

Rexroth IndraMotion MLC 13VRS First Steps

R911336293
Edition 02

Commissioning Manual



Title	Rexroth IndraMotion MLC 13VRS First Steps
Type of Documentation	Commissioning Manual
Document Typecode	DOK-MLC***-F*STEP**V13-CO02-EN-P
Internal File Reference	RS-b66169185d83bfec0a6846a5014fee5a-2-en-US-3
Purpose of Documentation	This documentation contains the following descriptions for IndraMotion MLC 13VRS: • First steps - IndraLogic basics of programming 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 V1, Creation of a Robot Control project until the execution of the RoCo source code • First steps - Robot Control V2 Creation of a Robot Control project until the execution of the project The documentation is the basis for the online help in the system.

Record of Revision	Edition	Release Date	Notes
	Edition 01	02.2013	First edition
	Edition 02	10.2013	Project sync, Robot Control V2

Copyright © Bosch Rexroth AG 2013

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.

Published by Bosch Rexroth AG

Bgm.-Dr.-Nebel-Str. 2 ■ 97816 Lohr a. Main, Germany

Phone +49 9352 18 0 ■ Fax +49 9352 18 8400

<http://www.boschrexroth.com/>

System Development Automation Motion Logic Control HBU (vha/PiGe)

Note This document has been printed on chlorine-free bleached paper.

Table of Contents

	Page
1 About this Documentation.....	5
1.1 Validity of the Documentation.....	5
1.2 Documentation Structure.....	5
1.3 Using the Safety Instructions.....	6
1.3.1 Safety Instructions – Structure.....	6
1.3.2 Explaining Signal Words and Safety Alert Symbol.....	6
1.3.3 Symbols Used.....	6
1.4 Required and Supplementing Documentations.....	7
1.5 Names and Abbreviations.....	12
2 First Steps - IndraLogic.....	13
2.1 Project Goal "First steps - IndraLogic".....	13
2.2 Technical Requirements.....	13
2.3 Creating a New IndraWorks Project.....	14
2.3.1 Resetting the Control (CMLx5).....	14
2.3.2 Starting IndraWorks.....	14
2.3.3 Creating a New Project.....	14
2.4 Adding a New Device to a Project.....	15
2.5 Allocating Basic Settings to the Device.....	16
2.5.1 Device Settings.....	16
2.5.2 Communication Settings.....	16
2.5.3 Function Module Settings.....	18
2.6 Creating a PLC Program.....	20
2.6.1 Controlling a Traffic Signal System.....	20
2.6.2 Creating Programs and Function Blocks.....	20
2.6.3 Editing Programs and Function Blocks - Part 1.....	24
2.6.4 Task Configuration.....	40
2.6.5 Declaring Global Variables.....	41
2.6.6 Testing "SEQUENCE" in Simulation Mode.....	42
2.6.7 Editing Programs and Function Blocks – Second Part.....	44
2.6.8 Testing the Application on the Control.....	51
2.7 Creating Visualization for the PLC Program.....	53
2.7.1 Visualization in IndraLogic.....	53
2.7.2 Creating a New Visualization.....	53
2.7.3 Editing Visualization.....	54
2.7.4 Testing Visualization.....	63
2.8 Restoring a Project.....	64
3 First Steps - Motion.....	67
3.1 Project Goal "First Steps - Motion".....	67
3.2 Technical Requirements.....	67
3.3 Creating the Motion Subproject.....	68
3.3.1 Resetting the Control (CMLx5).....	68

Table of Contents

	Page
3.3.2	Starting IndraWorks..... 68
3.3.3	Creating a New Project..... 68
3.3.4	Adding a New Device to a Project..... 69
3.3.5	Specifying Basic Settings for the Device..... 69
3.3.6	Adding sercos III Drives (Real Axes) to the Project 73
3.3.7	Adding a Virtual Axis to the Project..... 74
3.3.8	Downloading the Motion Subproject..... 75
3.4	Programming a Motion Sequence in FBD..... 78
3.4.1	General Information..... 78
3.4.2	Programming the Virtual Axis VA1, Programming Language FBD..... 79
3.4.3	Download of the PLC Sub-project Virtual Axis VA1 (FBD)..... 85
3.4.4	Recording the Motion of Axis VA1 (Trace), FBD..... 88
3.4.5	Programming the Real Axis RA1 (FBD)..... 94
3.4.6	Downloading the Complete PLC Project (FBD)..... 97
3.4.7	Workflow "Virtual Axis → Electronic Gear → Real Axis "..... 97
3.4.8	Recording the Motion of Both Axes (Trace), FBD..... 98
3.5	Programming a Motion Sequence in Structured Text (ST)..... 100
3.5.1	General Information..... 100
3.5.2	Programming the Virtual Axis VA1, Programming Language ST..... 101
3.5.3	Downloading the PLC Sub-project Virtual Axis VA1 (ST)..... 107
3.5.4	Recording the Motion of Axis VA1 (Trace), ST..... 111
3.5.5	Programming the Real Axis RA1 (ST)..... 117
3.5.6	Downloading the Complete PLC Project (ST)..... 120
3.5.7	Workflow "Virtual Axis → Electronic Gear → Real Axis "..... 121
3.5.8	Recording the Motion of Both Axes (Trace), ST..... 122
3.6	Restoring the Project..... 123
4	First Steps - FlexProfile..... 127
4.1	Project Goal "First steps - FlexProfile"..... 127
4.2	Technical Requirements..... 128
4.3	Creating the Motion Subproject..... 128
4.4	Creating a FlexProfile..... 129
4.4.1	Creating a FlexProfile, General Information..... 129
4.4.2	Creating the FlexProfile via IndraWorks Dialogs..... 129
4.4.3	Creating the FlexProfile via PLC..... 132
4.5	Completing the PLC Subproject, Executing a FlexProfile Motion Step..... 135
4.6	Downloading and Testing the PLC Subproject..... 139
4.7	Workflow "Virtual Axis → FlexProfile 0 → Real Axis "..... 140
4.8	Recording the Motion of Both Axes (Trace)..... 141
4.9	Restoring the Project..... 141
5	First Steps - Robot Control V1..... 143
5.1	Project Goal "First Steps - Robot Control V1"..... 143
5.2	Required Hardware and Software..... 143
5.3	Procedure when Creating an RoCo Project 144

Table of Contents

	Page
5.4 Details on the Project "RoCoTestV13".....	154
5.5 Robot Control - Application Examples.....	156
5.5.1 RoCo Basics for Pick & Place Applications.....	156
6 First Steps - Robot Control V2.....	167
6.1 Project Goal "First Steps - Robot Control V2"	167
6.2 Required Hardware and Software.....	167
6.3 Procedure when Creating an RoCo Project	167
6.4 Details on the Project "RoCoV2TestV13"	178
7 Service and Support.....	183
Index.....	185

1 About this Documentation

1.1 Validity of the Documentation

This documentation contains the following descriptions for IndraMotion MLC 13VRS:

- First steps - IndraLogic,
- First Steps - Motion
- First steps - FlexProfile,
- First steps - Robot Control V1,
- First steps - Robot Control V2



Please refer to the following documentation for "First steps with a SafeLogic control":

Rexroth IndraWorks

SafeLogic 13VRS

First Steps

Part number R911339471

Target Group

This documentation is intended for users (first-time users) wanting to learn the basics of an IndraWorks Engineering systems by using a 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.

Application phases

This documentation supports the user during the following phases:

- Engineering
- Commissioning
- Operation and programming

1.2 Documentation Structure

The [chapter 2 "First Steps - IndraLogic" on page 13](#) describes the basics of programming with IndraLogic and of the visualization. The chapter can also be understood without control in the simulation mode (soft PLC).

The [chapter 3 "First Steps - Motion" on page 67](#) describes the basics of the control of a drive from a PLC program (PLCopen), recording motion of axes via Trace. This chapter can also be understood with a real control without drive (with virtual axes).

The [chapter 4 "First Steps - FlexProfile" on page 127](#) contains information to generate and use Motion profiles. This chapter is based on [chapter 3 "First Steps - Motion" on page 67](#).

The [chapter 5 "First Steps - Robot Control V1" on page 143](#) describes the creation of a Robot Control V1 project until the execution of the RoCo source code

The [chapter 6 "First Steps - Robot Control V2" on page 167](#) describes the creation of a Robot Control V2 project including its execution.

The [chapter 7 "Service and Support" on page 183](#) provides information on the customer service helpdesk.

About this Documentation

1.3 Using the Safety Instructions

1.3.1 Safety Instructions – Structure

The safety instructions are structured as follows:

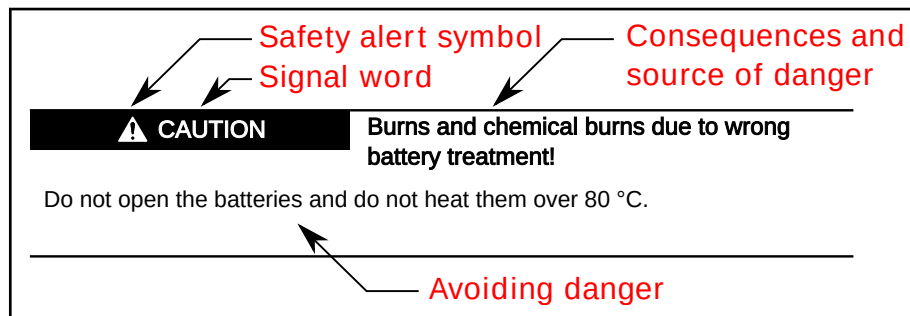


Fig. 1-1: Safety instructions – Structure

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

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

⚠ DANGER	In case of non-compliance with this safety instruction, death or serious injury will occur.
⚠ WARNING	In case of non-compliance with this safety instruction, death or serious injury can occur.
⚠ CAUTION	In case of non-compliance with this safety instruction, minor or moderate injury could occur.
NOTICE	In case of non-compliance with this safety instruction, property damage could occur.

1.3.3 Symbols Used

Notes are displayed as follows:



This is a note.

Tips are displayed as follows:



This is a tip.

1.4 Required and Supplementing Documentations

Documentation titles with type codes and parts numbers

IndraWorks		MLC	XLC
/36/	Rexroth IndraWorks 13VRS Software Installation DOK-IWORKS-SOFTINS*V13-CORS-EN-P, R911336880 This documentation describes the IndraWorks installation.	X	X
/5/	Rexroth IndraWorks 13VRS Engineering DOK-IWORKS-ENGINEE*V13-APRS-EN-P, R911336870 This documentation describes the application of IndraWorks in which the Rexroth Engineering tools are integrated. It includes instructions on how to work with IndraWorks and how to operate the oscilloscope function.	X	X
/20/	Rexroth IndraMotion MLC 13VRS Functional Description DOK-MLC***-FUNC****V13-APRS-EN-P, R911336295 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraMotion MLC.	X	
/20/	Rexroth IndraLogic XLC 13VRS Functional Description DOK-XLC***-FUNC****V13-APRS-EN-P, R911336352 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraLogic XLC.		X
/7/	Rexroth IndraWorks 13VRS CamBuilder DOK-IWORKS-CAMBUIL*V13-APRS-EN-P, R911336291 This documentation describes the basic principles and operation of the CamBuilder, the cam editing tool.	X	X
/37/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Automation Interface DOK-XLCMLC-AUT*INT*V13-APRS-EN-P, R911336356 This documentation describes the script-based access to IndraWorks project data via the interface of the Automation Interface.	X	X
/38/	Rexroth IndraWorks 12VRS FDT Container DOK-IWORKS-FDT*CON*V12-APRS-EN-P, R911334398 This documentation describes the IndraWorks FDT Container functionality. It includes the activation of the functionality in the project and working with DTMs.	X	X

About this Documentation

IndraWorks		MLC	XLC
/29/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Project Conversion DOK-XLCMLC-PROCONV*V13-APRS-EN-P, R911336366 This documentation describes the project conversion of IndraLogic 04VRS and IndraMotion MLC04VRS on IndraWorks version 12 with IndraLogic 2G. Changes with regard to Motion and PLC are described in detail.	X	X
/28/	Rexroth IndraMotion MLC 13VRS Commissioning DOK-MLC***-STARTUP*V13-CORS-EN-P, R911336308 This documentation describes the steps to commission and service the IndraMotion MLC system. It includes checklists for frequent tasks and a detailed description of the steps.	X	

Tab. 1-1: XCL/MLC documentation overview

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

About this Documentation

Motion		MLC	XLC
/10/	Rexroth IndraDrive Firmware for Drive Controllers MPH-, MPB-, MPD-, MPC-07 DOK-INDRV*-MP*-07VRS**-FKRS-EN-P, R911328670 This documentation describes all functional properties of the IndraDrive firmware in the variants MPH-07, MPB-07, MPD-07 and MPC-07.		
/11/	Rexroth IndraDrive MPx-17 Functions DOK-INDRV*-MP*-17VRS**-APRS-EN-P, R911331236 This documentation describes all functional properties of the IndraDrive firmware in the variants MPB-17, MPM-17, MPC-17 and MPE-17.		

Tab. 1-2: XCL/MLC documentation overview

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

Tab. 1-3: XCL/MLC documentation overview

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

Tab. 1-4: XCL/MLC documentation overview

About this Documentation

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

Tab. 1-5: XCL/MLC documentation overview

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

Tab. 1-6: XCL/MLC documentation overview

Hardware		MLC	XLC
/1/	Rexroth IndraControl L45/L65/L85 Control DOK-CONTRL-ICL45L65L85-PRRS-EN-P, R911332116 This documentation describes the IndraControl L45/L65/L85 controls.	X	X
/2/	Rexroth IndraControl L25 DOK-CONTRL-IC*L25*****-PRRS-EN-P, R911328474 This documentation describes the IndraControl L25 controls.	X	X
/24/	Rexroth IndraControl Lxx 13VRS Function Modules DOK-CONTRL-FM*LXX**V13-APRS-EN-P, R911336283 This documentation describes all function modules of the Lxx controls including engineering and diagnostics.	X	X

About this Documentation

Hardware		MLC	XLC
/12/	Rexroth IndraDrive Drive Controllers MPx-02 to MPx-07 DOK-INDRV*-GEN-**VRS**-PARS-EN-P, R911297317 This documentation describes all parameters implemented in the firmware for drive controllers of the IndraDrive family. It supports the parameterization of the drive controllers.		
/13/	Rexroth IndraDrive MPx-02 to MPx-07 and HMV DOK-INDRV*-GEN-**VRS**-WARS-EN-P, R911297319 This documentation describes all diagnostics implemented in the following firmwares: • Drive controller firmwares from MPx-02 to MPx-08 and • firmwares of the supply devices of the type "HMV". It supports the operating crew as well as the programmer at troubleshooting.		
/35/	Rexroth IndraDrive Drive Controller Control Sections CSB01, CSH01, CDB01 DOK-INDRV*-CSH*****-PRRS-EN-P, R911295012 This documentation is a project planning manual to select and use the control sections CSB01, CDB01 and CSH01 for drive controllers of the product families Rexroth IndraDrive M and Rexroth IndraDrive C.		

Tab. 1-7: XCL/MLC documentation overview

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

Tab. 1-8: XCL/MLC documentation overview

System Overview		MLC	XLC
/48/	Rexroth IndraMotion for Printing 13VRS System Overview DOK-IM*PR*-SYSTEM**V13-PRRS-EN-P, R911336289 This documentation describes the product IndraMotion for Printing. It introduces the control systems, drive systems and I/O systems as well as the commissioning and programming.	X	
/48/	Rexroth IndraMotion for Packaging 13VRS System Overview DOK-IM*PA*-SYSTEM**V13-PRRS-EN-P, R911336287 This documentation describes the product IndraMotion for Packaging. It introduces the control systems, drive systems and I/O systems as well as the commissioning and programming.	X	

About this Documentation

System Overview		MLC	XLC
/9/	Rexroth IndraMotion MLC 13VRS System Overview DOK-MLC***-SYSTEM**V13-PRRS-EN-P, R911336310 This documentation provides an overview on the hardware/software components of the automation system IndraMotion MLC in the mentioned version. It helps assembling a system.	X	
/9/	Rexroth IndraLogic XLC 13VRS System Overview DOK-XLC***-SYSTEM**V13-PRRS-EN-P, R911336354 This documentation provides an overview on hardware/software components of the automation system IndraLogic XLC in the respective version. It helps assembling a system.		X

Tab. 1-9: XCL/MLC documentation overview

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

Tab. 1-10: XCL/MLC documentation overview

1.5 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 the user through several dialogs for an ergonomic data input
Function block	Provides application-specific functionalities
IEC	International Electrotechnical Commission
sercos	sercos (SErial Realtime 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. 1-11: Names and abbreviations used

2 First Steps - IndraLogic

2.1 Project Goal "First steps - IndraLogic"

The draft and implementation of a simple traffic signal 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. When proceeding according to the operating sequence descriptions, the user familiarizes himself with the IndraWorks and receives information about how the individual components interact.

Project restoration:

Irrespective of the hardware used, for example XLC L25, L45, L65 (with or without sercos III bus) or IndraMotion MLC L25, L45, L65, it is possible to restore the project on the PC using your control.

To do so, use the export file "Application_FirstSteps_IndraLogic_Basics.iwx" filed in your installation directory under "drive:\All Programs\Rexroth\IndraWorks\Projects". More detailed information can be found under [Restoring a Project, on page 64](#).



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

Subtasks

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

2.2 Technical Requirements

- Software:
 - PC with installed IndraWorks Engineering for IndraMotion MLC 13VRS
- Control:
 - IndraMotion MLC L25
 - IndraMotion MLC L45
 - IndraMotion MLC L65

How to meet these requirements is described in:

- "Rexroth IndraMotion MLC13VRS Functional Description", chapter "Commissioning and Operation"

A control, e.g. type "**IndraMotion MLC L65.1**" or "**IndraLogic XLC L65.1**", and current firmware 13VRS has to be available. Additionally, the following requirements have to be met:

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

First Steps - IndraLogic

- If a sercos III bus is available, the bus 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> located below the display and when **BOOT 1.02** is displayed and keep them pressed until **BOOTSTOP** is displayed.

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

Confirm with <Enter>. **CLEANUP** and subsequently **REBOOT** is displayed.

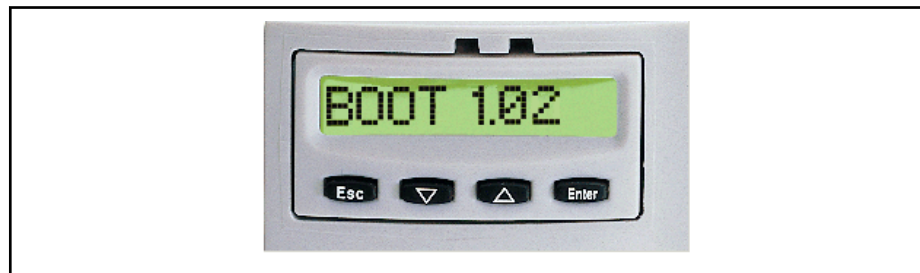


Fig.2-1: Rebooting the control

The control without sercos interface or in case of a bridged sercos interface now runs to **BB RDY**. The memory is cleared and any possible Motion components and the PLC components are ready for operation.

2.3.2 Starting IndraWorks

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

The IndraWorks engineering environment is started. Alternatively,

IndraWorks can also be started via a desktop icon .

2.3.3 Creating a New Project

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

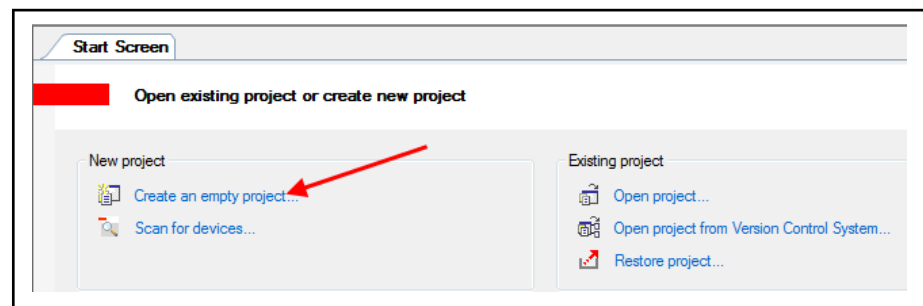


Fig.2-2: Creating an empty project

Alternatively, 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).

First Steps - IndraLogic

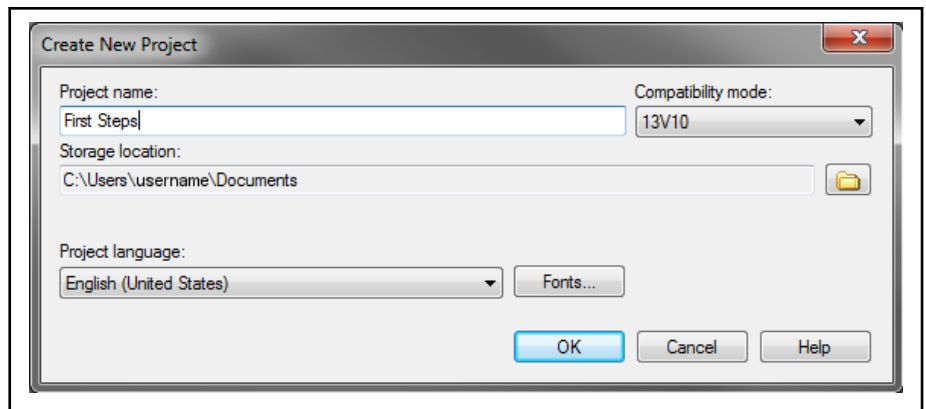


Fig.2-3: Settings to save a new IndraWorks Project

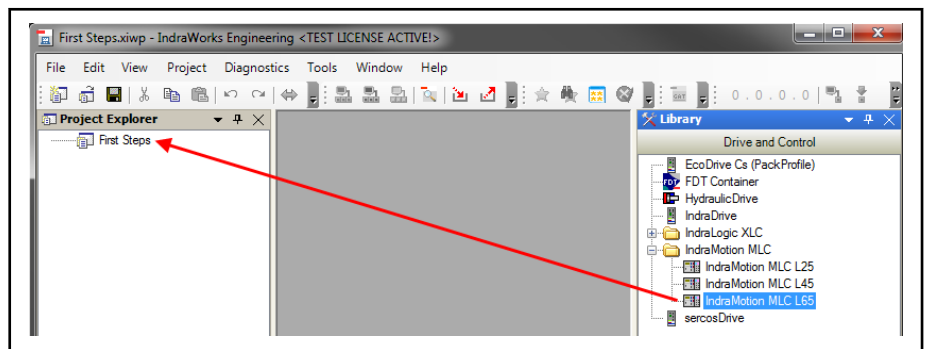
Project name	Specify a name for the project. Suggestion: "First steps".
Storage location	The field contains a storage location in the directory "C:\Documents and Settings\<username>\MyDocuments" by default. A different directory can also be specified.
Compatibility mode	The scope of the IndraWorks interface corresponds to version 13V10.
Project language	By selecting the project language, the language of the user texts is determined. The created project is displayed in the Project Explorer under the previously defined project name:



Fig.2-4: "First steps" project

2.4 Adding a New Device to a Project

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



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

Fig.2-5: Adding a device

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

First Steps - IndraLogic

2.5 Allocating Basic Settings to the Device

2.5.1 Device Settings

Enter the general information on the device to be added.

The screenshot shows a software window titled "Insert IndraMotion MLC L65". Inside, it says "Step 1: General Settings" and "Inserting IndraMotion MLC L65. Enter device name, a comment and the author." There are three input fields: "Device name" with the text "Mlc1", "Comment" which is a large text area containing the placeholder "<Please enter your comment here!>", and "Author" with the text "username".

Fig. 2-6: Device settings

- | | |
|--------------------|--|
| Device name | At "Device Name", enter the desired device name, e.g. "Mlc1". By default, the field contains the name from the library. The device is displayed later in the Project Explorer under the name specified here. |
| Comment | Add a comment that describes the device in detail. This field is optional. |
| Created by | Enter the name of the project author. This field is optional. |



These entries can also be changed later on.

Confirm the entries with **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).

First Steps - IndraLogic

Fig.2-7: Configuration

Device type

The device type is predetermined by selecting the control. If there are several different variants available for one device type, the device variant has to be selected from the drop-down list (example: IndraLogic XLC Lx5 is available with sercos III bus (CMLx5.1-3P) and without sercos III bus (CMLx5.1-NP, IndraMotion MLC Lx5 is always available with sercos III bus (CMLx5.1-3P)).

Firmware version

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

Firmware release

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

IP address



The Ethernet communication entries are not evaluated in the [Simulation mode](#), on page 42. If the control is operated in simulation mode, the connection test is not required.

Enter the target address in "IP address", i.e. the IP address of the control where the application should be loaded.

PLC Gateway

In "Gateway IP address", define the address of the computer where the gateway server is running. By default, "localhost" is displayed as computer name (address). "localhost" means that the locally installed gateway has been addressed. This setting is not to be changed.

PLC communication

Communication via TCP protocol is recommended. Communication via UDP must only be used for SafeLogic.

Connection test

Click on the **Execute** button to check whether the communication between computer and target control has been established successfully. The test result is recorded in the field to the right of "Connection test".

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.

