add POU.....	25, 30
Add program.....	28
Basic settings for the device	18
Communication settings.....	18
Control of a traffic light system.....	25
Copy visu-element.....	66
Create new project.....	16
Create new visualization.....	62
Create visualization for PLC program.....	61
Declare global variables.....	49
Device settings.....	18
Edit FBD.....	30
Edit PlcProg in CFC.....	55
Edit PlcProg in ST.....	52
Edit programs and function blocks - Part 2... ..	52
Edit SFC.....	40
Edit ST.....	38
Edit visualization.....	62
Function module settings.....	23
New IndraWorks project	16
Programs and function blocks.....	25
Restore.....	74
Restore project.....	74
Start IndraWorks.....	16
Task configuration.....	48
Technical requirements.....	15
Test in simulation mode.....	50

Test visualization.....	73
Visu-element "Rectangle".....	67
Visu-element button.....	70
Visu-element display of variable values.....	72
Visu-element ellipse.....	62
Visu-element labeling.....	71
Visualization in IndraLogic.....	61
First Steps - IndraLogic	
Create PLC program.....	25
First steps - Motion.....	77
Add new device to project.....	80
Add virtual axis.....	87
Basic settings for the device	80
Communication settings.....	80
Create new project	78
Device settings.....	80
Download Motion subproject.....	88
Download PLC project vAxis1 (FBD).....	99
Download PLC project vAxis1 (ST).....	122
Download PLC projects Drive1 (FBD).....	111
Download PLC projects Drive1 (ST).....	136
Function module settings.....	85
Function package selection.....	22, 84
PLC characteristics.....	21, 82
Program motion sequence (FBD).....	91
Program motion sequence (ST).....	113
Program real axis Drive1 (FBD).....	107
Program real axis Drive1 (ST).....	131
Program virtual axis vAxis1 (FBD).....	92
Program virtual axis vAxis1 (ST).....	115
Restore project.....	138
Scan Sercos III drives.....	86
Start IndraWorks.....	78
Technical requirements.....	77
Trace_vAxis1 (FBD).....	102
Trace_vAxis1 (ST).....	125
Trace_vAxis1_Drive1 (FBD).....	112
Trace_vAxis1_Drive1 (ST).....	137
Workflow virtual axis → Electronic gear →	
Real axis.....	111, 136
Write values, input variables (FBD).....	100
Write values, input variables (ST).....	124
First steps - Robot Control V2.....	159
Create RoCo project.....	159
Required hardware and software.....	159
First steps - FlexProfile	
Create FlexProfile.....	144
Create FlexProfile via IndraWorks dialog....	144
Create FlexProfile via PLC.....	146
Create Motion subproject.....	142
Download PLC subproject FlexProfile.....	153
Execute motion step, ML_FlexProfile.....	150
Restore project.....	156
Technical requirements.....	142
Trace_FlexProfile.....	155

Index

Workflow virtual axis → Virtual axis →
FlexProfile 0 → Real axis..... 154

First steps – Robot Control V2
RoCoTestV2..... 170

H

Hazard warning..... 6

Helpdesk..... 175

Hotline..... 175

N

Names..... 14

Names and abbreviations..... 14

S

Safety alert symbol..... 6

Safety instructions..... 6

Service hotline..... 175

Signal words..... 6

Suggestions..... 14

Support..... 175

Symbols used..... 8

W

Warnings..... 6

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



R911341517