First Steps - IndraLogic

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 IP address in the dialog or in the control (see Rexroth IndraMotion MLC 13VRS Functional Description, chapter "Commissioning and Operation", subchapter "Setting the IP address")

- The network mask settings of the PC and/or control is too small

Solution: Change IP address in the dialog or in the control (see Rexroth IndraMotion MLC 13VRS 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 Rexroth IndraMotion MLC 13VRS Functional Description, chapter "Control", subchapter "Firmware Management")

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

Secure online mode

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

PLC programming language

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

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

2.5.3 Function Module Settings



Depending on the device type, the "Interfaces" dialog may differ. In this project, the dialog of the "IndraMotion MLC L65" device is described as example.

Enter the desired device and function module settings.

First Steps - IndraLogic

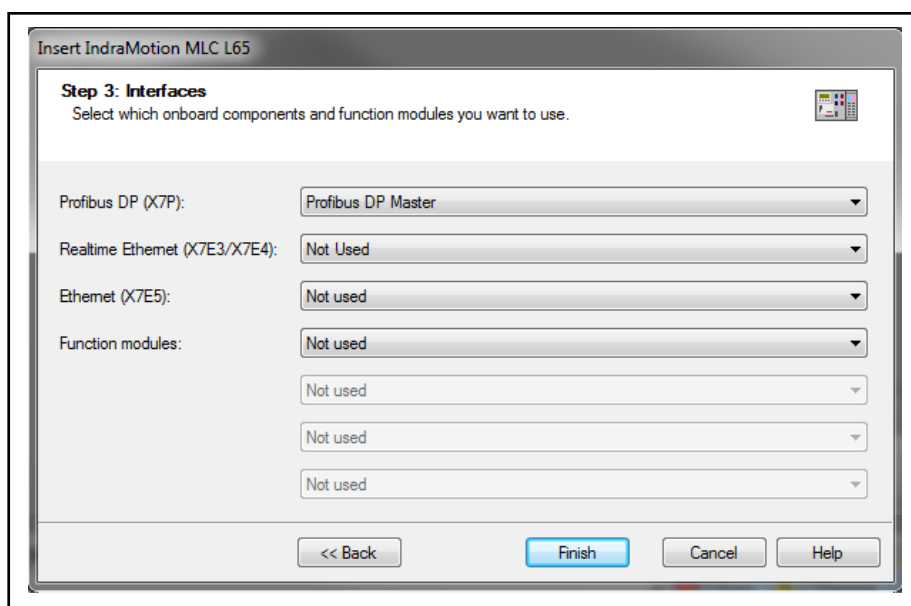


Fig.2-8: Function module settings

sercos (X7E1/X7E2)



This option is only available for IndraLogic XLC controls with sercos interface.

Profibus DP (X7P)

Current sercos III configuration that can be selected when the device is added. The selection includes "sercos III master" or "Not used".

Real-time Ethernet (X7E3/X7E4)

Current PROFIBUS configuration that can be selected when adding a device. "Profibus DP Master", "Profibus DP Slave" or "Not Used" can be selected.

Ethernet (X7E5)

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.

Function modules

As this interface is needed as an Engineering interface (connection directly to your PC or via the operating network), the Ethernet/IP Adapter cannot be checked.

Up to four function modules can be connected to the "IndraMotion MLC L65". If required, select the function modules from the drop-down list.

Confirm the entries with **Finish**.

The newly created device is displayed in the Project Explorer.

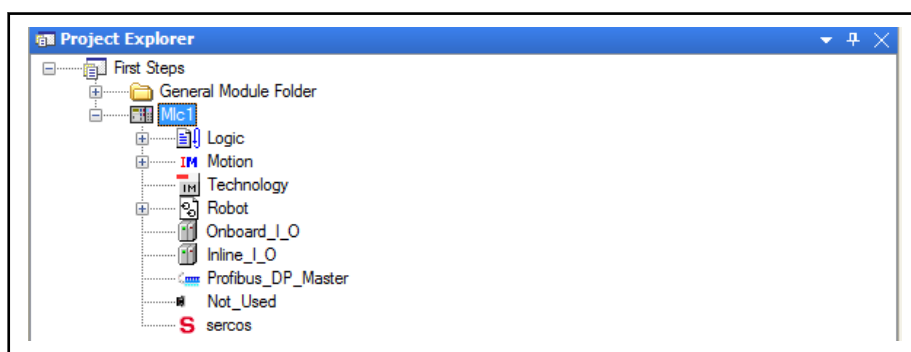


Fig.2-9: Project Explorer: "First Steps" project with an IndraMotion MLC L65 control (with sercos III bus)



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

2.6 Creating a PLC Program

2.6.1 Controlling a Traffic Signal System

The following describes the example program "Mini traffic signal". The program is supposed to control two traffic signals at an intersection. Both traffic signals will switch between red and green phases. To prevent accidents, a yellow switching phase or yellow and red switching phases should also be added between the red and green phases.

In this example, the following is described:

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

2.6.2 Creating Programs and Function Blocks

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

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

The POU "MlcProg" is not needed in this project.



For the "PlcProg" program it is possible to select:

- When creating in "Structured Text (ST)", omit the remainder of this chapter and continue with [Adding SEQUENCE Program in AS, on page 22](#).
- When creating in "Continuous Function Chart (CFC)", continue below

Proceed as follows:

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



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

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

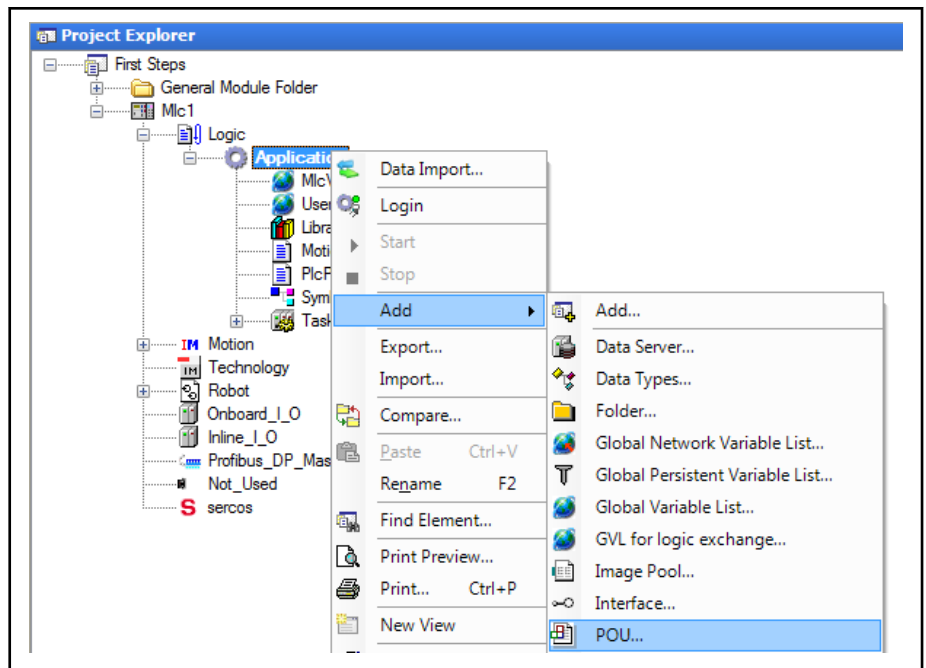


Fig.2-10: Adding a POU from the library

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

1. In the "Name" input field, give the POU the name "PlcProg".
2. Under "Type:" select "Program" and as "Implementation language:" (programming language), select "Continuous function chart (CFC)".

Click on **Finish** to confirm your entries.

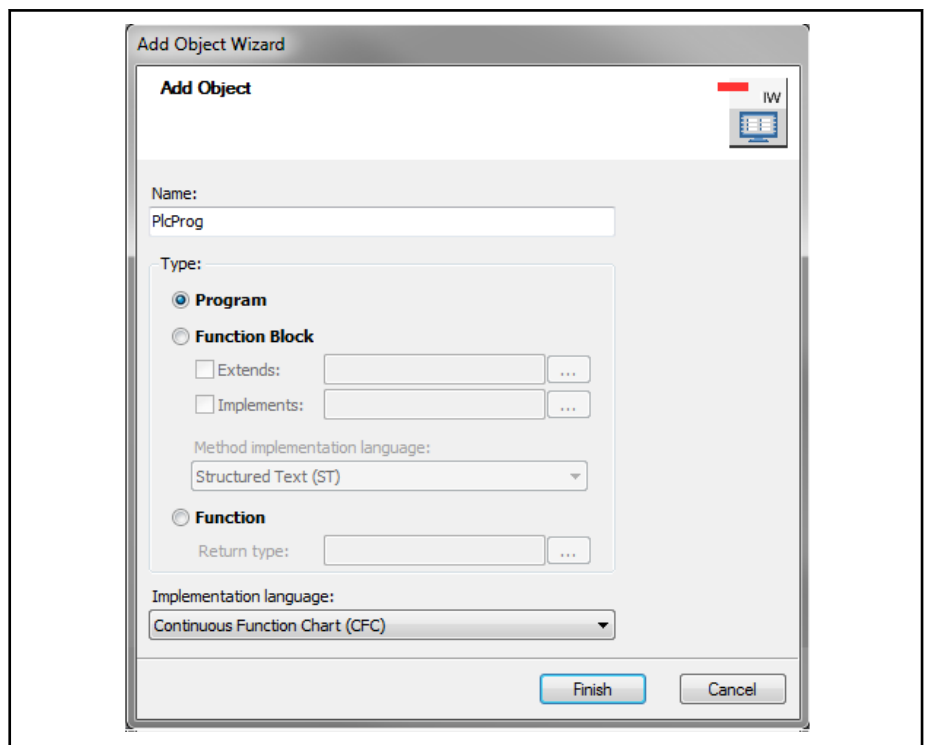


Fig.2-11: Configuring the POU "PlcProg"

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

First Steps - IndraLogic

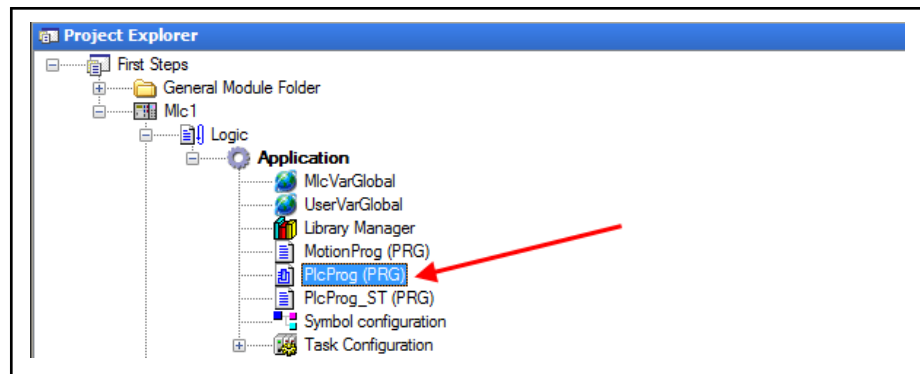


Fig.2-12: "PlcProg" POU has been added to the application

"PlcProg" tasks

In "PlcProg", an incoming start signal is coupled with the signal phase sequence and the outputs for controlling each of the traffic signal's colors are controlled.

Adding "SEQUENCE" Program in "AS"

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

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

1. In the "Name" input field, give the POU the name "SEQUENCE".
2. Under "Type:" select "Program" and as "Implementation language:" (programming language) "Sequential function chart (SFC)".
3. Click on **Finish** to confirm your entries.

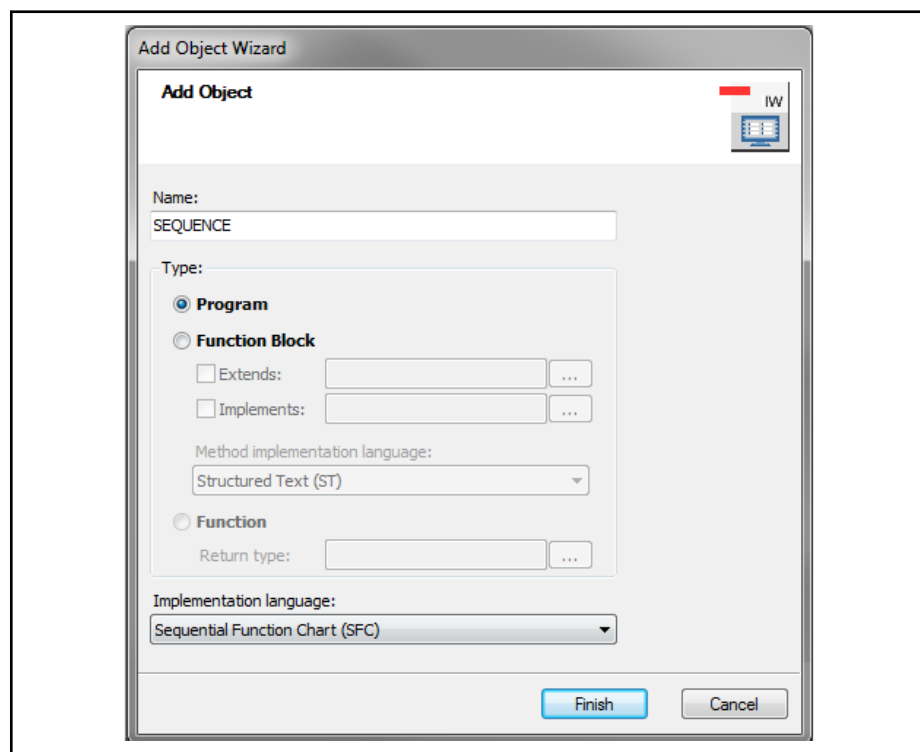


Fig.2-13: Add "SEQUENCE" program

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

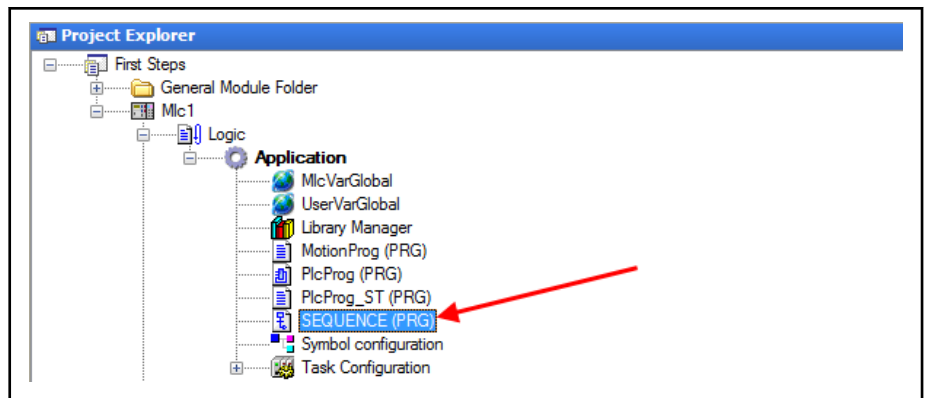


Fig.2-14: "SEQUENCE" program as part of the application

"SEQUENCE" tasks

The sequence of the traffic signal's control phases is programmed in the "SEQUENCE" program. The "WAIT" FB is called, for example, or values for the individual signal phases are saved in the "TRAFFICSIGNAL" with your device so that the correct signal is lit at the correct time for the correct duration.

Adding "TRAFFIC SIGNAL" program in "FBD"

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

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

1. In the "Name" input field, give the POU the name "TRAFFICSIGNAL".
2. Under "Type:" select "Function Block" and as "Implementation language:" (programming language) "Function block diagram (FBD)".
3. Click on **Finish** to confirm your entries.

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

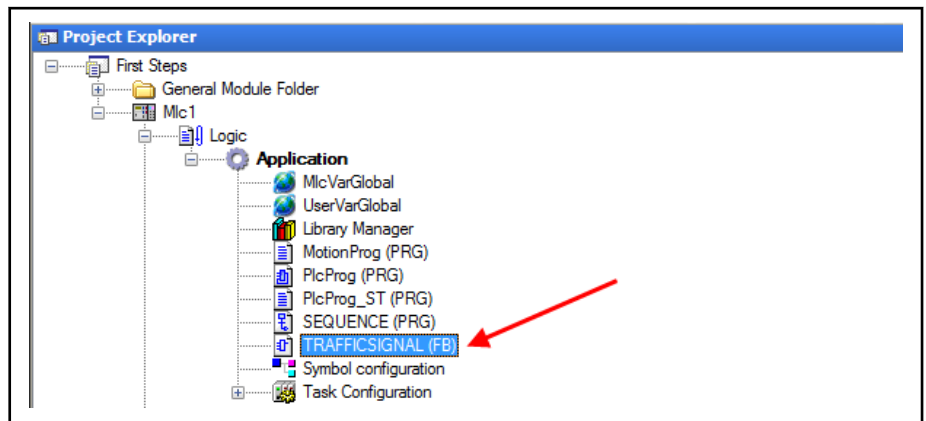


Fig.2-15: "TRAFFICSIGNAL" function block as part of the application

"TRAFFICSIGNAL" task

The individual signal phases are assigned to the traffic signals in the "TRAFFICSIGNAL" function block. The red signal light is actuated during the red or the yellow-red signal phase. The yellow signal light is actuated during the yellow or the yellow-red signal phase. The green light is actuated during the green signal phase. Also see the following table.

First Steps - IndraLogic

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

Tab.2-1: Traffic signal phase assignment

Adding Function Block "WAIT" in "ST"

The last FB to add is the POU called "WAIT".

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

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

1. In the "Name" input field, give the POU the name "WAIT".
2. Under "Type:" select "Function Block" and as "Implementation language:" (programming language) "Structured Text (ST)".
3. Click on **Finish** to confirm your entries.

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

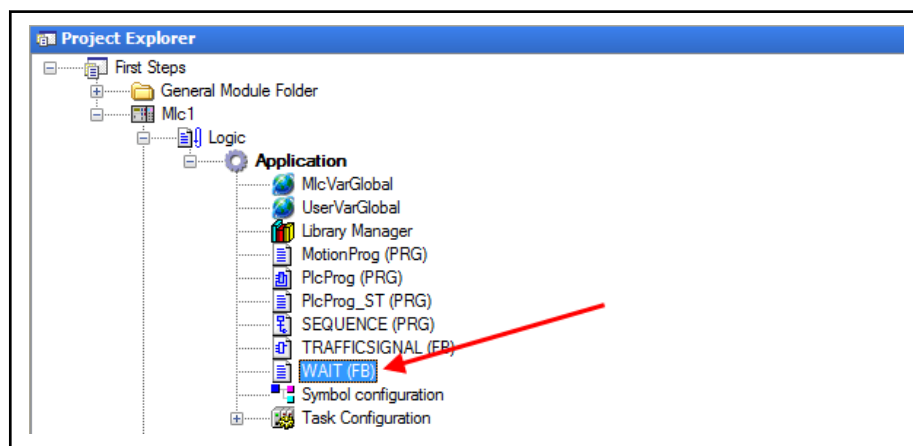


Fig.2-16: "WAIT" function block as part of the application

"WAIT" task

In the "WAIT" POU function block, a simple time step is programmed that uses a "TP" function block from the IndraWorks standard.library. The length of each signal phase can be entered with this timer. The input variable for the FB is the duration of the signal phase in milliseconds and it delivers the Boolean value "TRUE" as an output variable as soon as the time period has expired.

2.6.3 Editing Programs and Function Blocks - Part 1

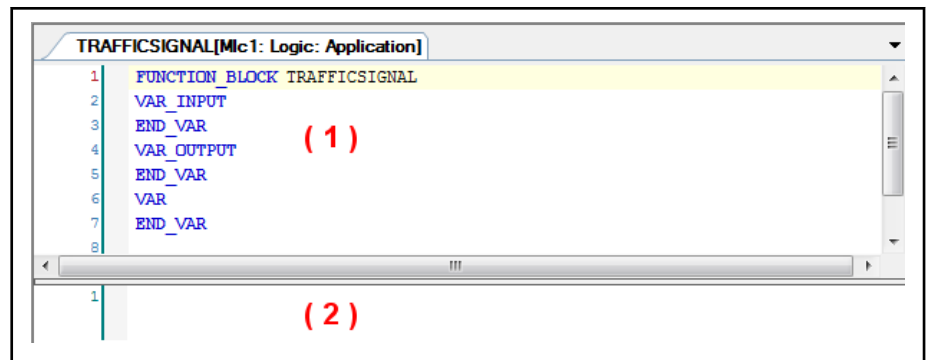
Editing "TRAFFIC SIGNAL" Function Block in "FBD"

Opening the POU

Select the "TRAFFICSIGNAL" function block. Right-click to open the context menu. Select **Open** from the context menu. The editor of the "TRAFFICSIGNAL" function block is opened (see the following figure).



Optionally, open the editor for a POU with a double-click.



(1) Declaration

(2) Implementation section or body

Fig.2-17: Editor of the "TRAFFICSIGNAL" function block opened

The POU editor

The editor is sub-divided into two sections:

1. Declaration part.
2. Implementation part.

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 ways to declare variables in a POU:

1. Place the variables directly in the declarations.
2. Begin programming without declaration and add the data identified by the system in the input window (recommended).

The input window opens automatically if the program does not recognize the variable entered.



1. Create instances of function blocks, of structures (STRUCT) and of data arrays (ARRAY) directly in the declaration section.
2. Variables can be more simply created during programming with the aid of the input window.
Please note the correct assignment to the validity range and the correct data type.
3. If a "declared" variable is offered for declaration, it is generally a typing error.

Creating the user program

The function block "TRAFFICSIGNAL" receives "STATUS", "INTEGER" type, as an input variable to process the program. The output variables "GREEN", "YELLOW" and "RED" for the corresponding signal phases are set to "TRUE". The "TRAFFICSIGNAL" function block is called later in "PlcProg".

Editing the first network

The user program is edited in the implementation part. Select the first network in the implementation part by clicking next to the "1" in the editor. Use the right mouse button to open the context menu and select **Add FB call**. Alternatively, select **FBD/LD/IL ► Add FB call** from the menu.

First Steps - IndraLogic

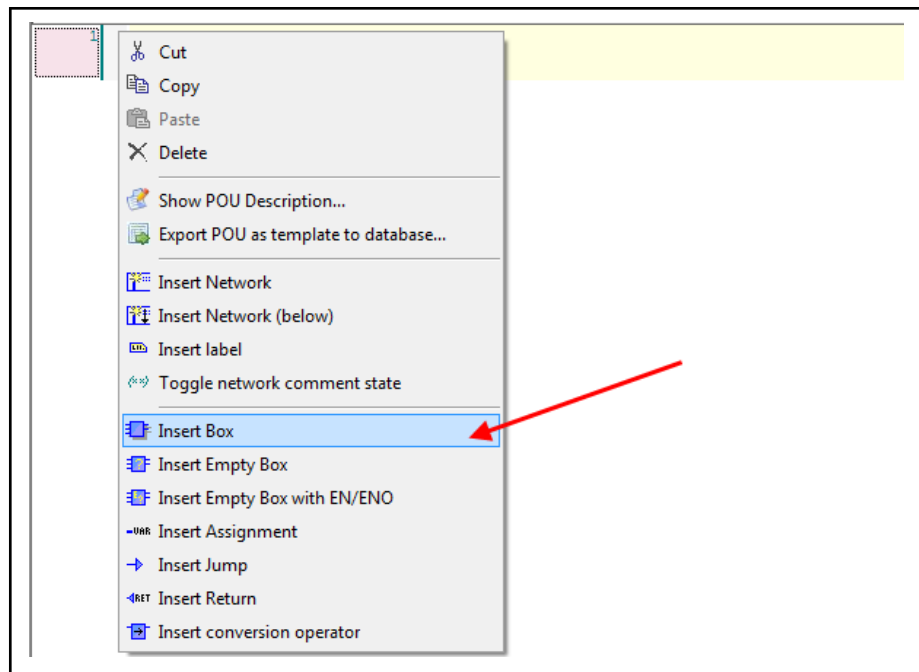


Fig.2-18: Call FB call

The "input assistance" window opens:

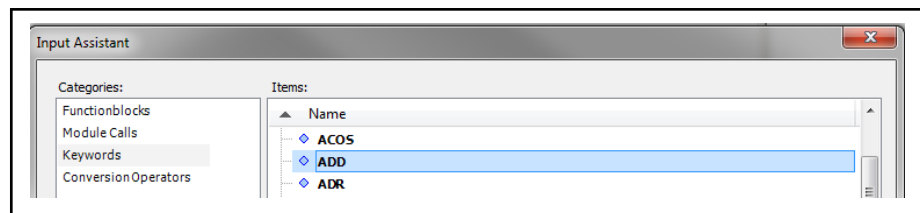


Fig.2-19: Input assistance - adding the function block "ADD"

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

1. In "Input assistance" under "Categories:" select the "key words".
2. Under "Elements:" highlight the operator "ADD".

Confirm the warning with **OK**.

The statement section looks like this:

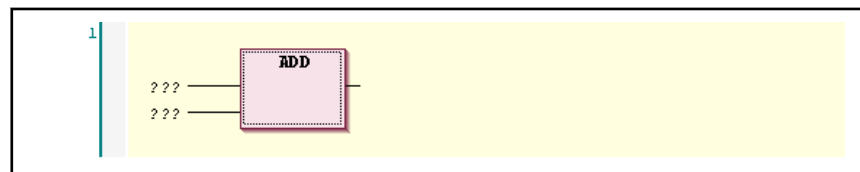


Fig.2-20: "ADD" function block added to the network

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

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

The window for the declaration of variables opens.

The following are offered: VAR, STATUS, INT.

First Steps - IndraLogic

3. As "STATUS" is an input variable of the function block, the suggestion must be changed in "VAR_INPUT, STATUS, INT".

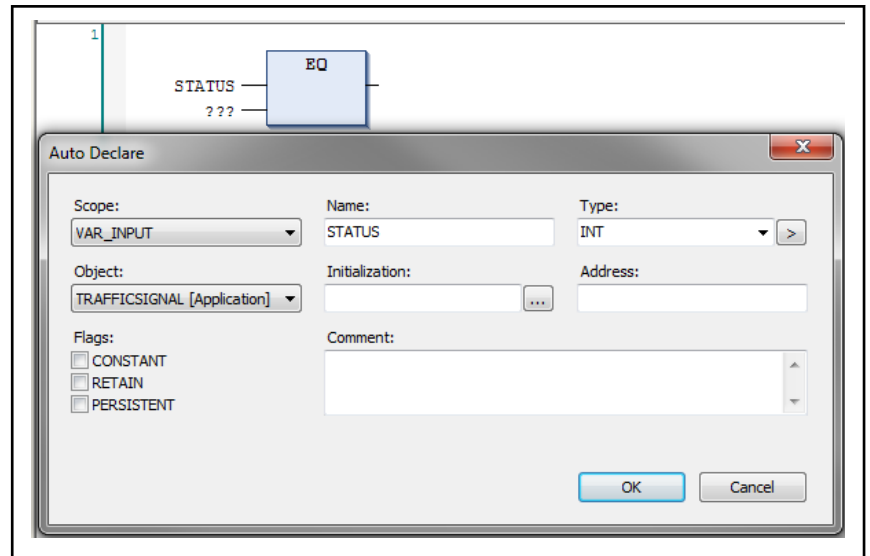


Fig.2-21: Declaration of the input variables "STATUS"

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

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

"STATUS" will have four possible states for the different signal phases. Status:

- Green
- Yellow
- Yellow-red
- Red

The four states should be decoded as output variables. The following three 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. Use the right mouse button to open the context menu and select **Insert assignment**. Click on the output of the "???" function block and give it the variable name "GREEN".

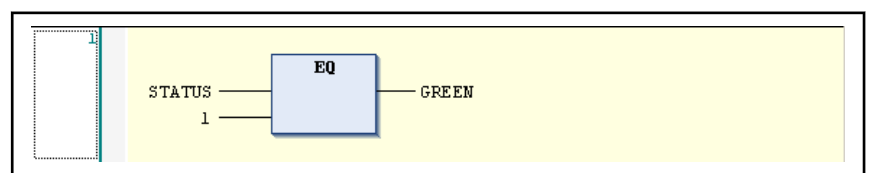


Fig.2-22: "EQ" function block

The window for the declaration of variables opens.

2. As "GREEN" is an output variable of the function block, the suggestion must be changed in "VAR_OUTPUT, GREEN, BOOL".

First Steps - IndraLogic

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

"EQ" task The "STATUS" input variable is compared with "1". The result is "TRUE" if "STATUS" has "1" for a value. The result is "FALSE" if "STATUS" does not have "1" for a value. The result is assigned to the output variable "GREEN".

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

1. Click in the first network and select **Add network (below)** via the context menu.

Editing the second network

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

1. Click with the right mouse button (right click) in the second network.
2. Select **Add FB call** in the context menu.
3. In "Input assistance" for "Add FB call" under "Categories:" select "Key-words".

Under "Elements:" select the element "EQ".

Use the **OK** button to confirm your entries.

4. Highlight the output for the "EQ" function block and click the right mouse button.

5. Select **Add function block call** in the context menu.

6. In "Input assistance" for "Add FB call" under "Categories:" select the "key words".

Under "Elements:" select the element "OR".

Use the **OK** button to confirm your entries.

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

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

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

1. Now select the second (free) input of the "OR" function block by right-clicking on the function block on the connection to the right of "???".
2. Right click to select **Add FB call** from the context menu.
3. In "Input assistance" for "Add FB call" under "Categories:" select the "key words".

Under "Elements:" select the element "EQ".

Use the **OK** button to confirm your entries.

The inputs for both "EQ" function blocks must still be assigned:

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



As "STATUS" is already declared, the declaration window does not appear this time.

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

Copying a network

The third network is the easiest to create since it involves copying the second network.

To copy the network, proceed as follows:

1. Select 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".

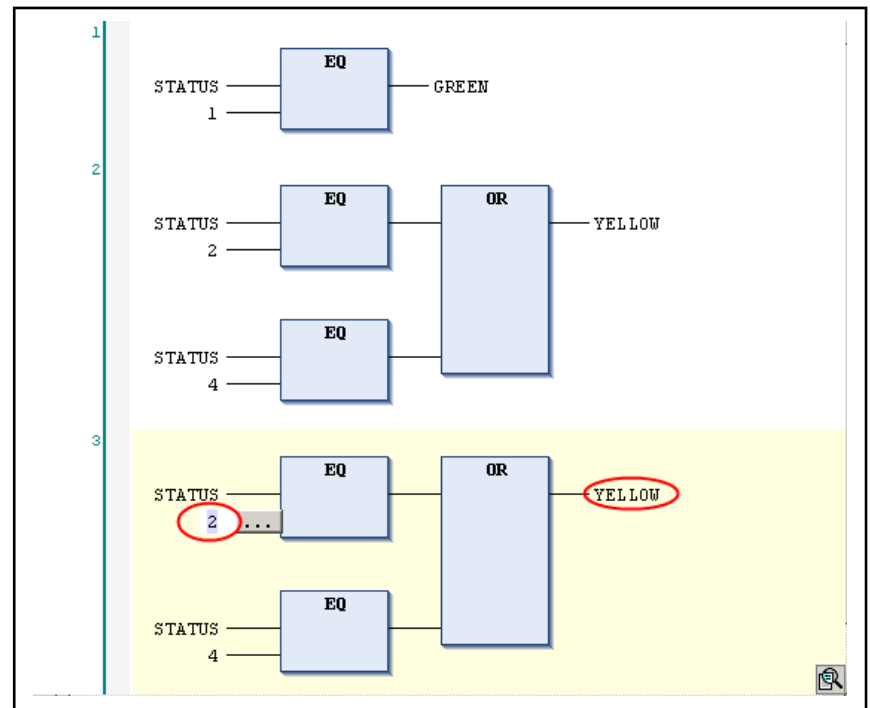


Fig.2-23: Third network added by copying

Editing the third network

To edit the third network, proceed as follows:

1. In network 3 on the upper "EQ function block", select the lower input with "2".
2. With left mouse button, click on the "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.

First Steps - IndraLogic

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

The first POU function block "TRAFFICSIGNAL" is complete.

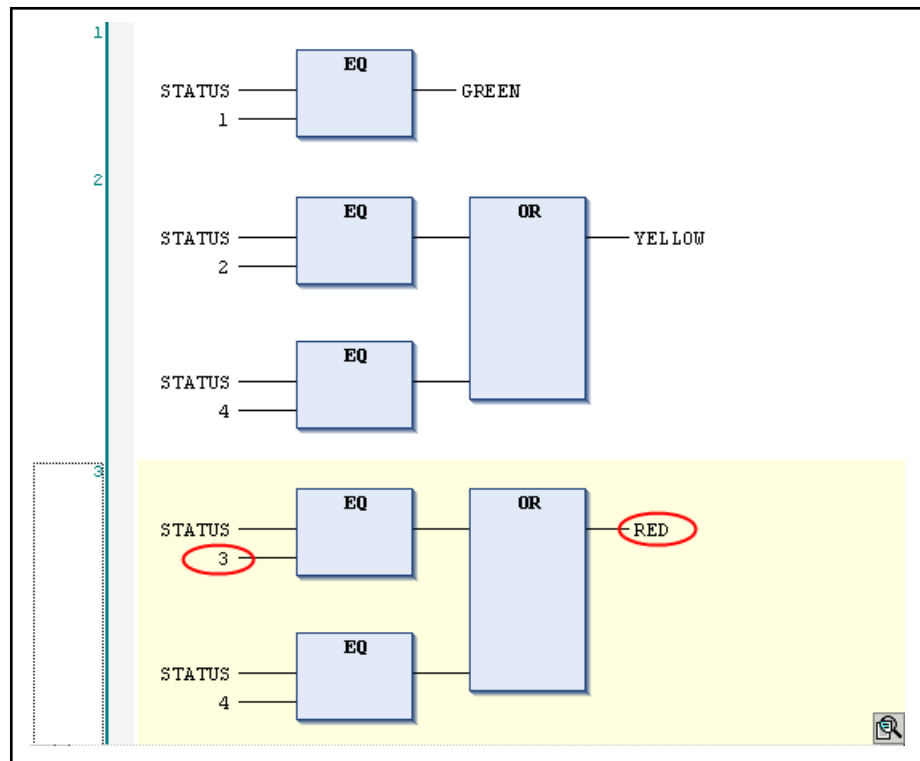


Fig.2-24: POU "TRAFFICSIGNAL" completed

The declaration section is created incidentally to the implementation:

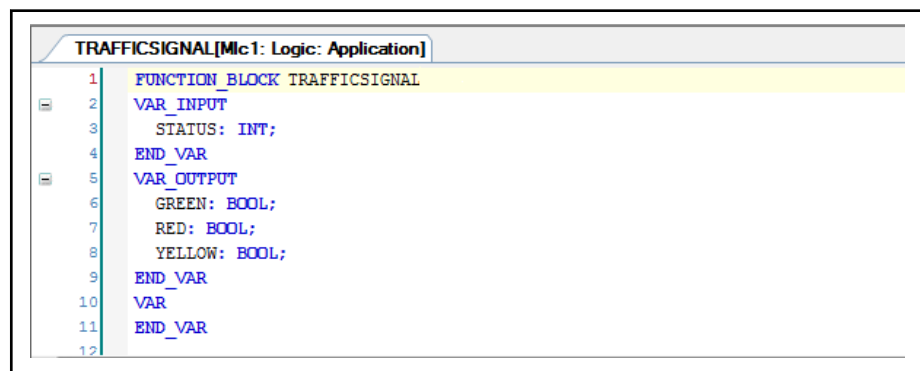


Fig.2-25: Declaration part of the "TRAFFICSIGNAL" function block

Editing "WAIT" Function Block in "ST"

Opening the POU

Open the POU "WAIT" by double-clicking on the object in the Project Explorer:

Declaring and pre-assigning variables

The "WAIT" function block obtains a "WAITINGTIME" variable, type "TIME" as an input variable. The "WAIT" function block delivers an "OK" variable, data type "BOOL", as an output.

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

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

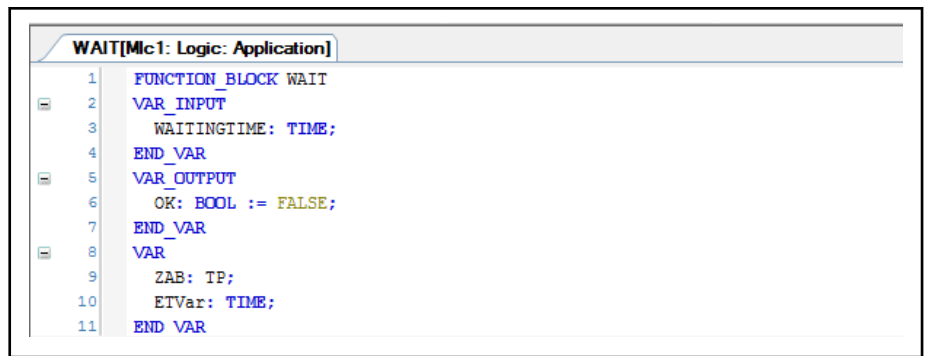


Fig.2-26: Variable declaration POU "WAIT"

"TP" function block of the standard library

The "TP" function block, which has a pulse generator function, is located in the IndraWorks standard library. To access the contents 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. Open the tree structure of the "Timer" folder by clicking on the "+" to the left of the "Timer" folder.

The "TP" function block is in the tree. Click on the "TP" function block for more information.

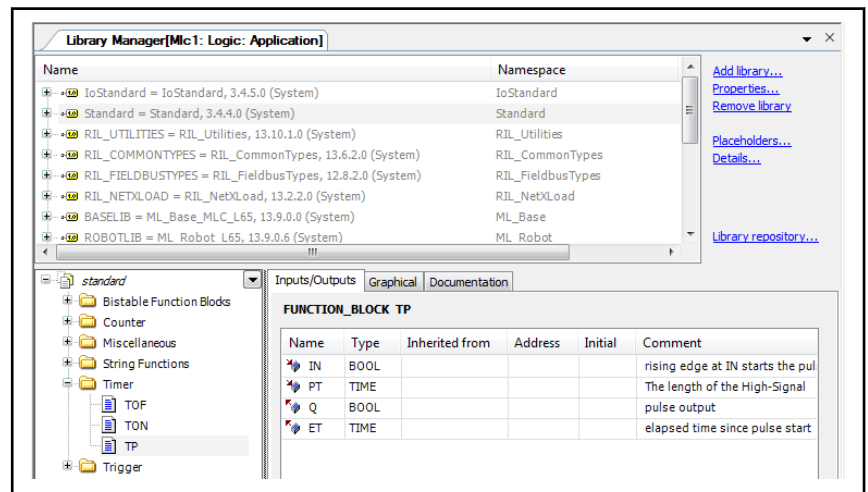


Fig.2-27: Library manager standard function block "TP"

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

"TP" tasks

As long as "IN" has the Boolean value "FALSE", "ET" = "0" and "Q" is "FALSE".

As soon as "IN" delivers the Boolean value "TRUE", the time is added in milliseconds at the "ET" output.

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

"Q" delivers as long as "TRUE" and "ET" are less than "PT".

As soon as the value of the variable "PT" is achieved, "Q" delivers the Boolean value "FALSE".

First Steps - IndraLogic



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

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

or via the point operator

ZAB.Q.

Creating implementation in ST

Complete the implementation section as shown in the following:

```

WAIT[MLC1: Logic: Application]
1  FUNCTION_BLOCK WAIT
2  VAR_INPUT
3    WAITINGTIME: TIME;
4  END_VAR
5  VAR_OUTPUT
6    OK: BOOL := FALSE;
7  END_VAR
8  VAR
9    ZAB: TP;
10   ETVar: TIME;
11 END_VAR

1  IF ZAB.Q = FALSE THEN // Restart Timer: IN: 1-0-1
2    ZAB(IN:= FALSE, PT:= WAITINGTIME, ET=> ETVar);
3    ZAB(IN:= TRUE, PT:= WAITINGTIME, ET=> ETVar);
4  ELSE
5    ZAB(); // Timer continues
6  END_IF
7  OK:= NOT ZAB.Q; // Waiting while Timer runs
  
```

Fig.2-28: Completed "WAIT" POU in ST

"WAIT" task

In the event that the time step has expired, the time step is restarted with a 1-0-1 signal change at "IN" and the current waiting time "WAITINGTIME". Otherwise the time step continues by updating its outputs.

Following expiry of the time step (ZAB.Q = 0), the output "OK" is "TRUE" for a PLC cycle.

Editing "SEQUENCE" Program in "AS"

Opening the program

Open the "SEQUENCE" POU by double-clicking on the object in the Project Explorer:

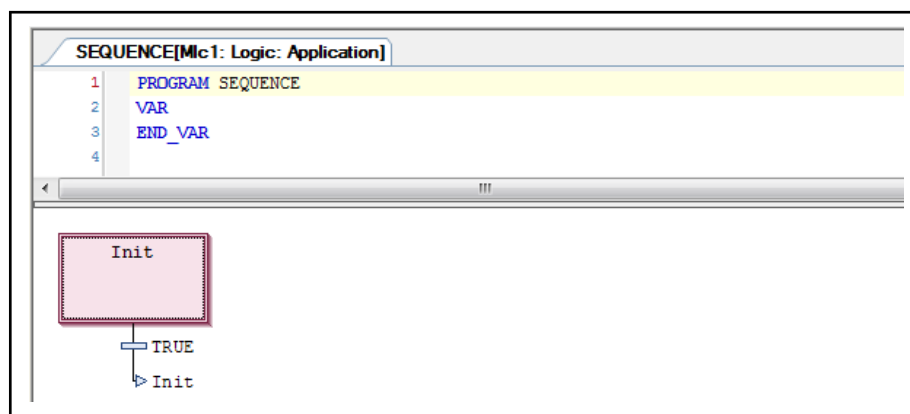


Fig.2-29: Editor: "SEQUENCE" program opened

Declaring variables

The "SEQUENCE" function block obtains a "START" variable, type "BOOL", as an input variable.

The function block delivers the output variables "TRAFFICSIGNAL1" and "TRAFFICSIGNAL2", type "INT", and as a local variable, the function block

has the "DELAY" variable, type "WAIT". First, only the variable DELAY: WAIT; is specified. The other variables are declared later using the Input Assistant in the implementation section.

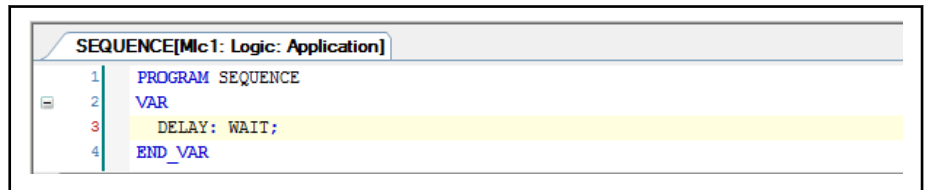


Fig.2-30: Variable declaration for "SEQUENCE" POU - first part

Creating the user program

At the beginning of a sequential function chart, an "Init" action, a subsequent "TRUE" transition and a return to "Init" (see [Editor POU "SEQUENCE" opened on page 32](#)) are located in the body of the "SFC editor".

Specifying the structure of the sequence cascade

First, the structure of the sequence cascade is specified by adding the individual steps and transitions. After adding the necessary steps and transitions, the individual actions for the steps and transitions are programmed. A step is required for each signal phase. In order 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 afterward** from the context menu.
3. Highlight each of the last transitions respectively and add three more steps and transitions.

Sequential function chart:

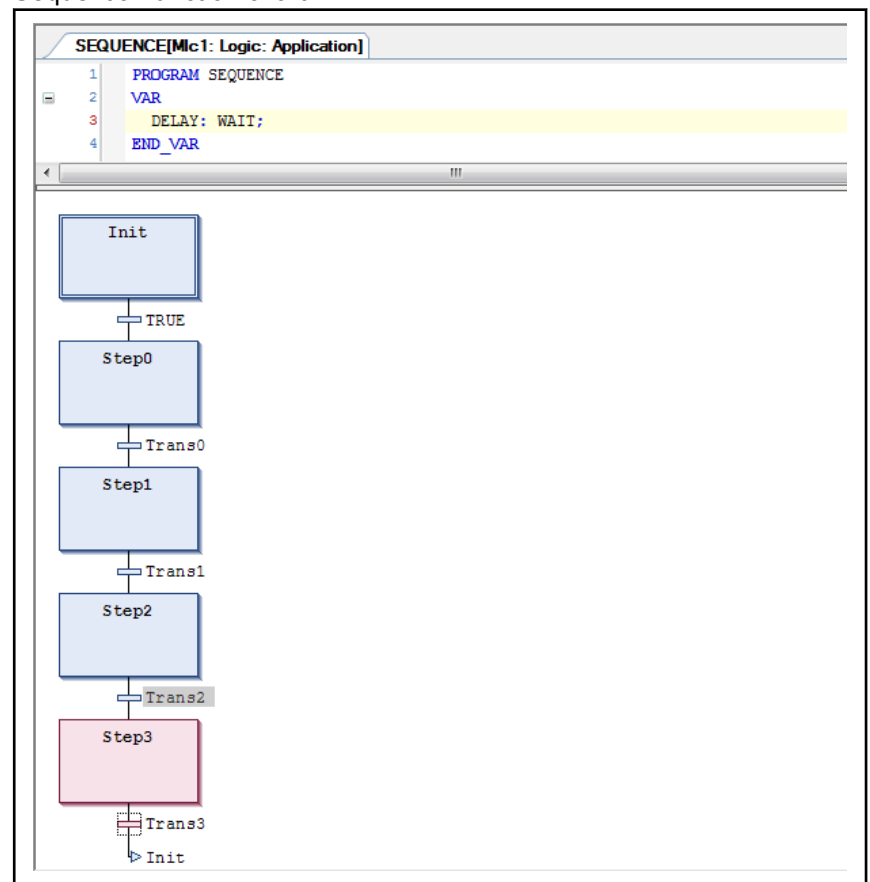


Fig.2-31: Steps inserted in the "SEQUENCE" program

First Steps - IndraLogic



Alternatively you can add steps and transitions using **SFC ► Add step transition afterward**.

Programming transitions

There are two ways to program transitions:

1. Changing the transition name: This change is only useful if only one advance condition is needed, e.g. with a variable inquiry.
2. Programming the transition: Programming sensible if the result of linked variables is intended as an advance condition.

To program a transition, a transition must be added. By double clicking on the "transition", a name can be assigned to the transition and the implementation language (programming language), in which the transition is to be written, can be selected. Subsequently, like a function, the transition can be programmed in an editor.

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

1. Click on the transition name "TRUE" for the first transition.
2. Enter "START" in the input field.

Since the "START" variable has not yet been declared in the declarations, a window opens for declaring variables.

Fig.2-32: Input assistance for variable declaration

3. Under "Type:" select data type "BOOL".
4. For "Visibility:" select variable type "VAR_INPUT".
5. Subsequently, use **OK** to confirm your entries.

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

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

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

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

SFC structure:

First Steps - IndraLogic

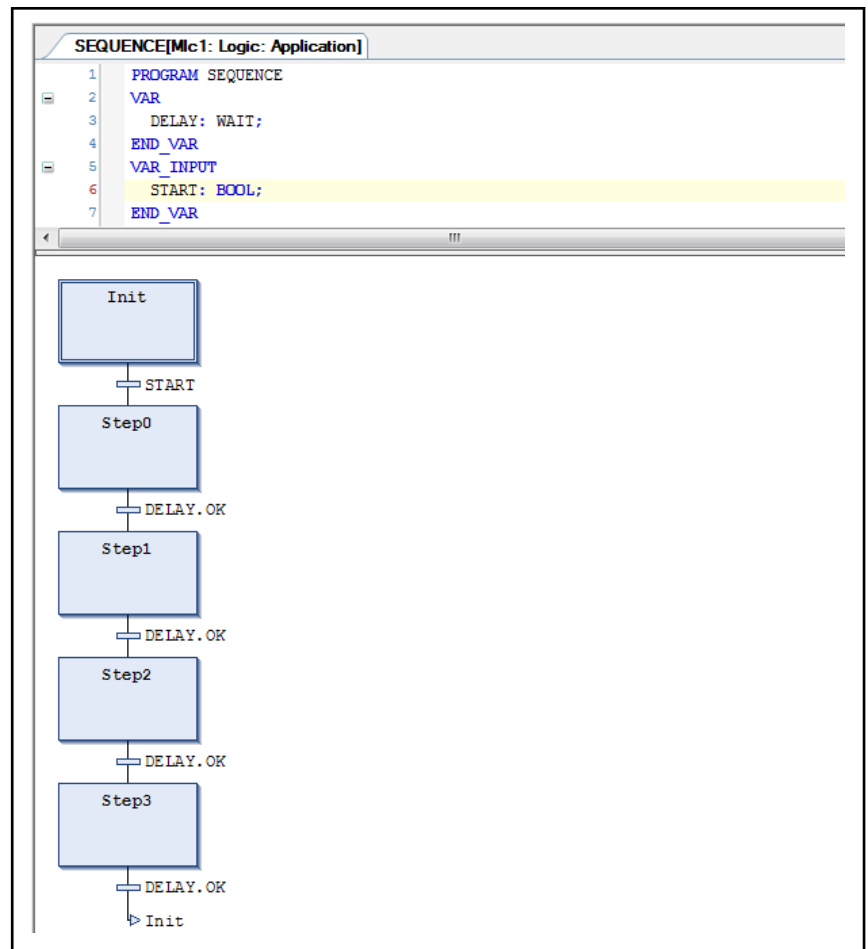


Fig.2-33: Steps of the sequence cascade

Naming the steps Only the "Init" step retains its name. The rest of the steps are renamed. The steps should be named "Switch1", "Green2", "Switch2" and "Green1".

To rename individual steps, proceed as follows:

1. Click on the name of the step.
2. Enter the desired name.
3. Click on <Enter> to confirm your entries.

Function of the steps In steps "Switch1" and "Switch2", the yellow phases are implemented.

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 address Now, after the steps have been run there should no longer be a return after "Init"; instead, a jump should be made to step "Switch1". Click on the name of the "Init" return and enter the new return target "Switch1".

Sequential function chart:

First Steps - IndraLogic

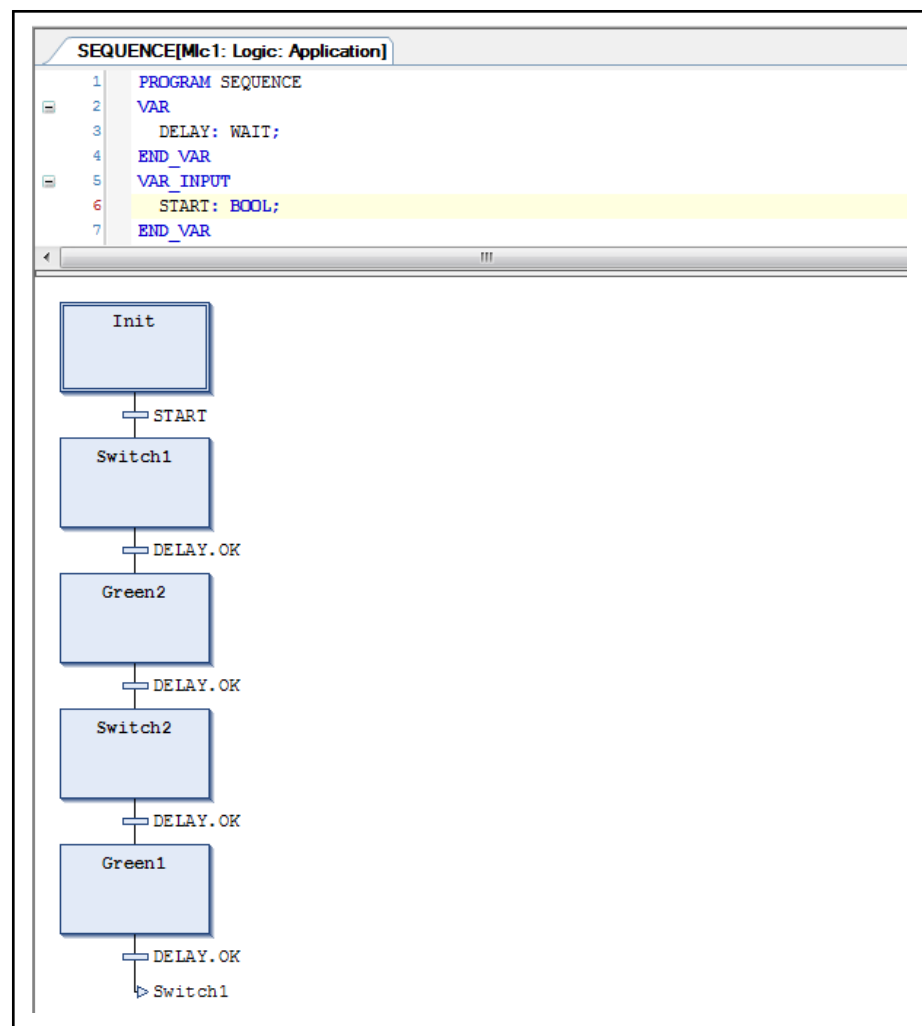


Fig.2-34: Steps of the sequence cascade after naming

Steps and actions

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

IndraLogic supports the following types of actions:

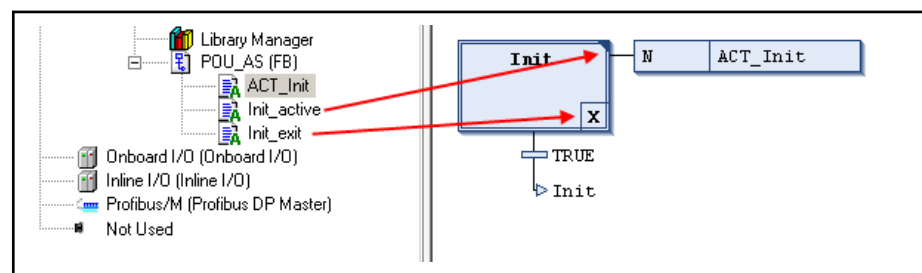


Fig.2-35: Types of step actions

- Input action:** An input action is processed once precisely when the step becomes active. In this case: "Init_active".
 Init_active, a step with an input action is designated by a black triangle in the upper right corner

First Steps - IndraLogic

- **Associated action** The action is created separately as a component 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 appear to the right of the step: If such an action is disabled, the action is post-processed once.

- **Output action:** An output action is only executed once, before the step is disabled. Here: "Init_exit"

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

Programming the "Init" action with "ST"

In the action for the "Init" step, 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 the implementation language (programming language) here.
3. Use the **Open** button to confirm the settings.

In the Project Explorer, under the "SEQUENCE (FB)" POU, a branch is generated with the action "Init_active".

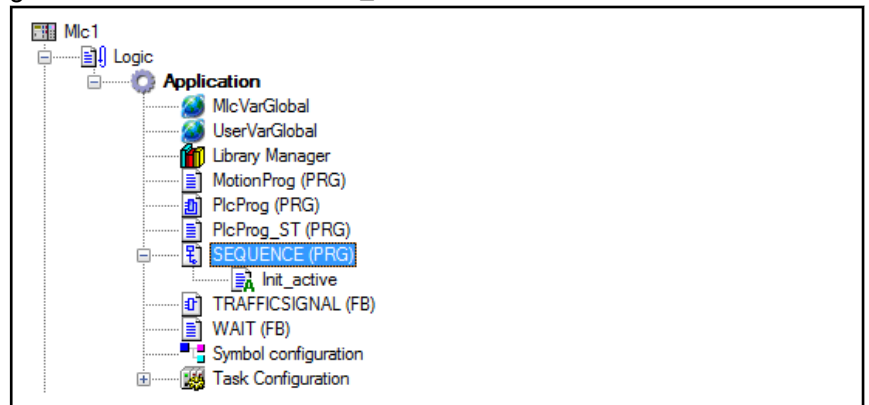


Fig.2-36: POU "SEQUENCE" in the Project Explorer with "Init_active" action

Open the editor for the "Init_active" action.

4. Enter the following program lines in 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 "VAR_OUTPUT" for visibility and "INT" for data type.

The editor for the "Init" action looks like this:

First Steps - IndraLogic

Programming the "Switch1" action with "ST"

In the action for the step "Switch1" the STATUS of traffic signal 1 (TRAFFIC-SIGNAL1) changes to "2" (yellow) and the STATUS for traffic signal 2 (TRAFFIC-SIGNAL2) to "4" (yellow-red). A delay time of two seconds is specified.

To edit "Switch1", proceed as follows:

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

The "Add action" window opens.

2. Here, select "Structured Text (ST)" as the implementation language (programming language).
3. Use the **Open** button to confirm your entries.

In the Project Explorer, under the "SEQUENCE (FB)" POU, a branch is generated with the action "Switch1_active".

The editor for the "Switch1_active" action opens.

4. Enter the following program lines in the editor:

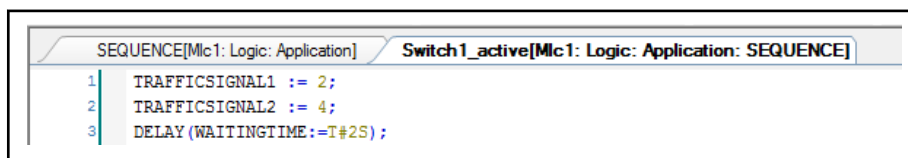


Fig.2-37: Editor of the "Switch1_active" action

Programming the "Green2" action with "ST"

In the action for the step "Green2" the STATUS of traffic signal 1 (TRAFFIC-SIGNAL1) changes to "3" (red) and the STATUS for traffic signal 2 (TRAFFIC-SIGNAL2) to "1" (green). A delay time of five seconds is specified.

To edit "Green2", proceed as follows:

1. Double click on the box for the third step, "Green2", in the "SEQUENCE" POU.

The "Add action" window opens.

2. Here, select "Structured Text (ST)" as the implementation language (programming language).
3. Use the **Open** button to confirm the entries.

In the Project Explorer, under the "SEQUENCE (FB)" POU, a branch is generated with the action "Green2_active".

Open the editor for the "Green2_active" action.

4. Enter the following program lines in the editor:



Fig.2-38: Editor of the "Green2_active" action

Programming the "Switch2" action with "ST"

In the action for the step "Switch2" the STATUS of traffic signal 1 (TRAFFIC-SIGNAL1) changes to "4" (yellow-red) and the STATUS for traffic signal 2 (TRAFFIC-SIGNAL2) to "2" (yellow). A delay time of two seconds is specified.

To edit "Switch2", proceed as follows:

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

An "Add action" window opens.

2. Here, select "Structured Text (ST)" as the implementation language (programming language).
3. Use the **Open** button to confirm the entries.

In the Project Explorer, under the "SEQUENCE (FB)" POU, a branch is generated with the action "Switch2_active".
The editor for the "Switch2_active" action opens.

4. Enter the following program lines in the editor:

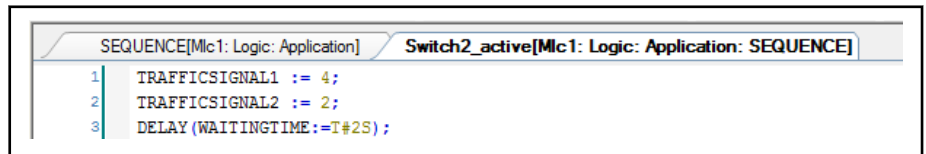


Fig.2-39: Editor of the "Switch2_active" action

Programming the "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 for fifth step, "Green1".

An "Add action" window opens.

2. Here, select "Structured Text (ST)" as the implementation language (programming language).
3. Use the **Open** button to confirm the entries.

In the Project Explorer, under the "SEQUENCE (FB)" POU, a branch is generated with the action "Green1_active".
Open the editor for the "Green1_active" action.

4. Enter the following program lines in the editor:



Fig.2-40: Editor of the "Green1_active" action

The completed "SEQUENCE" function block

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

First Steps - IndraLogic

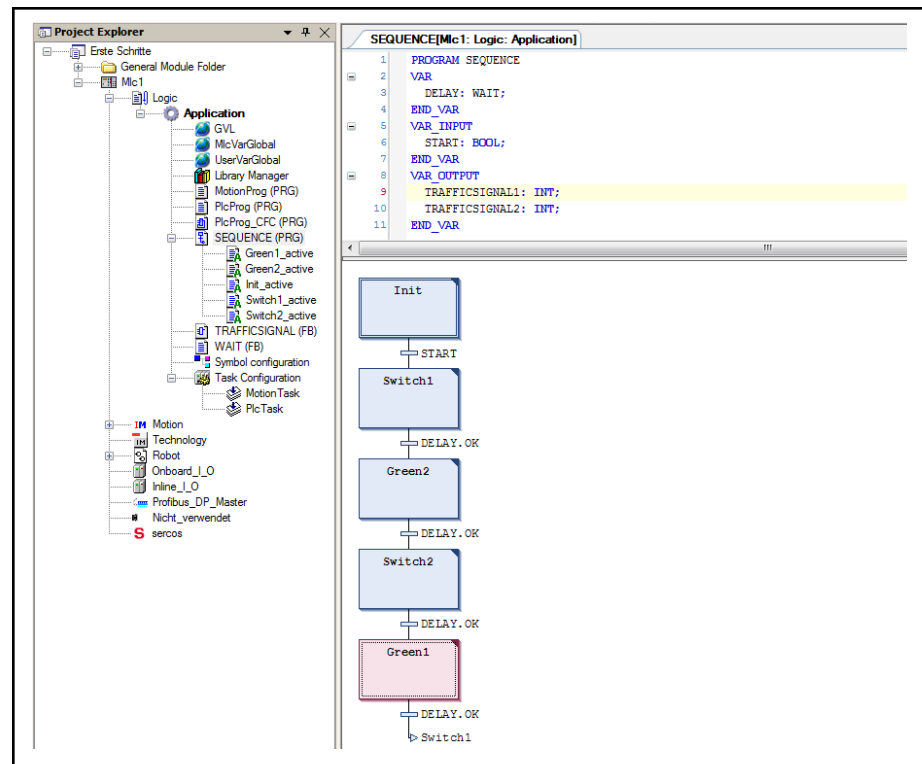


Fig.2-41: POU "SEQUENCE" completed

2.6.4 Task Configuration

The task A "task" is the work that is to be carried out. The task is to call programs by declaring "task types" such as "cyclical" or "free" and by assigning runtimes. The program sequence is organized using a "task".

The task configurator The task configurator defines one or more tasks that control the execution of an application.

Adding the "SEQUENCE" task In the application are two "program" type POU: "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 so that the "SEQUENCE" POU can be called.

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

1. In the Project Explorer, open the "PlcTask" configuration by double-clicking below "Task configuration".
2. In the "POUs" section select the open configuration "Insert POU".
The "input assistance" window opens automatically.
3. In "Input assistance" under "Elements:" open the tree under "Mlc1" until "SEQUENCE" and "PlcProg" are visible.
4. Select "SEQUENCE" and confirm with the **OK** button.

The "Plc task configuration" now looks like this:

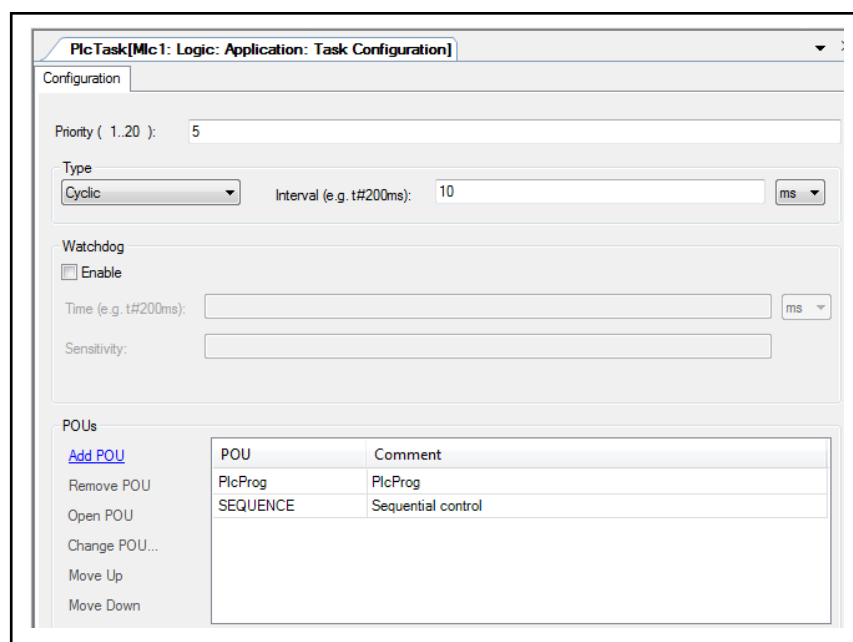


Fig.2-42: Task configuration

2.6.5 Declaring Global Variables

GVL function The traffic signal system is switched on by the "ON" switch.

The respective traffic signal of the two traffic signals is controlled depending on the program status for the corresponding signal phase.

Input and output variables must be created for the one input and the six outputs for both traffic signals.

The physical addresses for the inputs and outputs can be accessed with the syntax "%IX" for an input and "%QX" for an output.

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

The variables are not only available locally in the "PlcProg" function block but also in other function blocks. For this reason, the variables are not declared in the "PlcProg" POU; but instead in a global variable list "GVL".

Creating "global variable lists (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
```

First Steps - IndraLogic

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

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



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

The controls IndraLogic XLC / IndraMotion MLC L45 and L65 have these outputs as onboard outputs (standard mapping).

The IndraLogic XLC / IndraMotion MLC L25 controls do not have any onboard outputs. These controls have to be extended by In-line output terminals or a field bus with I/O modules. If not, avoid the absolute address or you will obtain a warning when generating the code.

2.6.6 Testing "SEQUENCE" in Simulation Mode

The "SEQUENCE" program function block is completed to the point that the program function block can be tested. The test should be carried out in simulation mode.

Compiling the "Application"

In order to load the application into the control, the application must first be compiled by the "compiler" with no errors. To compile the application, select **Create ► Build**. After compiling, the following is displayed in the "Task list" below the task window: "Compiling completed -- 0 errors, 0 warnings"

Simulating a control

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

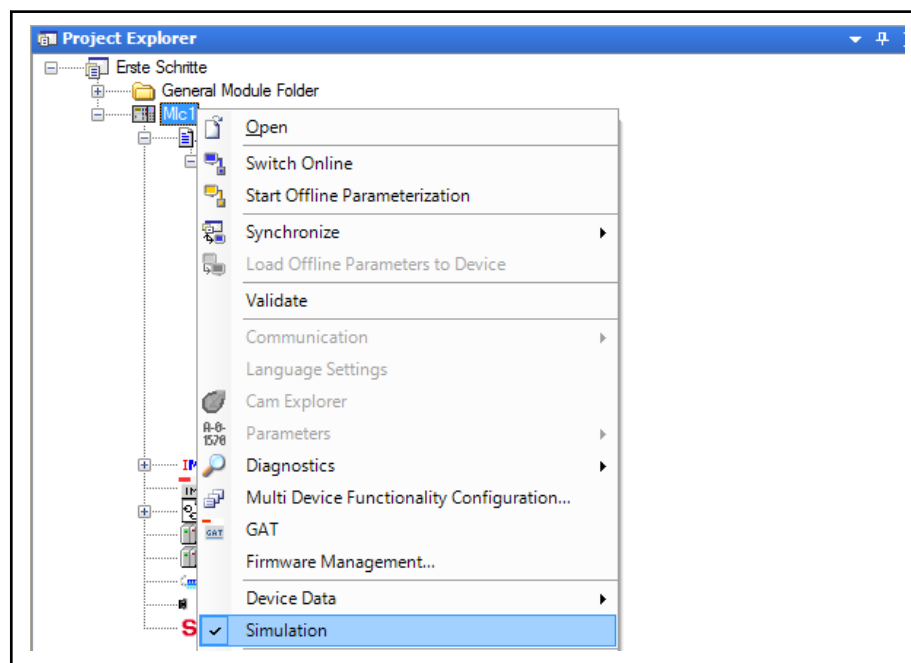


Fig.2-43: Selecting "Simulation"

Logging in to the control

Log into the control and load the application into the control. To log in on the control, select **Debug ► Login** in the IndraWorks menu.

First Steps - IndraLogic

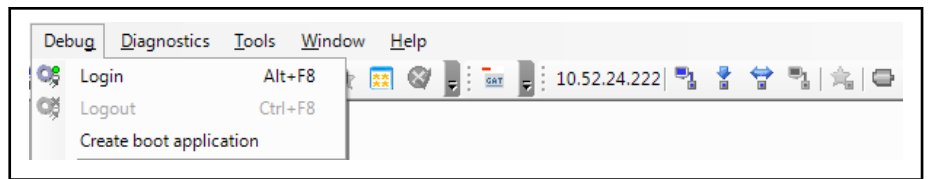


Fig.2-44: Logging into the control

the Project Explorer looks like this:

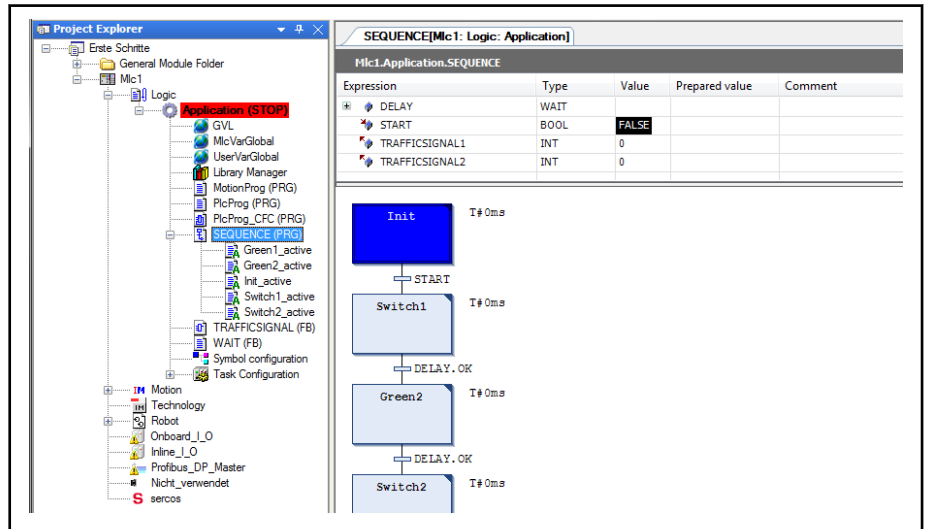


Fig.2-45: Logged into control (Simulation) - "STOP" application

Starting the application

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

The application is running on the control.

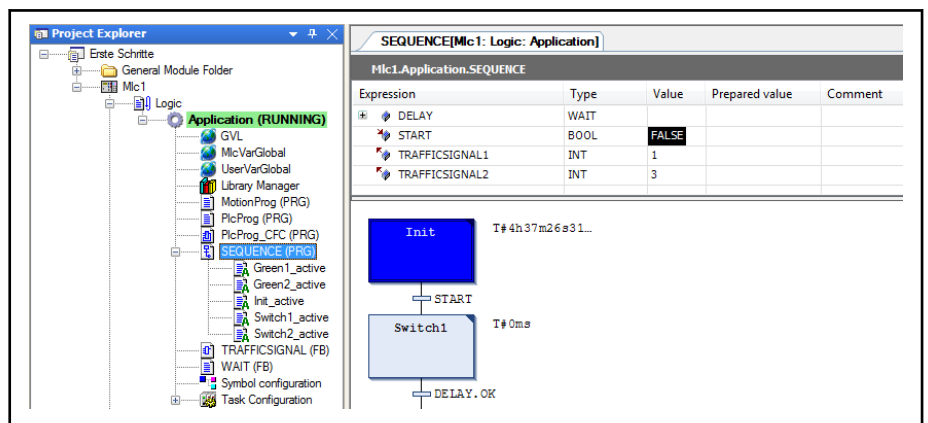


Fig.2-46: Logged into the control - application is running

Starting "SEQUENCE"

Open the "SEQUENCE" POU. In the declarations' online mode you can see the variables and their current values. In the declaration part, click in the line next to 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.2-47: Declaration part POU "SEQUENCE" online
Select **Debug ► Write values** (<Ctrl>+<F7>) in the IndraWorks menu.
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.2-48: Declaration part POU "SEQUENCE" online
The individual steps and transitions of the "SEQUENCE" POU are processed. At the end, a jump is made back to the "Switch1" step.
The values for the "START" variable are set later in the "PlcProg" POU.
Before you log out of the control, stop the application. To stop the application, right click on the application and select **Stop** in the context menu or **Debug ► Stop** in the IndraWorks menu.
The application is stopped on the control.
Log out by selecting **Debug ► Logout from** the IndraWorks menu.
IndraWorks is in offline mode.

Logging out of the control



Upon a new login, the values at the time of the last log out are used.

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

2.6.7 Editing Programs and Function Blocks – Second Part
Editing "PlcProg" Program in "ST"



The implementation of the "PlcProg" program was alternatively offered in two programming languages:

- For programming in "Continuous Function Chart (CFC)", refer to [Editing PlcProg Program in CFC, on page 47](#)
- For programming in "Structured Text (ST)", continue below.

"PlcProg" tasks

The time sequence of the signal phases for both traffic signals was implemented in the "SEQUENCE" block. The inputs and outputs of a PLC are to be controlled and evaluated. Input and output variables for the one input and six outputs for the two traffic signals 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 also be tested in a real control.

Declaring an instance

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

This instances the "TRAFFICSIGNAL" function block.

To specify the variables, refer to [Declaring variables, page 25](#).

Create variable LIGHT1: TRAFFICSIGNAL;

and LIGHT2: TRAFFICSIGNAL;;

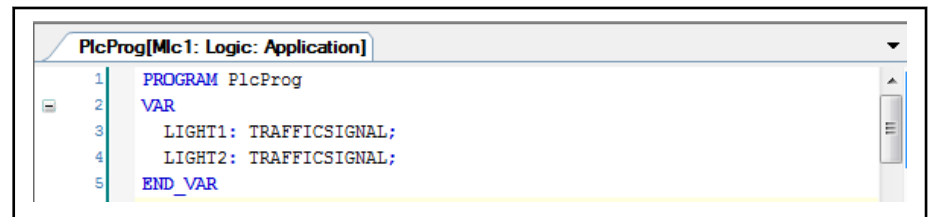


Fig.2-49: Declaration part POU "PlcProg"

Editing the body

The "PlcProg" program is edited in the "ST" programming language. The first function block called 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", "TRAFFICSIGNAL" type, are called in the following.

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

Proceed with the function block instance "LIGHT2" in the same way 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 still uses the global variable of the same name as a replacement.

The "PlcProg" program looks like this:

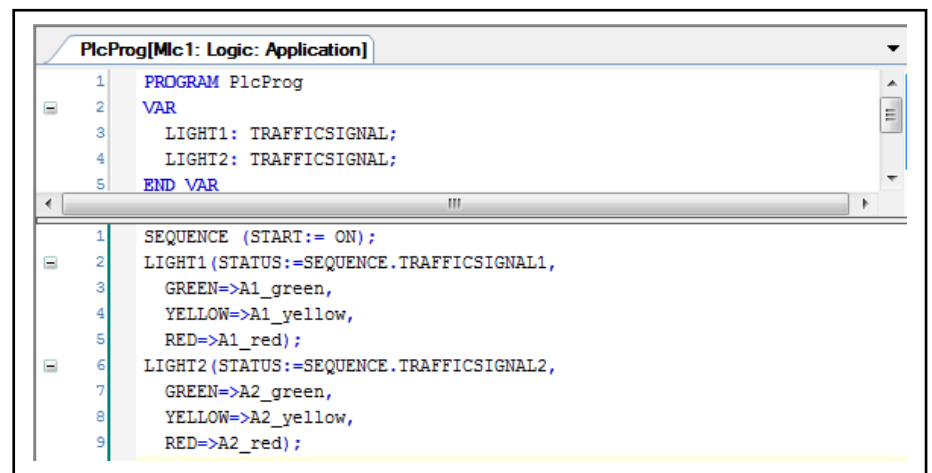


Fig.2-50: "PLCProg" program completed

Testing the Application in Simulation Mode, PlcProg in ST

Build application In order to load the application into the control, the application must first be compiled by the "compiler" with no errors. To compile the application, select **Create ► Build**. After compiling, the following is displayed in the "Task list" below the task window: "Compiling completed -- 0 errors, 0 warnings"



Only the application that is currently active is compiled.
If **several** applications are **available**, the application that is to be compiled must be set as active (context menu "Application" **Set Active Application**).

Simulating a control Since there may not be a control available for testing, it can be simulated. To simulate a control, in Project Explorer click on "Mlc1" and right click to open the context menu. Click on **Simulation** to activate simulation mode. If the simulation is active, "Simulation" is selected.

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

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

Switching on the traffic signal and monitoring the signal cycle After the application is started, "LIGHT1" is in "green" status and "LIGHT2" is in "red" status. The traffic signals must be switched on so that the signal cycle is processed. To switch the traffic signals on, proceed as follows:

1. Double click on the "GVL" in the Project Explorer to open the "Global Variable List".

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.2-51: "GVL" online

2. In variable "ON", select the "Prepared value" column so that "TRUE" is displayed there.
3. To write the value to the control, click on **Debug ► Write values** in the IndraWorks menu.
The signal cycle will be processed. The signal cycle can be seen in the "PlcProg" function block.

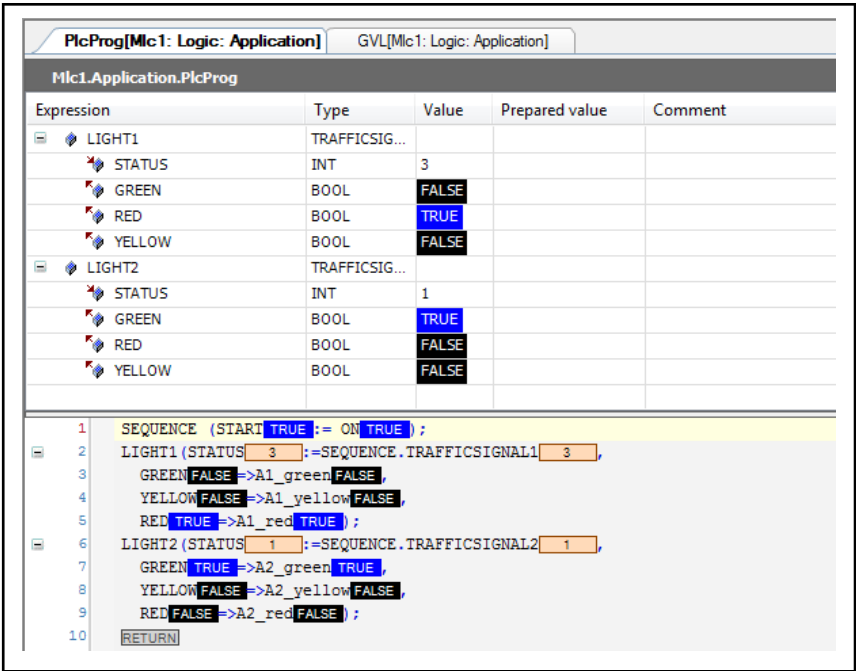


Fig.2-52: "PLCProg" program online

- Stopping an application
- To stop the PLC program, select **Debug ► Stop** in the IndraWorks menu.
- Logging out of the control
- To log out from the control, select **Debug ► Logout** in the IndraWorks menu.
- Editing "PlcProg" Program in "CFC"

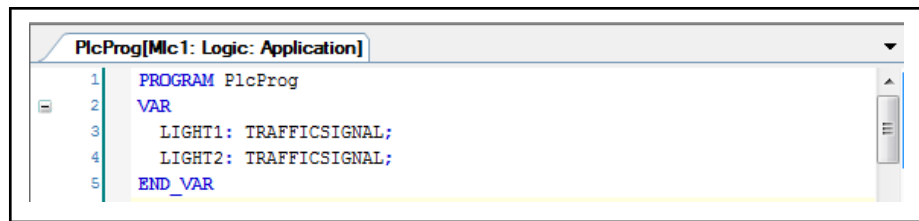


The implementation of the "PlcProg" program was alternatively offered in two programming languages:

- When programming in "Structured Text (ST)", refer to [Editing PlcProg Program in ST, on page 44](#),
- When programming in "Continuous Function Chart (CFC)", continue below

- Function of "PlcProg"
- The time sequence of the signal phases for both traffic signals was implemented in the "SEQUENCE" function block. The inputs and outputs of a PLC are to be controlled and evaluated. Input and output variables for the one input and six outputs for the two traffic signals 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, type "TRAFFICSIGNAL", and a "LIGHT2" instance, also "TRAFFICSIGNAL" type.
This instances the "TRAFFICSIGNAL" function block.
Create variable LIGHT1: TRAFFICSIGNAL; and LIGHT2: TRAFFICSIGNAL:

First Steps - IndraLogic

Fig.2-53: Declaration part POU *PlcProg*

Editing the body

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

1. In the "ToolBox" window, click on it to select "Box".

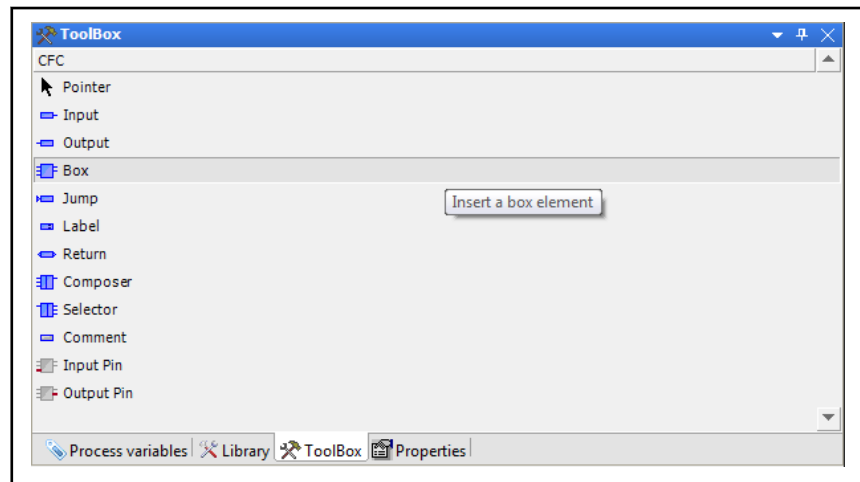


Fig.2-54: Tools – Element "function block"

2. Position the mouse in the body of the "PlcProg" Editor and left click to add the function block there.
3. Click on "???", the name of the function block and enter name the function block "SEQUENCE".

The FB looks like this:

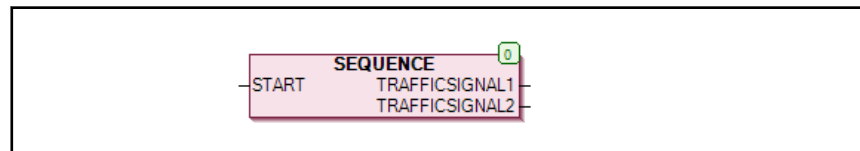


Fig.2-55: SEQUENCE function block

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. Position the mouse in the body of the "PlcProg" Editor and left click to add the function block there.
3. Click on "???", the name of the function block, and give the function block the type name "TRAFFICSIGNAL".
4. At "???" give the "TRAFFICSIGNAL" function block the instance name "LIGHT1".

The FB looks like this:

First Steps - IndraLogic

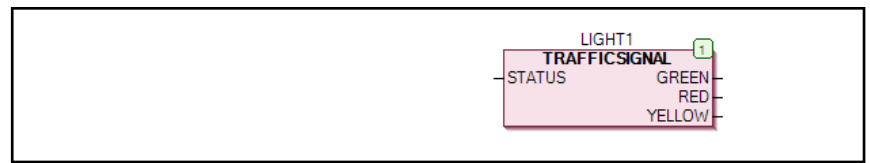


Fig.2-56: TRAFFICSIGNAL – LIGHT1 function block

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

1. In the "ToolBox" window, click to select the "Block (Box)" element.
2. Position the mouse in the body of the "PlcProg" Editor and left click to add the function block there.
3. Click on "???", the name of the function block, and name the function block "TRAFFICSIGNAL".
4. At "???" give the "TRAFFICSIGNAL" function block the instance name "LIGHT2".

The FB looks like this:

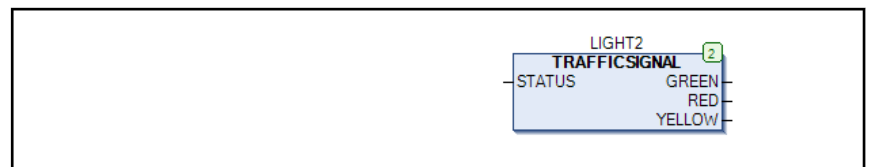


Fig.2-57: TRAFFICSIGNAL – LIGHT2 function block

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

1. In the "ToolBox" window, click to select the "Input" element.
2. Position the mouse in the body of the "PlcProg" Editor and left click to add the input there.
3. Click on "???", the name of the input, and name the input "ON".
4. In the "ToolBox" window, click to select the "Output" element.
5. Position the mouse in the body of the "PlcProg" Editor and left click to add the output there.
6. Add five more outputs in "PlcProg".
7. Assign the following names to 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 still uses the global variable of the same name as a replacement.

To complete the network, proceed as follows:

1. Left click the output for the "ON" element and hold the mouse key down. Move the mouse pointer to the "START" input of the "SEQUENCE" function block so that both elements are connected to each other.
2. Left click the "TRAFFICSIGNAL1" output for the "SEQUENCE" element and hold the mouse key down. Move the mouse pointer to the "STATUS" input of the "TRAFFICSIGNAL" function block so that both elements are connected to each other.

First Steps - IndraLogic

- Left click the "TRAFFICSIGNAL2" output for the "SEQUENCE" element and hold the mouse key down. Move the mouse pointer to the "STATUS" input of the "LIGHT2" function block so that both elements are connected to each other.
- Left click the "GREEN" output for the "LIGHT1" element and hold the mouse key down. Move the mouse pointer to the "A1_green" input of the output, so that both elements are connected to each other.
- Repeat the final step as needed for the other outputs.

The "PlcProg" program looks like this:

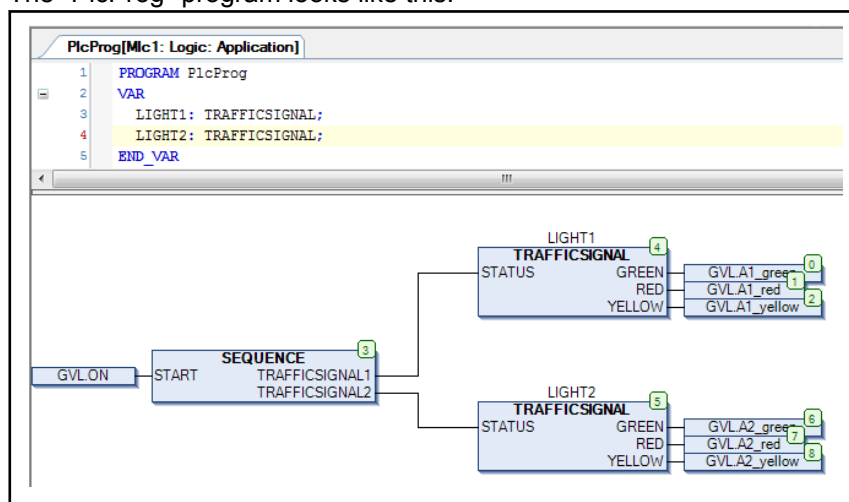


Fig.2-58: POU "PlcProg" completed



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

Testing the Application in Simulation Mode, PlcProg in CFC

Build application

In order to load the application into the control, the application must first be compiled by the "compiler" with no errors. To compile the application, select **Create ► Build Application**. After compiling, the following is displayed in the "Task list" below the task window: "Compiling completed -- 0 errors, 0 warnings: Ready for download!"



Only the application that is currently active is compiled.

If **several** applications are **available**, the application that is to be compiled must be set as active (context menu "Application" **Set Active Application**).

Simulating a control

Since there may not be a control available for testing, it can be simulated. To simulate a control, in "Project Explorer" click on "Mlc1" and right click to open the context menu. Click on **Simulation** to activate simulation mode. If the simulation is active, "Simulation" is selected.

Login to the control

Log into the control and load the application into the control. To log in on the control, select **Debug ► Login** in the IndraWorks menu. The new application can be loaded into the control either by changing it online or recompiling it.

Starting the application

To start the application, right click on the "Application" node and select **Start** in the context menu. The "Application" is now running as simulation.

Switching on the traffic signal and monitoring the signal cycle

After the application is started, "LIGHT1" is in "green" status and "LIGHT2" is in "red" status. The traffic signals must be switched on so that the signal cycle is processed. To switch the traffic signals on, proceed as follows:

- 1. Double click on the "GVL" in the Project Explorer to open the "Global Variable List".

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.2-59: "GVL" online

- 2. In variable "ON", select the "Prepared value" column so that "TRUE" is displayed there.
- 3. To write the value to the control, click on **Debug ► Write values** in the IndraWorks menu.

The signal cycle will be processed. The signal cycle can be seen in the "PlcProg" function block.

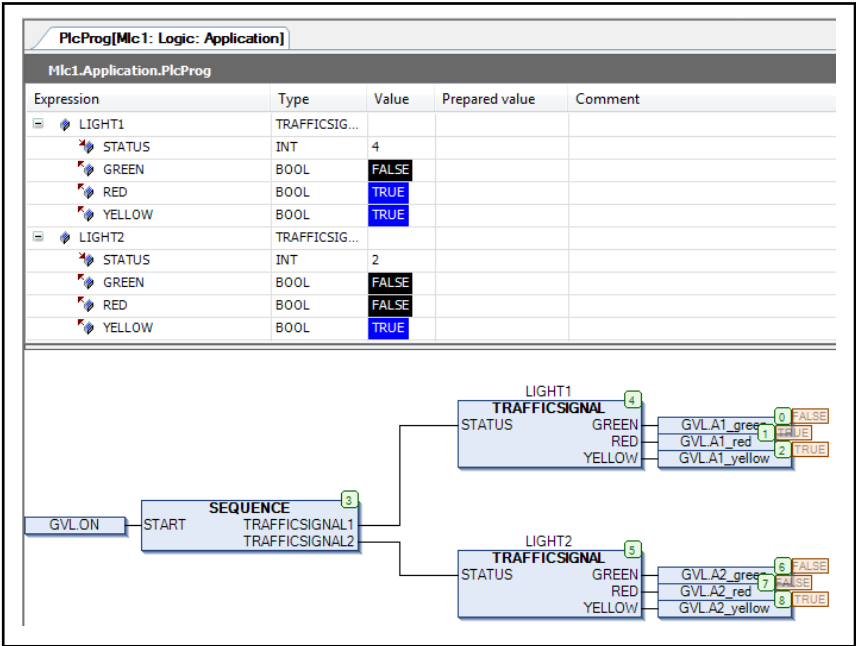


Fig.2-60: "PlcProg" program online

Stopping an application

To stop the PLC program, select **Debug ► Stop** in the IndraWorks menu.

Logging out of the control

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

2.6.8 Testing the Application on the Control

Deactivating simulation mode

If a control is available, the application should be tested with it.
To do so, the [Communication Settings in the project, on page 16](#), must be consistent with the control required.

First Steps - IndraLogic

End simulation mode by right-clicking in the "Project Explorer" on the "Mlc1" control to open the context menu. Click on **Simulation** to deactivate simulation mode. If simulation is no longer active, "Simulation" is deselected.

Build application

In order to load the application into the control, the application must first be compiled by the "compiler" with no errors. To compile the application, select **Create ► Build** in the IndraWorks menu. After compiling, the following is displayed in the "Task list" below the task window: "Compiling completed -- 0 errors, 0 warnings"



Only the application that is currently active is compiled.

If **several** applications are **available**, the application that is to be compiled must be set as active (context menu "Application" **Set Active Application**).

Login to the control

Log into the control and load the application into the control. To log in on the control, select **Debug ► Login** in the IndraWorks menu.

The application is then in "STOP" state.

Starting the application

To start the application, right click on the application and select **Start** in the context menu. The application is running on the control.

Generating a boot application

To create a boot project using the menu bar, select **Debug ► Generate boot application** in the IndraWorks menu. If this command is executed online, the compiled project will be stored in the control so that it loads automatically at the next start up.

Switching on the traffic signal and monitoring the signal cycle

After the application is started, "traffic signal 1" is in "green" status and "traffic signal 2" is in "red" status. The traffic signals must be switched on so that the signal cycle is processed. To switch the traffic signals on, proceed as follows:

1. Double click on "GVL" in the Project Explorer to open the Global Variable List.

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.2-61: "GVL" online

2. In variable "On", select the "Prepared value" column so that "TRUE" is displayed there.
3. To write the value to the control, click on **Debug ► Write values** in the IndraWorks menu.

The signal cycle will be processed.

The processing of the traffic signal cycle can be followed in the individual program POU's (PlcProg, SEQUENCE) and the Global Variable List (GVL).

Open the TRAFFICSIGNAL function block. In a selection window, two options can be selected: "PlcProg.LIGHT1" or "PlcProg.LIGHT2".

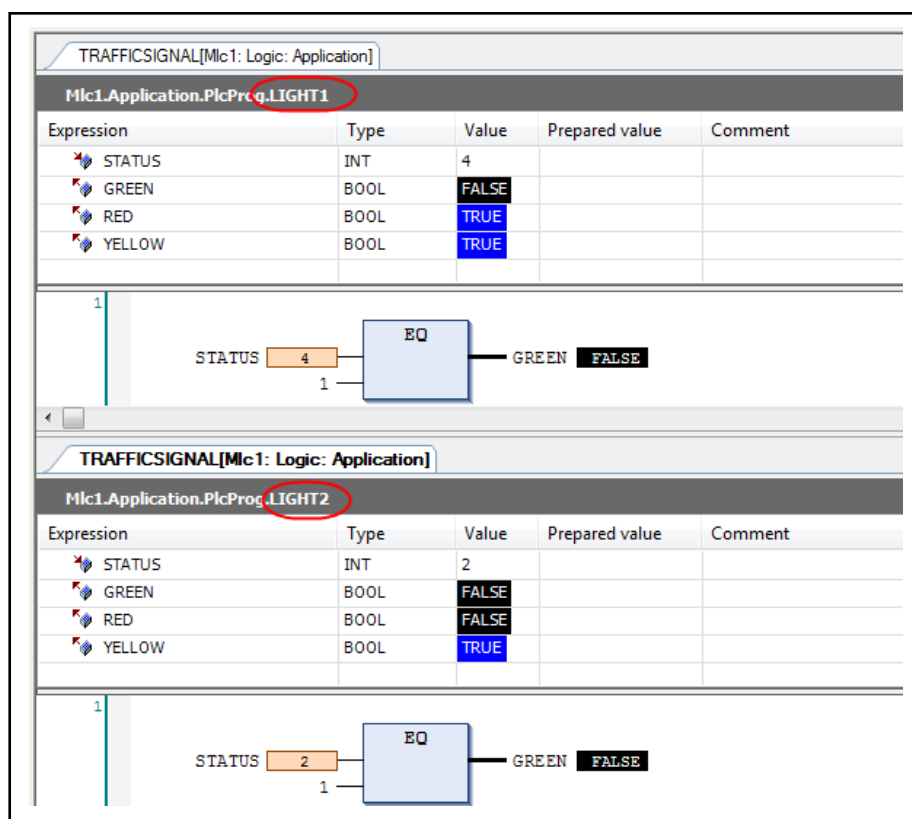


Fig.2-62: Status display in the two function block instances of TRAFFICSIGNAL

Control of real inputs and outputs

The IndraMotion MLC / IndraLogic XLC L45 and L65 controls have onboard outputs that are assigned to the addresses %QX0.0 ... %QX0.7 (mapping). The signal cycle can also be followed on these outputs.

Stopping an application

To stop the PLC program, select **Debug ► Stop** in the IndraWorks menu.

Logging out of the control

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

2.7 Creating Visualization for the PLC Program

2.7.1 Visualization in IndraLogic

With IndraLogic visualization, visual depictions of dynamic procedures can be created easily and quickly. The status of the variables is queried at defined intervals and the corresponding graphic elements are displayed according to their respective status. Here, the PLC program described in ["Creating a PLC program" page 20](#), is visualized. The following two signals for the programmed traffic lights and an ON switch for visualization are to be visualized and their corresponding status displayed.

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 onto the application.

The following window opens:

First Steps - IndraLogic

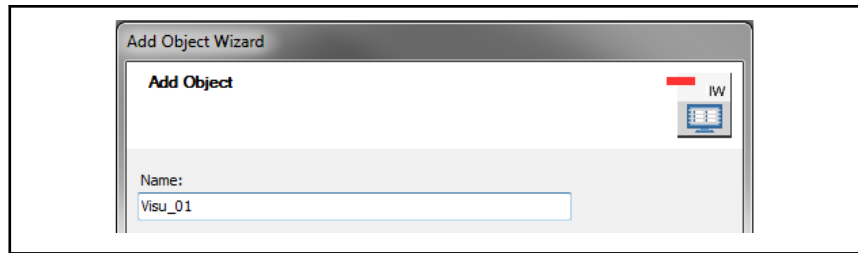


Fig.2-63: Adding a visualization to the application

2. Name the visualization as desired and confirm with **Finish**.

Visualization is added together with an instance of the Visualization Manager:

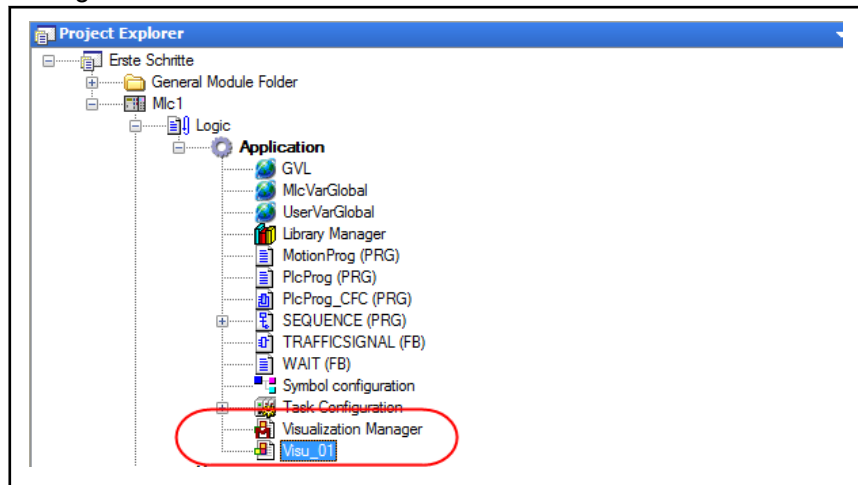


Fig.2-64: Visualization added to the application

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

2.7.3 Editing Visualization

Adding and Editing the First Element "Traffic Signal"

Opening the visualization editor Open the visualization editor by double clicking on the visualization in the Project Explorer.

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

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

Five folders in the ToolBox are offered:

- **Basis:** general visualization objects
- **General control elements:** Interfaces, check boxes, progress bar...
- **Measuring devices:** Progress displays, pointer instruments, potentiometer, histogram...
- **Lamp, switches. Images:** Lamps, rotary switch and toggle switch, image slider...
- **Special control elements:** Trace objects, wait symbols...

Adding an "ellipse" element

The first element of the "traffic signal", "Light", is to be added into the visualization. Elements that can be added to a visualization are located in the "Tool-Box" window (see the following figure).

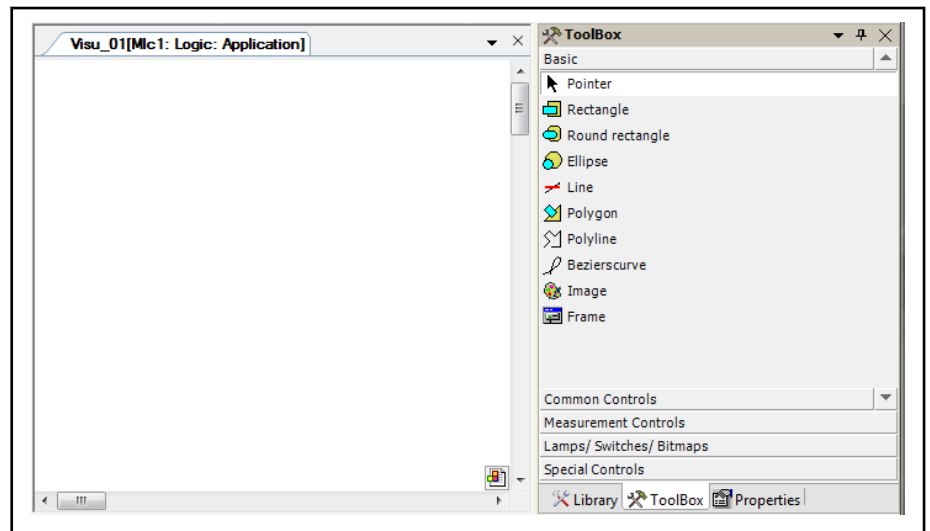


Fig.2-65: "ToolBox" window

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

1. In the "ToolBox" window, left click to select the "ellipse" element.
2. Move the mouse pointer to the visualization window and left click again.

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

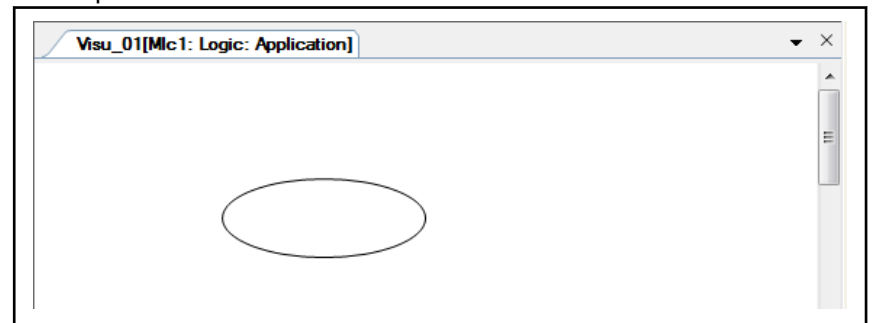


Fig.2-66: Visualization editor - ellipse

Changing the form of the ellipse

Here, the shape of the "ellipse" you added is to be changed to match that of a "traffic signal".

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

1. Highlight the "ellipse" by left clicking on the "ellipse".
2. Open the "Properties" window by left clicking on the "Properties" tab.

First Steps - IndraLogic

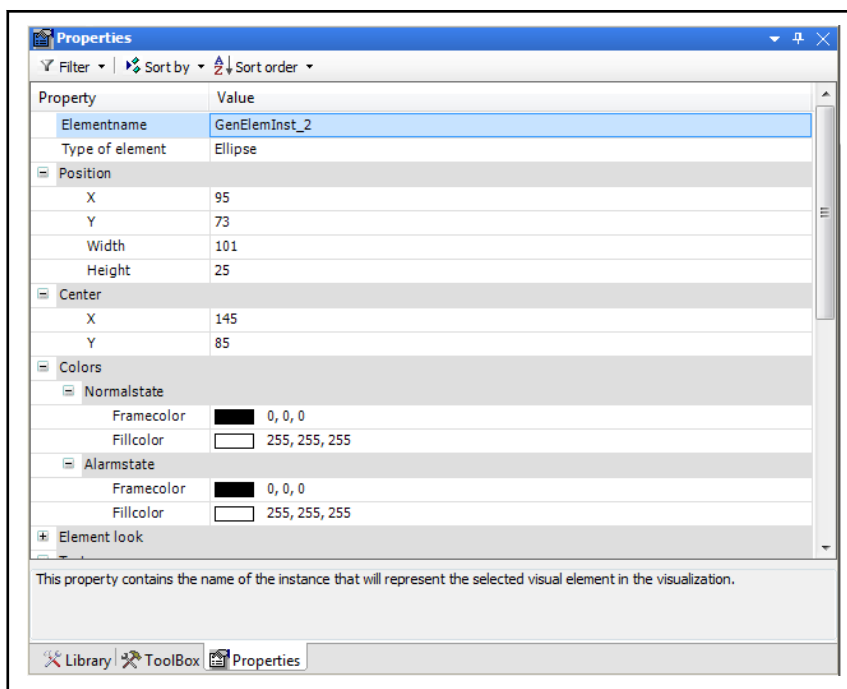


Fig.2-67: "Properties" window of the ellipse

- To change the (X, Y) position and the size (width, height), select the following under the "Position" property:

X = "60"

Y = "150"

Width = "30"

Height = "30"

- To change the center point of the "ellipse", select (X, Y) under "Center":

X = "75"

Y = "165"

Changing the properties of the traffic signal

Here the traffic signal's fill color is to be changed. The traffic signal should be gray when it is inactive and red when the signal light is active.

To change the fill color, proceed as follows:

- To change the fill color for the normal state, under the "Colors" property in "Normal state" under "Fill color", click on the ... button.

The "Color" window opens.

- In the "Color" window, click on **FillColor**. Select "Silver".

- To change the fill color for the alarm state, under the "Colors" property in "Alarm state" under "Fill color", click on the ... button.

The "Color" window opens.

- In the "Color" window, click on **FillColor**. Select "red".

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

Proceed as follows:

First Steps - IndraLogic

1. To communicate the variable for changing the signal color to the visualization, under "Color variables" in "ToggleColor" double click in the "Value" column.
2. Click on the ... button.
The "Input assistance" window opens.
3. "Application", "GVL" can be seen under "Elements" in the "Input Assistant" window.
4. Under "GVL variables" select the "A1_red" variable.
The "GVL variable" is added to "ToggleColor":

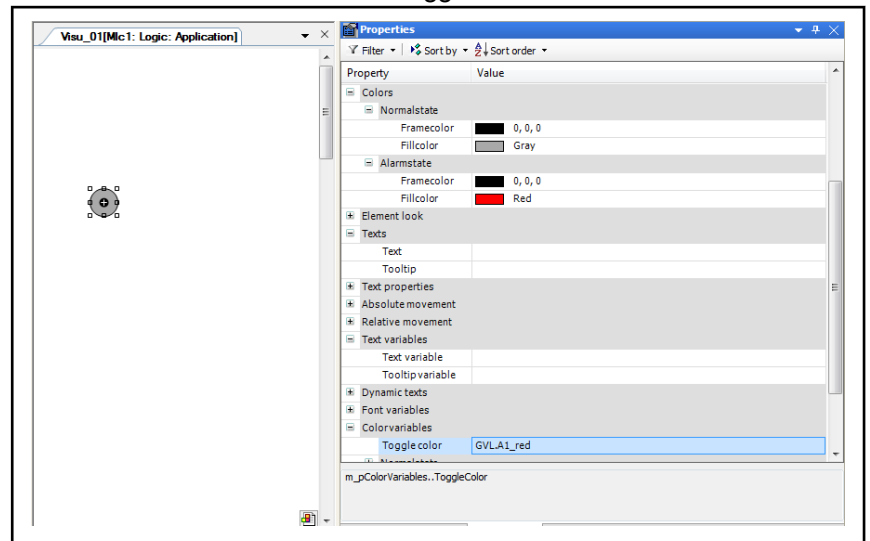


Fig.2-68: Properties - ToggleColor

Copying, Adding and Editing the "Traffic Signal" Element

Copying and inserting the "traffic signal" element

The "red" signal for the first signal lamp is now complete. The first traffic signal only needs signal lights in the signal colors "yellow" and "green". Since the two other traffic signals have the same shape as the first, the easiest method is to copy the first traffic signal.

To copy and insert the traffic signal, proceed as follows:

1. Select the traffic signal.
2. Use the right mouse button to open the context menu and select **Copy**.
3. Click in the visualization area and select **Insert** in the context menu.
4. To add another traffic signal, click again in a free area in the visualization editor and select **Insert** in the context menu.
5. Select the first traffic signal added.

Changing the position of the "traffic signal" element

Both of the traffic signal added must now be repositioned.

To reposition the traffic signals, proceed as follows:

1. Select the first traffic signal added.
2. In the "Properties" window under "Position", select for (X, Y) = (60, 195) to change the position.
3. To change the center point, select for (X, Y) = (75, 210) under "Center".
4. Select the second traffic signal added.

First Steps - IndraLogic

**Changing the properties of the
"traffic signal" element**

5. In the "Properties" window under "Position", select for (X, Y) = (60, 240) to change the position.

6. To change the center point, select for (X, Y) = (75, 255) under "Center".

Here both of the traffic signal are assigned new colors for the "Alarm state".

Proceed as follows:

1. Select the traffic signal in the center.
2. To change the fill color for the alarm state, under the "Colors" property in "Alarm state" under "Fill color", click on the ... button.
3. The "Color" window opens.
4. In the "Color" window, click on **FillColor**. Select "Yellow".
5. Repeat steps 1 - 3 for the other traffic signals.
6. In the "Color" window, click on **FillColor**. Select "Green".

You still have to adapt the variables for addressing the various traffic signal colors.

Proceed as follows:

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

Click on the ... button.

The "Input assistance" dialog opens.

3. "Application", "GVL" can be seen under "Elements" in the "Input Assistant" window.
4. Under "GVL variables" select the "A1_yellow" variable.
5. Select the lower traffic signal.
6. Under "GVL variables" select the "A1_green" variable.

Adding and Editing the "Traffic Signal Housing" Element**Elements of the traffic signal housing**

The three lights for displaying the signaling conditions are now complete. The traffic signal's appearance must be adapted so it looks like a traffic signal. The following three elements are added to the visual display of the traffic signal:

1. the housing for the signal lights.
2. the mast for the traffic signal housing.
3. the traffic signal base.

Adding an "rectangle" element

To add the traffic signal housing, proceed as follows:

1. In the "ToolBox" window, select the "rectangle" element.
2. Click in a free area of the visualization editor to insert the "rectangle" element.

Changing the position of the "traffic signal housing" element

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

1. Select the "rectangle" element added and open the "Properties" window.
2. In the "Properties" window under "Position", select (X, Y) = (50, 135) to change the position.
3. In the "Properties" window under "Position", select (width, height) = (50, 150) to change the position.
4. To change the center point, select (X, Y) = (50, 210) under "Center".

Changing the properties of the "traffic signal housing" element

5. Right click next to the selected object to open the context menu. Select **Sequence ► Send to back**.

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

1. In the "Properties" window under "Colors", "Normal state" and "FillColor" on the ... interface.

The "Color" window opens.

2. In the "Color " window, click on the **Define color** button.
3. In the "Red", "Green" and "Blue" fields, enter the following:
(Red, Green, Blue)= (5, 5, 10).

Adding and Editing "Traffic Signal Post" Element

Adding an "rectangle" element

Next, the traffic signal post is constructed.

Proceed as follows:

1. In the "ToolBox" window, select the "rectangle" element.
2. Click in a free area of the visualization editor to insert the "rectangle" element.

Changing the position of the "traffic signal post" element

1. Highlight the "rectangle" element you added and open the "Properties" window.
2. In the "Properties" window under "Position", select for (X, Y) = (70, 285) to change the position.
3. In the "Properties" window under "Position", select (width, height) = (10, 195) to change the position.
4. To change the center point, select for (X, Y) = (75, 380) under "Center".

Changing the properties of the "traffic signal post" element

Select dark gray (DimGray) (Colors, Normal state, FillColors, (Red, Green, Blue) = (105, 105, 105)) as the fill color for the traffic signal post.

Adding and Editing "Traffic Signal Base" Element

Adding an "rectangle" element

The first traffic signal still needs a base.

To construct the traffic signal base, proceed as follows:

1. In the "ToolBox" window, select the "Rectangle" element and add the element to the working area.
2. Highlight the "rectangle" element you added and open the "Properties" window.
3. In the "Properties" window under "Position", select (X, Y) = (15, 480) to change the position.
4. In the "Properties" window under "Position", select (width, height) = (120, 30) to change the position.
5. To change the center point, select for (X, Y) = (75, 495) under "Center".

Changing the properties of the "traffic signal base" element

Select gray (DimGray) (Colors, Normal state, FillColors, (Red, Green, Blue) = (105, 105, 105)) as the fill color for the traffic signalbase.

First traffic signal:

First Steps - IndraLogic

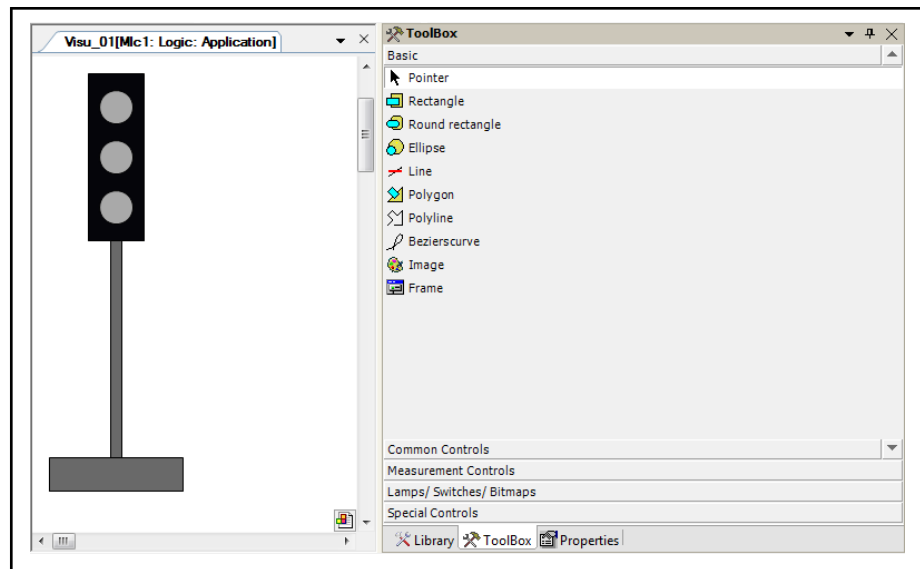


Fig.2-69: First traffic signal

Adding and Editing Two Traffic Signals

The second traffic signal

To create the second traffic signal, select all traffic signal elements. Left click in an area below and to the right of the traffic signal and keep the mouse button pressed. Move the mouse pointer to a position to the left above the traffic signal. Release the mouse button. All of the elements are now selected.



Alternatively, select individual elements by pressing and holding the <Ctrl> key when selecting the desired elements.

In the context menu, select **Copy** and **Insert**. Store the selected object next to the first traffic signal.

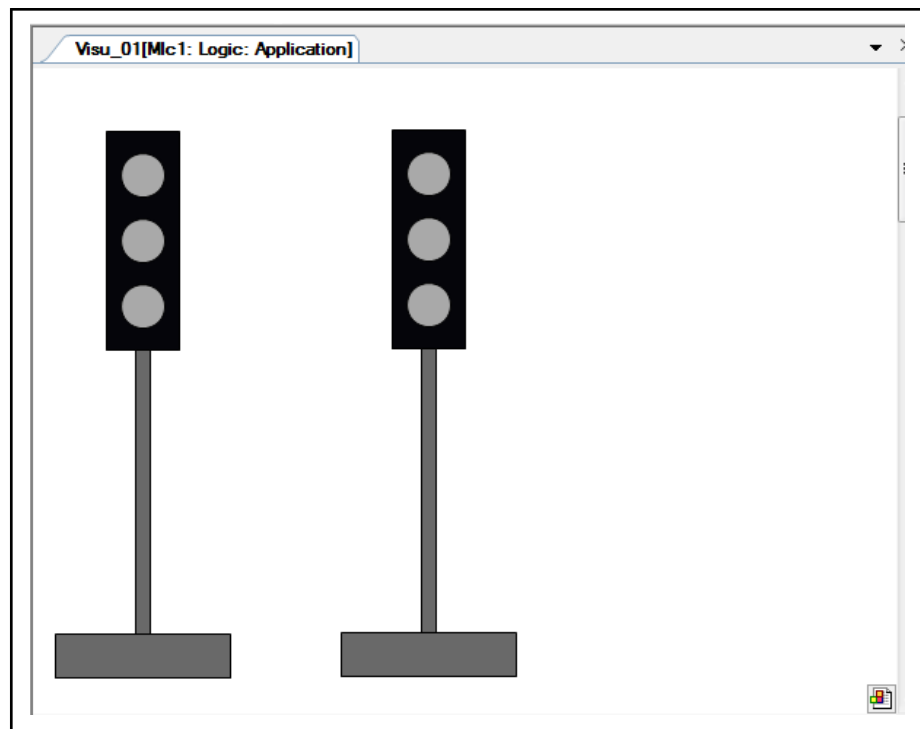


Fig.2-70: Two traffic signals

You still have to adapt the variables for switching among the various signal 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 an ON Switch

Adding a "button" element

It should be possible to switch the traffic signal on at the beginning. A "button" has to be inserted:

1. In **ToolBox ► General control elements**, select the "button" element.
2. Click in a free area of the visualization editor to insert the "button" element.

Changing the position of the "button" element

1. In the "Properties" window under "Position", select (X, Y) = (130, 55) to change the position.
2. In the "Properties" window under "Position", select (width, height) = (70, 40) to change the position.
3. To change the center point, select for (X, Y) = (165, 75) under "Center".

Changing the properties of the "button" element

1. In the "Texts" property under "Text" enter "ON".
2. In the "Input configuration" property under "OnMouseClicked" click on "Configure".

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. Use the **OK** button to confirm the entry.

Adding and Editing Labeling

Adding an "rectangle" element

Now both traffic signals need to be labeled as "Traffic signal 1" and "Traffic signal 2".

To label the traffic signals, proceed as follows:

1. In the "ToolBox" window, select the "rectangle" element.
2. Click in a free area of the visualization editor to insert the "rectangle" element.

Changing the position of the "rectangle" element

1. Highlight the "rectangle" element you added and open the "Properties" window.
2. In the "Properties" window under "Position", select (X, Y) = (15, 520) to change the position.
3. In the "Properties" window under "Position", select (width, height) = (120, 30) to change the position.
4. To change the center point, select (X, Y) = (75, 535) under "Center".

Changing the properties of the "rectangle" element

1. In the "Text" field next to "Texts" enter "Traffic signal 1".
2. In the "Properties" window under "Colors", "Normal state" and "Frame Color" on the ... interface.

The "Color" window opens.

First Steps - IndraLogic

Copying and adapting the "rectangle" element

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

1. Copy the label and store it under the second traffic signal.
2. In the "Text" field next to "Texts" enter "Traffic signal 2".

The visualization is complete:

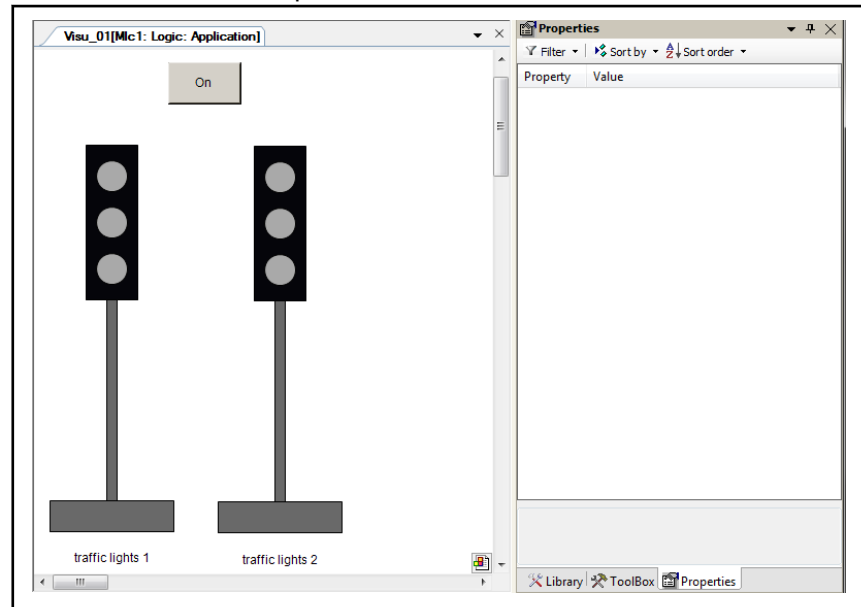


Fig.2-71: Completed visualization in visualization editor

Displaying Variable Values

Adding an "rectangle" element

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

The two traffic signals have to be assigned a display of their "STATUS" variable under the labels "Traffic signal 1" and "Traffic signal 2".

To label the traffic signals, proceed as follows:

1. In the "ToolBox" window, select the "rectangle" element.
2. Click in a free area of the visualization editor to insert the "rectangle" element.
3. Select the "rectangle" element added and open the "Properties" window.
2. In the "Properties" window under "Position", select (15, 570) for (X, Y) to change the position.
3. In the "Properties" window under "Position", select (width, height) = (120, 30) to change the position.
4. To change the center point, select (X, Y) = (75, 585) under "Center".

Changing the position of the "rectangle" element

Changing the properties of the "rectangle" element

1. In the "Text" field under "Texts", enter "Traffic signal 1: %s".
2. Click in the "Text Variable" field on "Text variables" and select the variable "PlcProg.LIGHT1.STATUS" via

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

Copying and adapting the "rectangle" element

1. Copy the label and store it under the second traffic signal.
2. For display 2, replace:

First Steps - IndraLogic

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

Visualization with completed display:

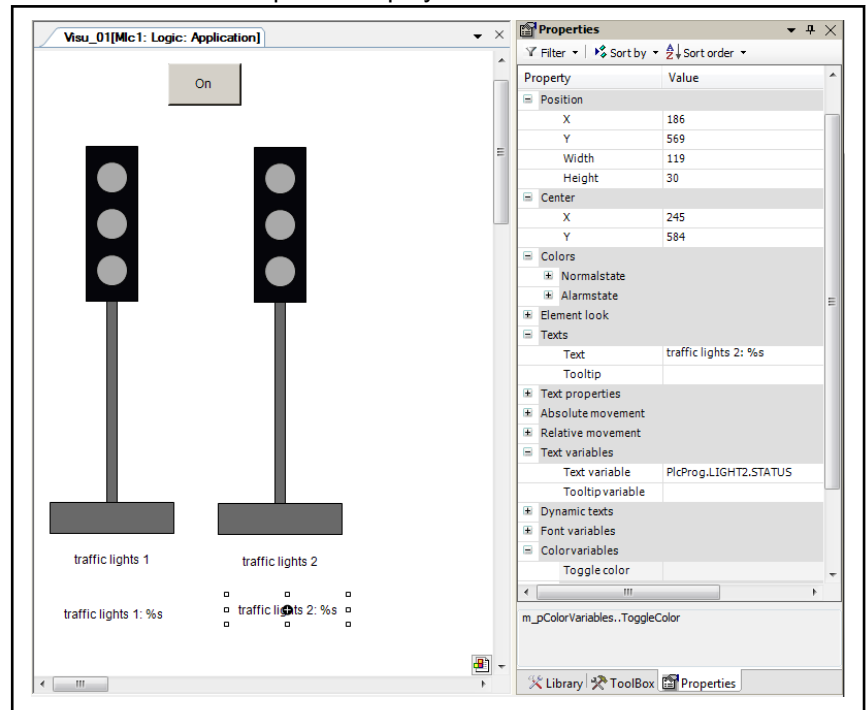


Fig.2-72: Visualization Editor completed by display

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 visualization in Simulation mode, refer to ["Testing the Application in Simulation Mode \(PlcProg in ST\)"](#), on page 46 or ["Testing the Application in Simulation Mode \(PlcProg in CFC\)"](#), on page 50.

To test the visualization with a control, refer to ["Testing the application with a control"](#), page 51.

When the application has been started, open the "Visualization" window:

First Steps - IndraLogic

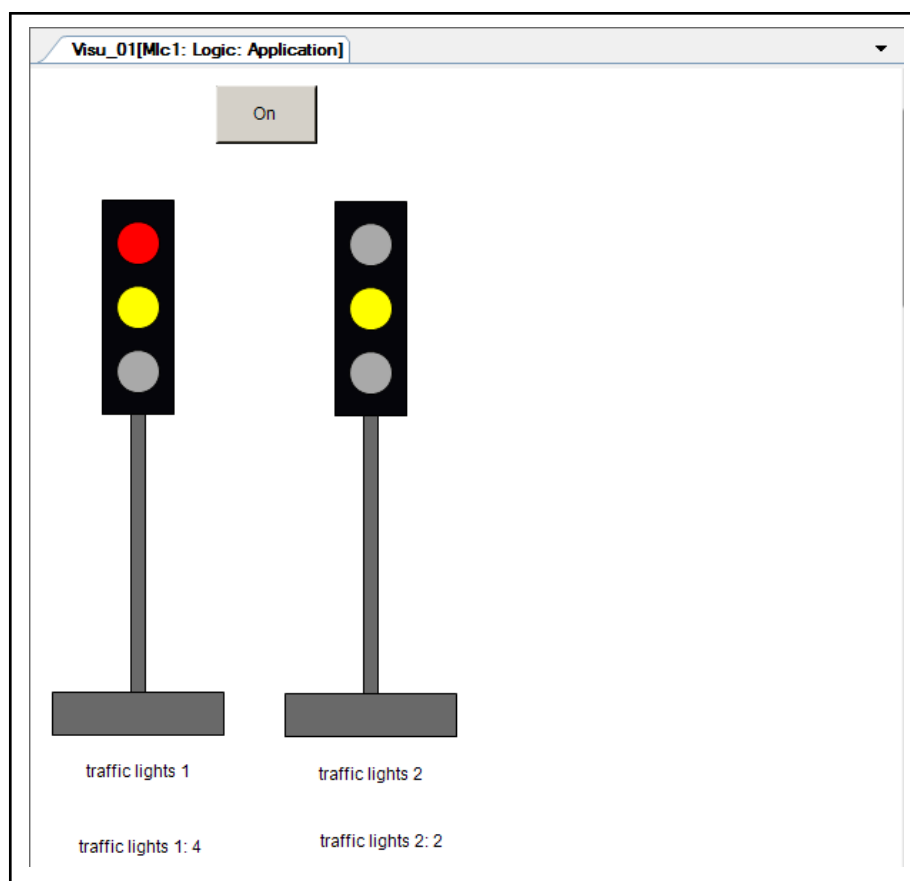


Fig.2-73: Online visualization

Click on **ON**. The traffic signal cycles are processed.

2.8 Restoring a Project

Location Based on the project created, an export file was generated that enables the project to be restored independently of its control hardware (XLC L25, L45, L65 (with or without sercos III bus) or MLC L25, L45, L65). The restored project can run as a simulation or using its control.

The export file has been created for the "Application" node and contains the following components:

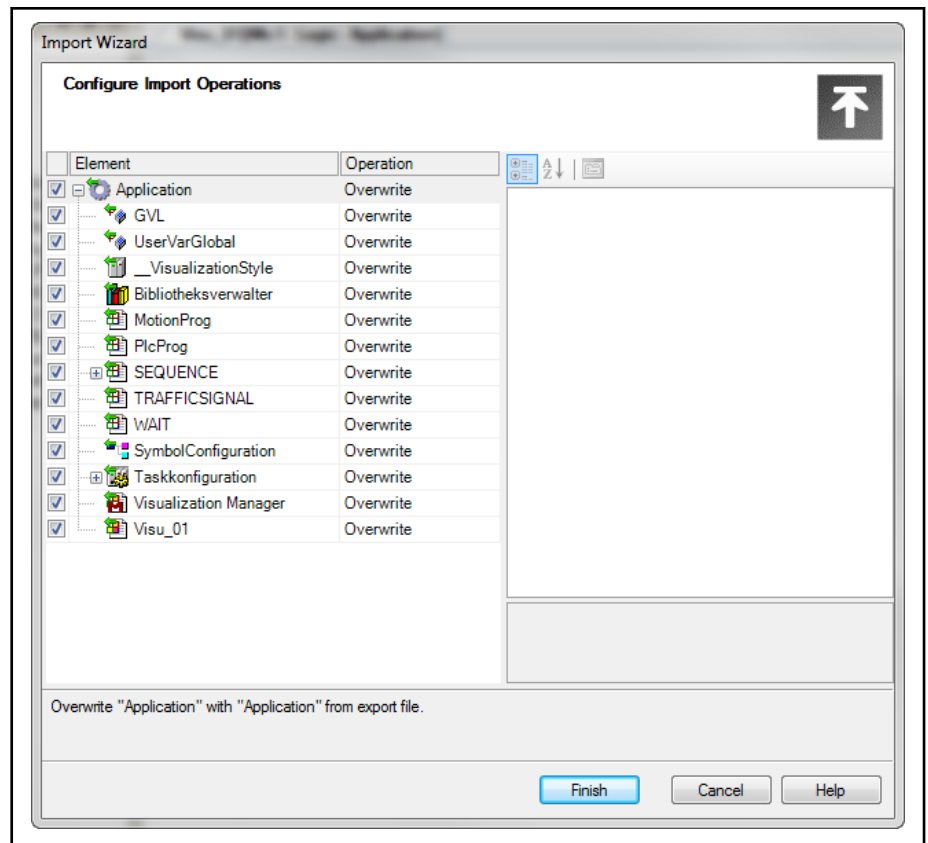


Fig.2-74: Components of the export file

The given components are overwritten during import.

Restore

To ensure that the project runs on a control (IndraLogic XLC L25, L45, L65 (with or without sercos III bus) or IndraMotion MLC L25, L45, L65) proceed as follows:

1. Clear old data in the control
Clear the old memory content of the control.
Details are described under [Resetting the Control \(CMLx5\)](#), page 14,.
The memory of the control is cleared and any possible Motion components present and the PLC components are ready for operation.
2. Create a new project
Create a new empty project (**File ► New ► Project**). The project name can be chosen at random, although "First Steps" is recommended.
A new empty project has been created.
3. Create the control
Drag a control of your choice (XLC L25, L45, L65 (with or without sercos III bus) or IndraMotion (MLC L25, L45, L65) from the Device Library to the empty project: The control name can be chosen at random, although "Mlc1" is recommended.
Complete device type and IP address.
Execute a connection test (not necessary if only "simulation" is required).
End adding as usual.

First Steps - IndraLogic

The desired control is now contained in the project.

4. Import the export file "Application_FirstSteps_IndraLogic_Basics.iwx"
In the Project Explorer, expand the control node to **Logic ► Application**.
Select **Import...** in the context menu of the "Application" node.
Select the export file "Application_FirstSteps_IndraLogic_Basics.iwx" in "Drive:\All Programs\Rexroth\IndraWorks\Projects". in the context menu opens the "Configure import actions" dialog. Select "Overwrite" for each element in the import dialog. The import is started using **Finish**.
The required files have been copied, the project is in Simulation mode and can be run on the control.

Testing a project



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

To test the project in Simulation mode, refer to ["Testing the Application in Simulation Mode \(PlcProg in CFC\)", on page 46](#) or ["Testing the Application in Simulation Mode \(PlcProg in ST\)", on page 46](#).

To test the project with a control, refer to ["Testing the application with a control", page 51](#).

3 First Steps - Motion

3.1 Project Goal "First Steps - Motion"

A virtual axis that is supposed 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" can be 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).

- [Programming a Motion Sequence in FBD on page 78](#)
- [Programming a Motion Sequence in Structured Text \(ST\) on page 100](#)

3.2 Technical Requirements

- Software:
 - PC with installed IndraWorks Engineering for IndraMotion MLC 13VRS
- Control:
 - IndraMotion MLC L25
 - IndraMotion MLC L45
 - IndraMotion MLC L65

Meeting these prerequisites is described in:

- "Rexroth IndraMotion MLC13VRS Functional Description", chapter "Commissioning and Operation"

A control, e.g. of type "IndraMotion MLC L65.1" or "IndraLogic XLC L65.1, is **possible with sercos III interface**", with current firmware 13VRS has to be available. 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:
- In case of 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

First Steps - Motion

- 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> located below the display and when **BOOT 1.02** is displayed and keep them pressed until **BOOTSTOP** is displayed.

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

Confirm with <Enter>. **CLEANUP** and subsequently **REBOOT** is displayed.

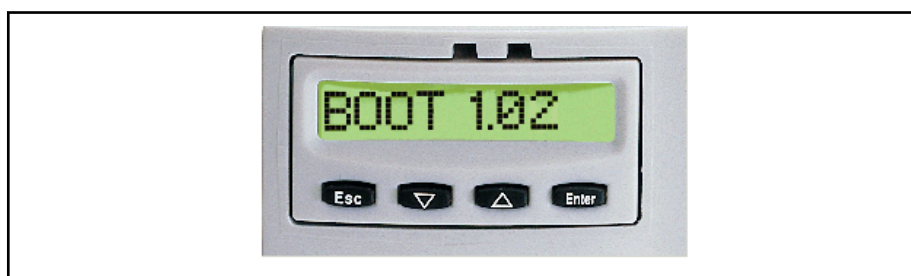


Fig.3-1: Rebooting the control



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

The control runs to **P2 RDY**. The memory is cleared. The Motion component, if available, is in phase P2 and the PLC component is ready for operation.

3.3.2 Starting IndraWorks

Start IndraWorks via **Start ▶ Programs ▶ Rexroth ▶ IndraWorks ▶ Engineering**.

The IndraWorks engineering environment is started. Alternatively,

IndraWorks can also be started via a desktop icon .

3.3.3 Creating a New Project

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

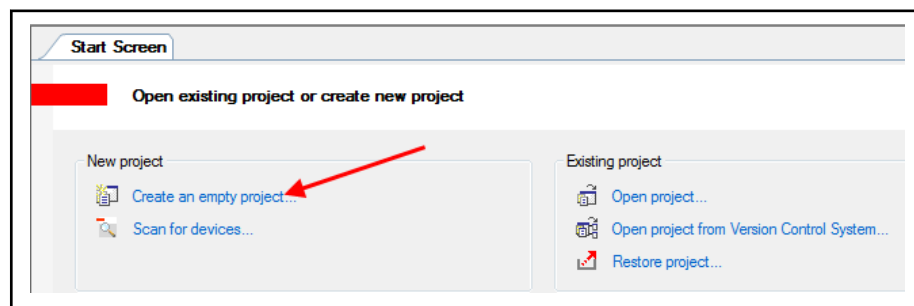


Fig.3-2: Creating an empty project

Alternatively, a new project can also be created via **File ▶ New ▶ Project....**

First Steps - Motion

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

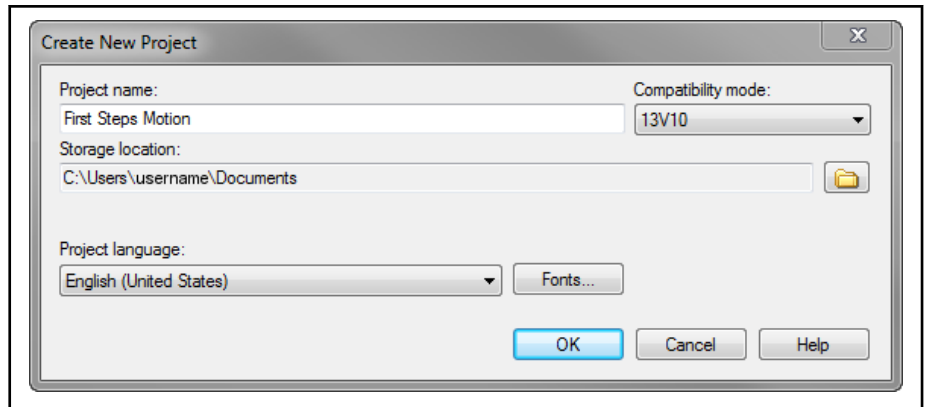


Fig. 3-3: Settings to save a new IndraWorks Project

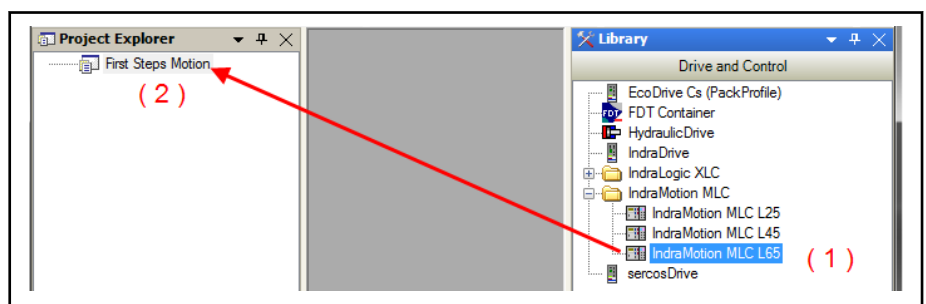
- Project name** Specify a name for the project. Suggestion: "First Steps Motion".
- Storage location** The field contains a storage location in the directory "C:\Documents and Settings\username\MyDocuments" by default.
A different directory can also be specified.
- Compatibility mode** The IndraWorks interface scope corresponds to version 13V10.
- Project language** By 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 names:



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

3.3.4 Adding a New Device to a Project

Drag the desired device (e.g. "IndraMotion MLC L65") from the "Drive and Control" library to 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 as described below to make the necessary basic settings.

3.3.5 Specifying Basic Settings for the Device

Device Settings

Enter the general information on the device to be added.

First Steps - Motion

Fig.3-6: Device settings

Device name At "Device Name", enter the desired device name, e.g. "Mlc1". By default, the field contains the name from the library. The device is displayed later in the Project Explorer under the name specified here.

Comment Add a comment that describes the device in detail. This field is optional.

Created by 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: Configuration

First Steps - Motion

Device type	The device type is predetermined by selecting the control. If there are several different variants for one device type, select the device variant from the drop-down list (example: IndraLogic XLC Lx5 is available with sercos III bus (CMLx5.1-3P) and without sercos III bus (CMLx5.1-NP, IndraMotion MLC Lx5 is always available with sercos III bus (CMLx5.1-3P)).
Firmware version	Select the firmware version here. This setting can generally remain.
Firmware release	The firmware release within a firmware version is set here. This setting should also remain.
IP address	Enter the target address in "Device IP address", i.e. the IP address of the control where the application should be loaded.
PLC Gateway	In "Gateway IP address", define the address of the computer where the gateway server is running. By default, "localhost" is offered as a computer name (address), "localhost" means that the locally installed gateway is addressed. This setting is not to be changed.
PLC communication	Recommended: Communication via TCP protocol. Communication via UDP is only recommended for SafeLogic.
Connection test	Click on the Execute button to check whether the communication between computer and target control is successful. The test result is recorded 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: • 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 (see Rexroth IndraMotion MLC 13VRS Functional Description, chapter "Commissioning and Operation", subchapter "Setting the IP address") – The network mask settings of the PC and/or control is too small Solution: Change the IP address in the dialog or the IP address of the control (see Rexroth IndraMotion MLC 13VRS 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 Rexroth IndraMotion MLC 13VRS Functional Description, chapter "Control", subchapter "Firmware Management") – The selected control does not correspond to the real control type
Secure online mode	If this option "Safe online mode" is checked, a prompt is displayed asking for confirmation before any security-relevant actions are executed.

First Steps - Motion

PLC programming language

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

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

Function Module Settings



Depending on the device type, the "Interfaces" dialog may differ. In this project, the dialog of the "IndraMotion MLC L65" device is described as example.

Enter the desired device and function module settings.

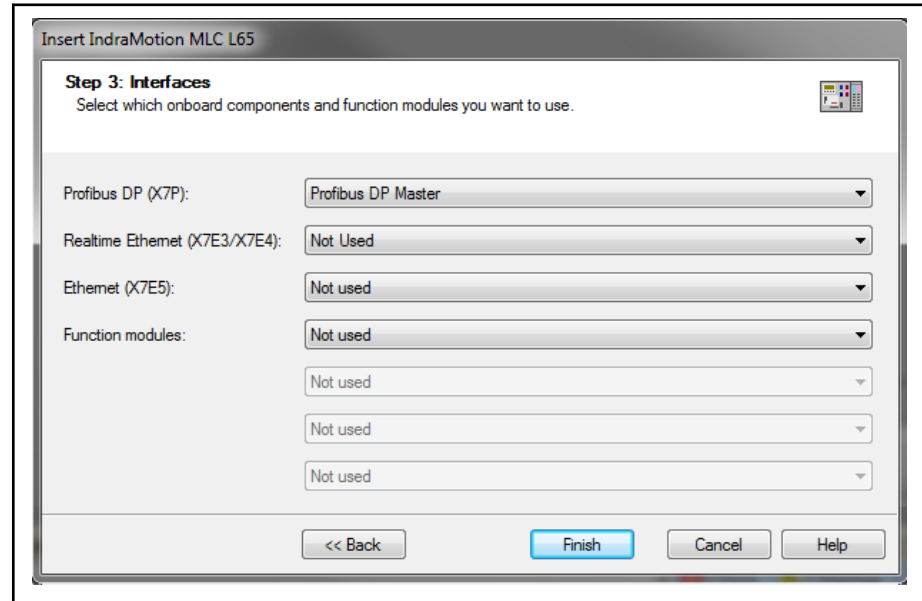


Fig.3-8: Function module settings

sercos (X7E1/X7E2)



This option is only available for IndraLogic XLC controls with sercos interface.

Current sercos III configuration that can be selected when the device is added. The selection includes "sercos III master" or "Not used".

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 an Engineering interface (connection directly to your PC or via the operating network), the Ethernet/IP Adapter cannot be checked.

Function modules

Up to four function modules can be connected to the "IndraMotion MLC L65". If required, select the function modules from the drop-down list.

Confirm the entries with **Finish**.

The newly created device is displayed in the Project Explorer.

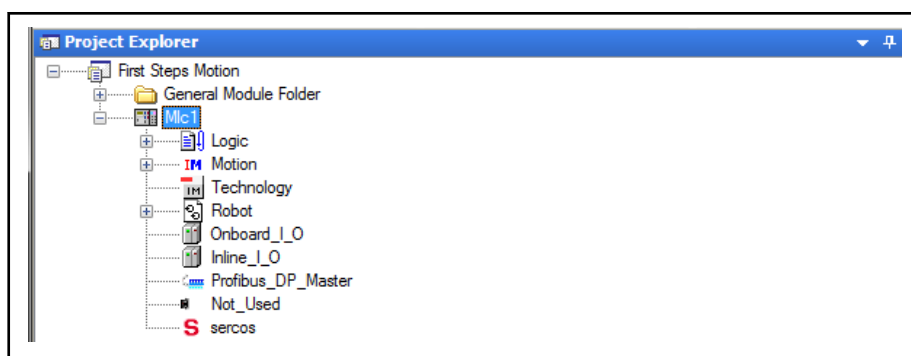


Fig. 3-9: Project Explorer: "First Steps" project with an IndraMotion MLC L65 control with sercos III bus



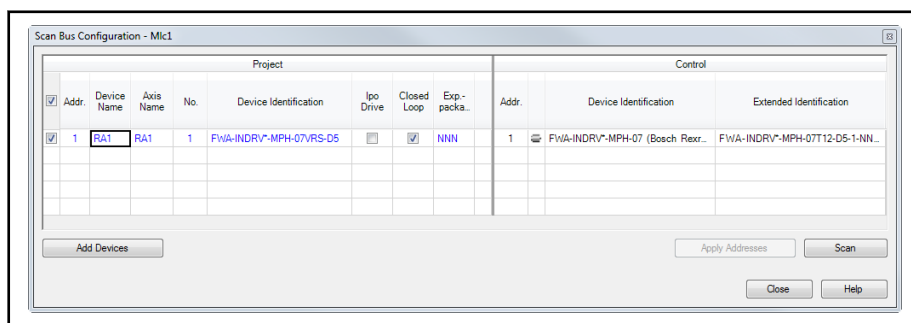
The Project Explorer reflects the components selected in the "Configuration" and "Function modules" dialogs.

3.3.6 Adding sercos III Drives (Real Axes) to the Project

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

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

The result is shown below:



Addr. sercos address
Device name Default drive_n, renamed
Axis name Name of the axis / I/O device
No. Axis number
Device identi- Drive family/firmware ID
fication
Closed loop Position control possible
Ipo drive Interpolation in the control or drive, in this case in the control

Fig. 3-10: List of sercos drives, drive name changed to "RA1"

Click on **Add devices**.

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

The drive is mirrored as a "Real Axis" in the Motion node.

The axis number "1" and the axis names "RA1" have to be used depending on the addressing type.

The project contains the "Mlc1" control, e.g. of type "IndraMotion MLC L65" and the real axis "RA1" (folder **Motion** ► **Real Axes**) or the drive "RA1" (folder **sercos**):

First Steps - Motion

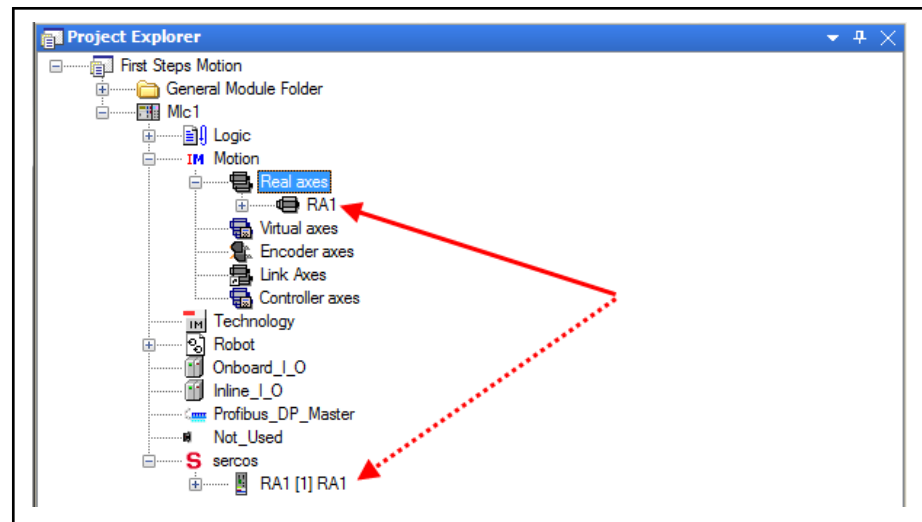


Fig.3-11: Drive/real axis "RA1" is added to the project

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 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**.

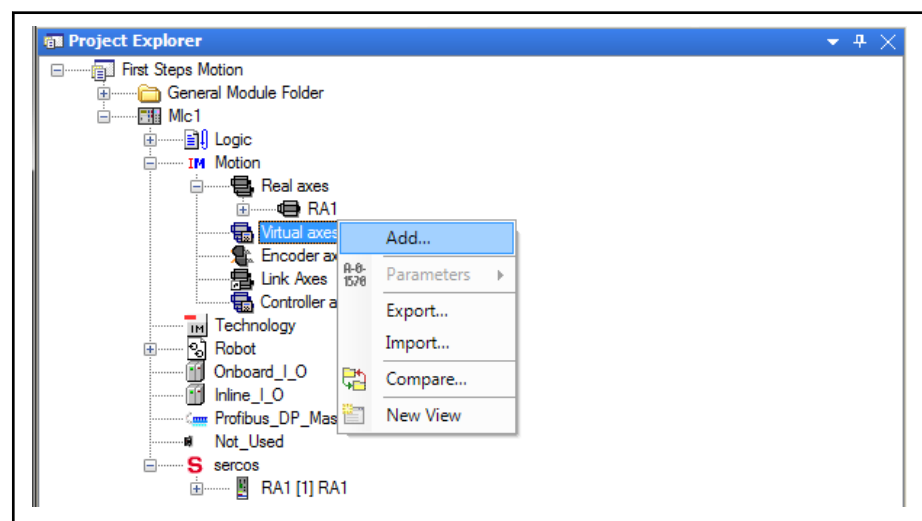


Fig.3-12: Declaring a virtual axis

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

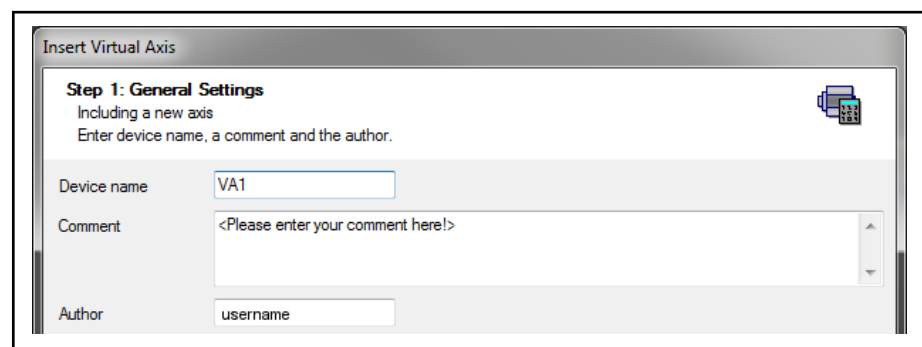


Fig.3-13: "General Settings" dialog for a virtual axis

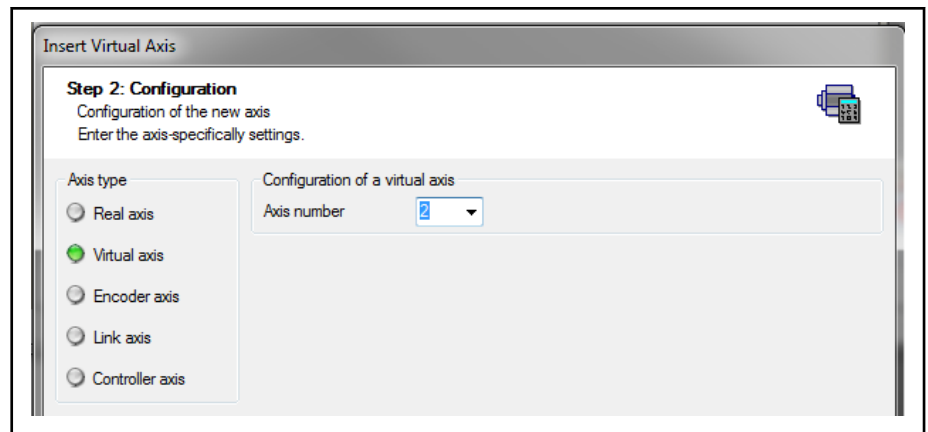


Fig.3-14: "Configuration" dialog for a virtual axis

The axis number "2" and the axis name "VA1" have to be used depending on the addressing type.

Click on **Finish**.

3.3.8 Downloading the Motion Subproject

The first editing part is completed by creating the control and all of the axes in the "First Steps Motion" project and parameterizing control and axis components.

The information collected up to this point must now be loaded into the control. Therefore, highlight the desired device (Mlc1) and click on the icon of the symbol toolbar "Project" or on the menu item **Project ► Switch device online**:

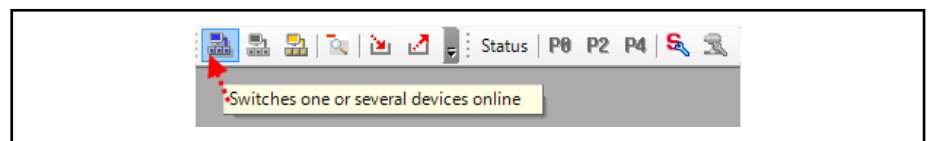


Fig.3-15: Switch device online

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

(1): IndraMotion MLC is now **partially synchronized** and the Motion sub-project download is expected.

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

First Steps - Motion

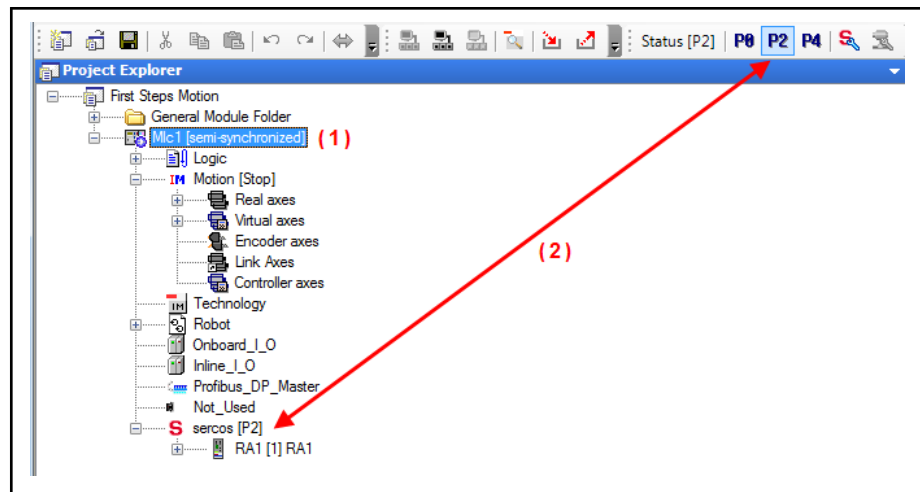


Fig.3-16: Partially synchronized IndraMotion MLC with sercos III field bus in phase P2

To synchronize the control with the project, adjust the axis configuration data (symbol bar "").

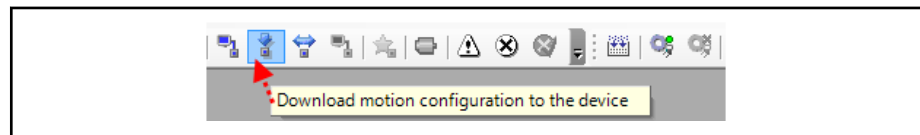


Fig.3-17: Downloading the Motion configuration

In the next dialog window, project data can be transmitted to the control ("Use IndraWorks axis configuration") or the control data can be applied to the project ("Use axis configuration control").

The selected axes are transmitted by confirming with <OK>.

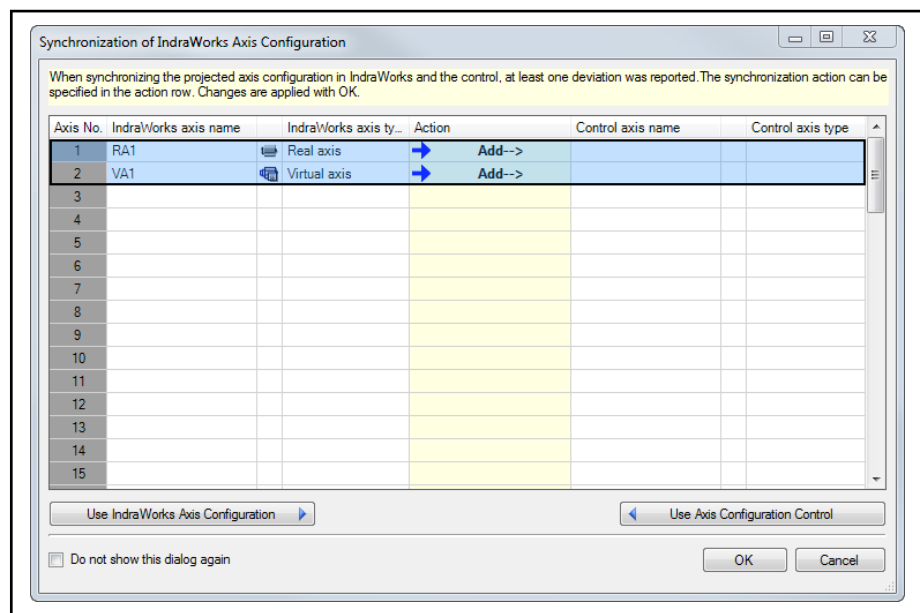


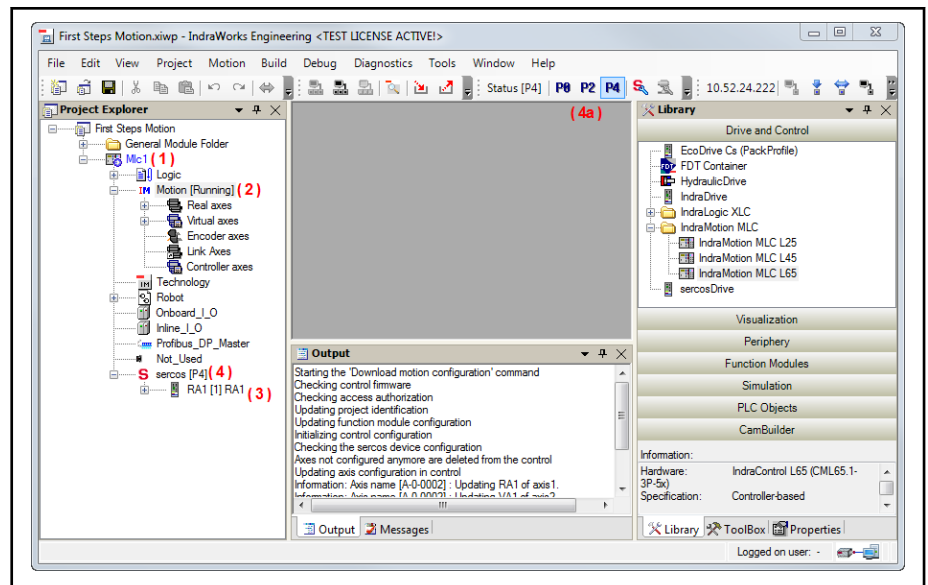
Fig.3-18: Dialog adjusting the IndraWorks axis configuration

The control is now synchronized.



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

First Steps - Motion



- (1) Active control (online)
 - (2) Motion component is now running smoothly
 - (3) Active drive (online)
 - (4, 4a) Communication in mode "P4", operation mode
- Fig.3-19: Download completed successfully (message 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 occur that do not allow phase BB to be reached and making another phase switch to P2 necessary (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 has been switched from IndraDrive (calculation in the drive) to IndraDrive (calculation in the control) or vice versa, an error message can be emitted when changing from P2 to BB. In this case, the basic parameters of the drive must be reloaded. The basic parameters can be loaded via the context menu **Parameter** ▶ **Load basic parameters...** of the axis (RA1).

If the control is in "Parameterization mode (P2)", it has to be switched to "Operating mode (BB)" (context menu of sercos node **sercos state** ▶ **Phase P4** or symbol bar sercos, icon **P4**).

First Steps - Motion



Depending on the type of encoder, an error "F2174, Reference loss by encoder" can occur.

The error can either be acknowledged via <ESC> at the displays of the drive and the control or even better, it can be removed via **Mlc1 ► Motion ► Real Axis ► RA1 ► Establish Position Data Reference ► Position data reference - motor encoder "Set absolute position"**.

Subsequently, "Clear error"

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

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

3.4 Programming a Motion Sequence in 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).

During a project creation with the control Mlc1, the real axis RA1 and the virtual axis VA1, first programming activities of the motion sequence have been started below the "Logic" folder:

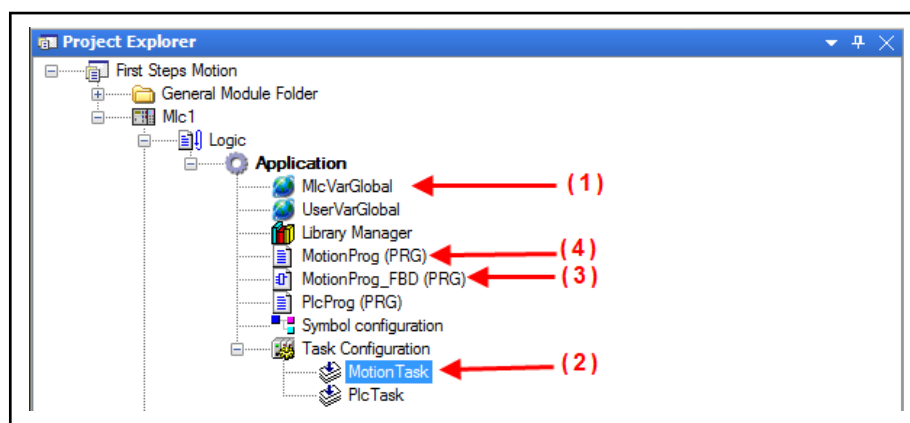


Fig.3-20: Project Explorer after expanding the "Logic" node

The following programs were created:

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

The third program **MotionProg_FBD (3)**, was manually supplemented: **Application**, right mouse button, **Add ► POU**, Entering the name and selecting FBD as programming language.

The programs are currently still empty.

Under "Task Configuration", the following tasks were created:

- **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 part of the sercos cycle. If the tasks are not processed within the specified part of the sercos cycle, the control is stopped.

First Steps - Motion

- **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 for:

- **UserVarGlobal**, reserved for the user, not used in "First Steps - Motion"
- **MlcVarGlobal (1)**, contains the address constants for the axes RA1 and VA1.

Program:

```
(* Automated generated code by MLC. *)
(* Please don't edit, the code will be overwritten...*)
VAR_GLOBAL CONSTANT
(*virtual axis "VA1" with axis number 2*)
VA1: AXIS_REF :=(CntlNo:=LOCAL_CNTRL,AxisNo:=AXIS_2);
(*real axis "RA1" with axis number 1*)
RA1: AXIS_REF :=(CntlNo:=LOCAL_CNTRL,AxisNo:=AXIS_1);
(*...*)
END_VAR
```



Only the Motion task is required for the "First Steps Motion" project.

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

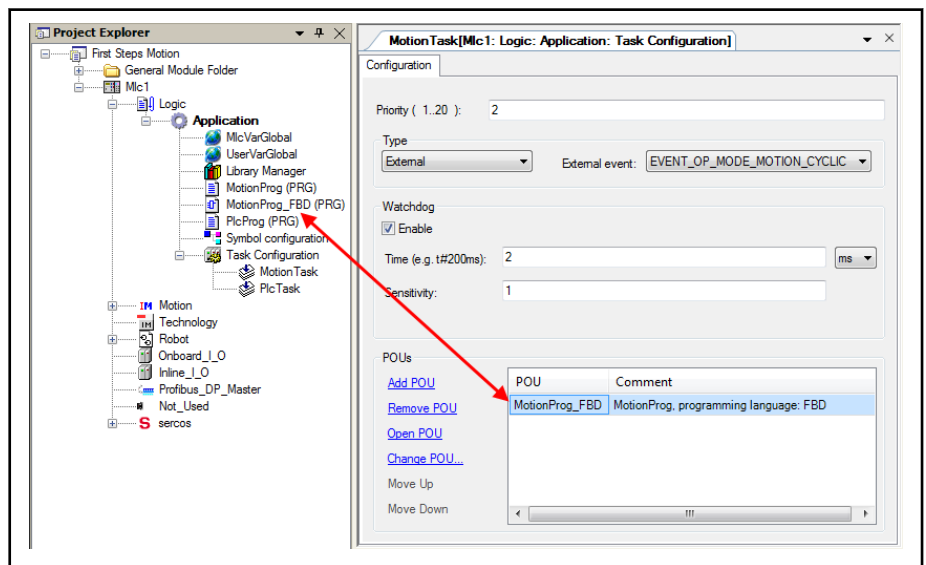


Fig.3-21: Delete call: MotionProg. Add call: MotionProg_FBD

3.4.2 Programming the Virtual Axis VA1, 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 [Programming the Virtual Axis VA1, Programming Language ST](#), page 101.

Similarly to [Project Goal First Steps - Motion](#), page 67, instances of the PLCopen blocks MC_MoveVelocity and MC_Stop are called using "VA1".



The MC_Power instance is not required for virtual axes.

First Steps - Motion

Proceed as follows:

1. Assign the "MotionTask" to the "MotionProg_FBD" program.
2. Open the "MotionProg_FBD" program (by double-clicking on it).

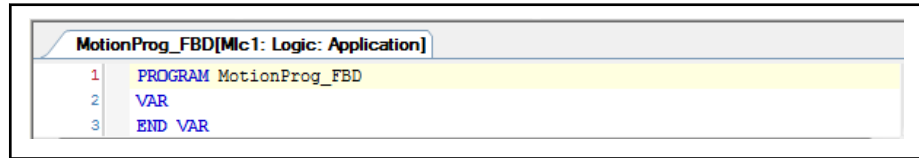


Fig.3-22: New program in FBD (Function Block Diagram)

Insert the first function block into the implementation, network 1, by right-clicking on it.

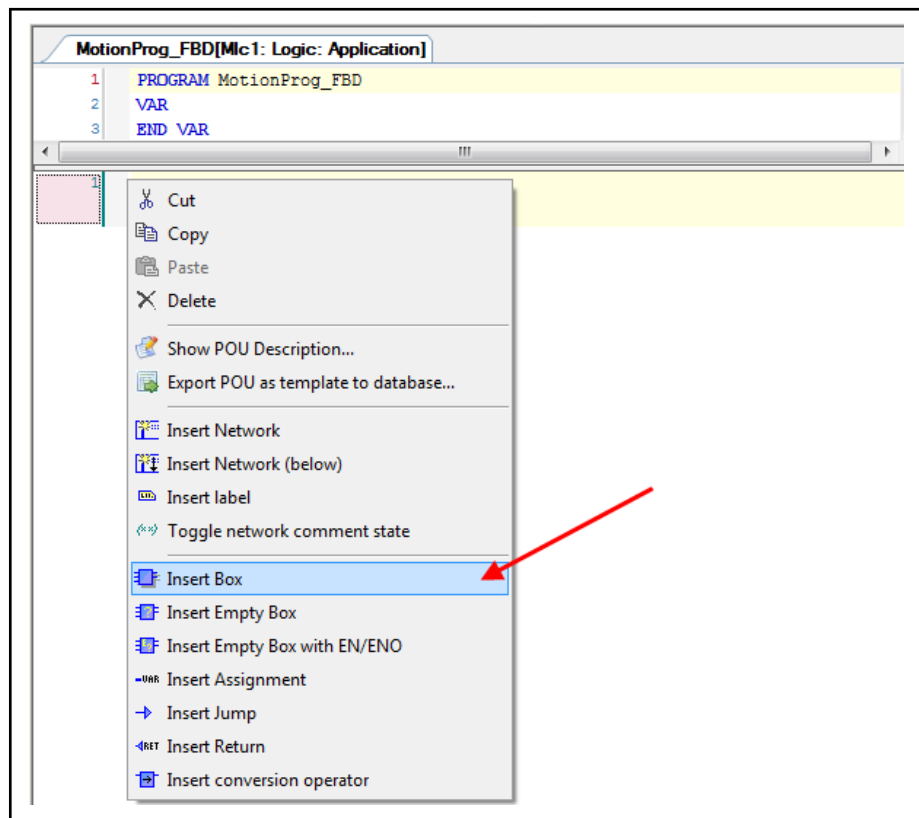


Fig.3-23: Generating a new function block

To do so, select the following in the open selection window:

Function Blocks ► ML_PLCOPE ► POU ► Motion ► MC_MoveVelocity.

First Steps - Motion

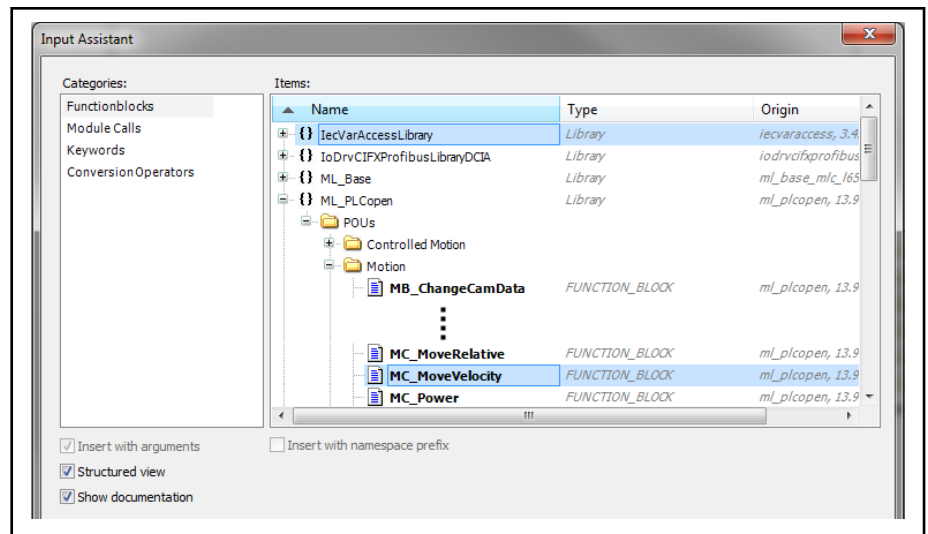


Fig.3-24: Selection window for function block types

Confirm the insertion of the "MC_MoveVelocity" function block with **OK** and highlight the three question marks **???** in the "MotionProg_FUP" program by right-clicking on them. Enter the instance name for this "MoveVelVA1" function block and complete the entry with <Enter>. The "Variable Declaration" window opens which can be acknowledged immediately with **OK**:

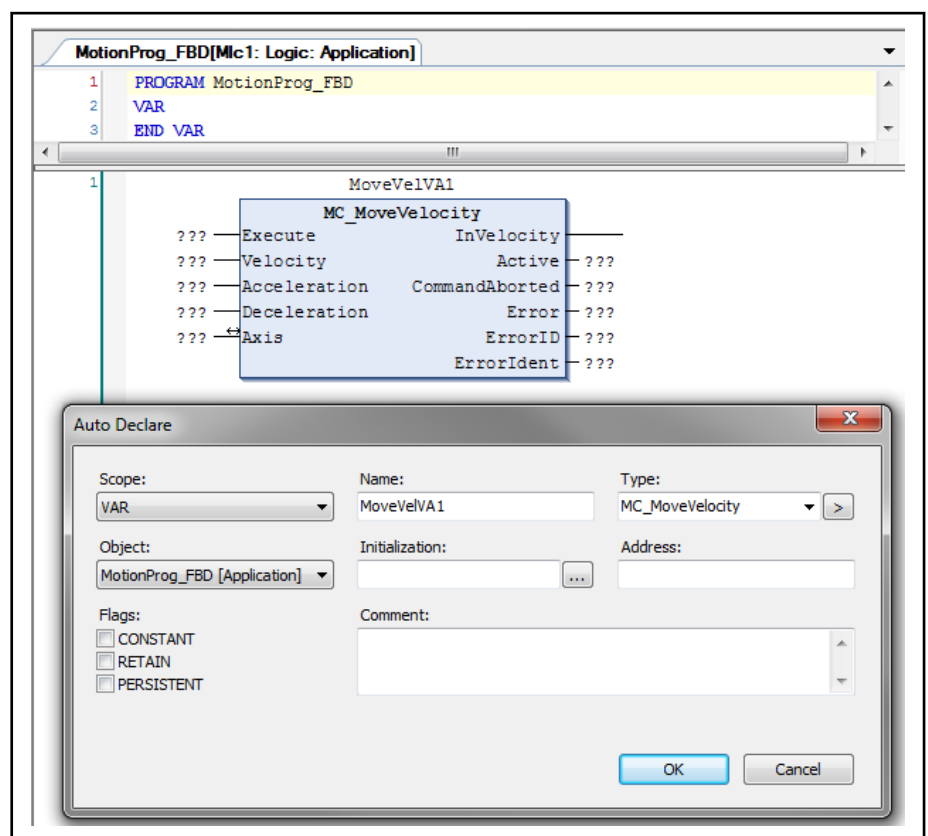


Fig.3-25: Declaration of the instance name of an FB, type "MC_MoveVelocity" using the "Variable Declaration" window

Now, the "MoveVelVA1" instance of the PLCopen function block MC_MoveVelocity is declared. Assign the FB inputs.

First Steps - Motion

Simultaneously to the instance name of the FB, assign the "Execute" input with a variable called "ex_MoveVelVA1", type BOOL:

- Highlight the 3 question marks in front of the "Execute" input
- Enter the name "ex_MoveVelVA1" and complete with <Enter>
- The open window "Variable Declaration" can be confirmed immediately with <Enter>

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

- Highlight the 3 question marks in front of the "Velocity" input
- Enter the name "VcomVA1" and confirm using <Enter>
- Enter REAL as type and 200 (rpm) as initial value. Press <Enter> to confirm

For the "Acceleration" inputs, please enter 20 (rad/s²) and for "Deceleration" 20 (rad/s²).



Enter the values directly into the FB inputs.

Alternatively a variable with the respective value can be used.

To assign the "Axis" input, execute the following:

- Click on ??? and select the adjacent Input Assistant :
Select **Variables** ► **Application** ► **MlcVarGlobal** ► **VA1**, insert with namespace prefix.

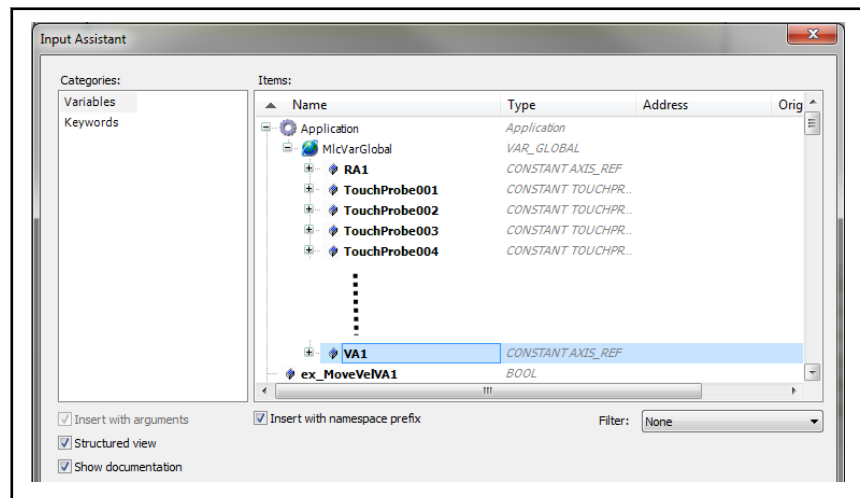


Fig.3-26: Selecting the axis instance when assigning a function block of the type "MC_MoveVelocity"

- Confirm using **OK**.
- Complete the entry of the name with <Enter>.



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

MlcVarGlobal.VA1

To obtain feedback about whether the function block has worked correctly, its outputs are assigned.

Highlight the area on the right, next to the output "InVelocity", using the mouse and select in the context menu (right mouse button):

Insert Assignment:

First Steps - Motion

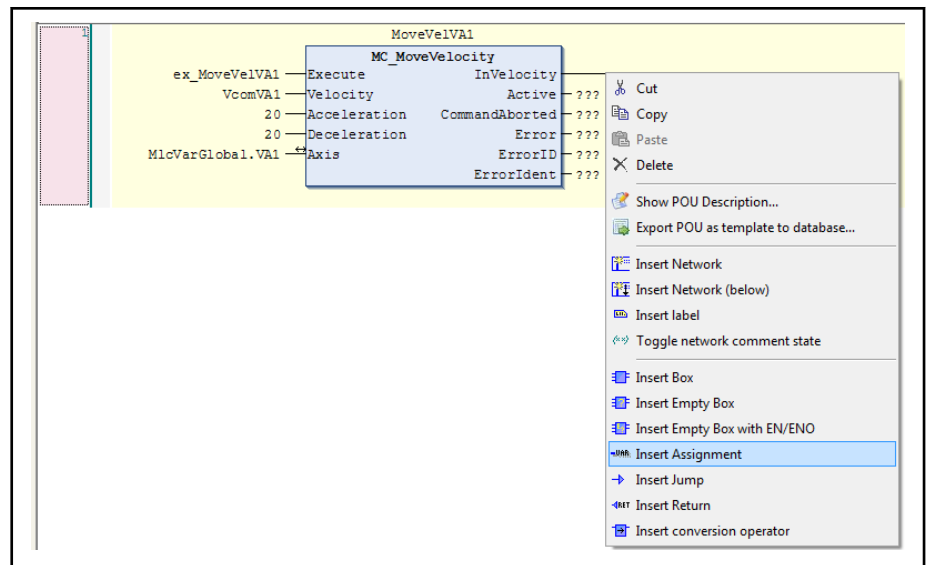


Fig.3-27: Assignment at the function block output

Assign the "InVelocity" output of the block as usual using the "in_MoveVelVA1" variable, type BOOL:

- Select the 3 question marks **???** at the output "InVelocity"
- Enter the name "in_MoveVelVA1" and confirm using <Enter>
- The open window "Variable Declaration" can be confirmed immediately with <Enter>
- Assign "ac_MoveVelVA1" to the Active output.
- Assign "ca_MoveVelVA1" to the CommandAborted output
- Assign "er_MoveVelVA1" to the Error output
- Assign "ID_MoveVelVA1" to the ErrorID output
- Assign "Ident_MoveVelVA1" to the Error output

The PLCopen FB, type MC_MoveVelocity, is thus assigned according to its task. The declaration section was automatically completed.

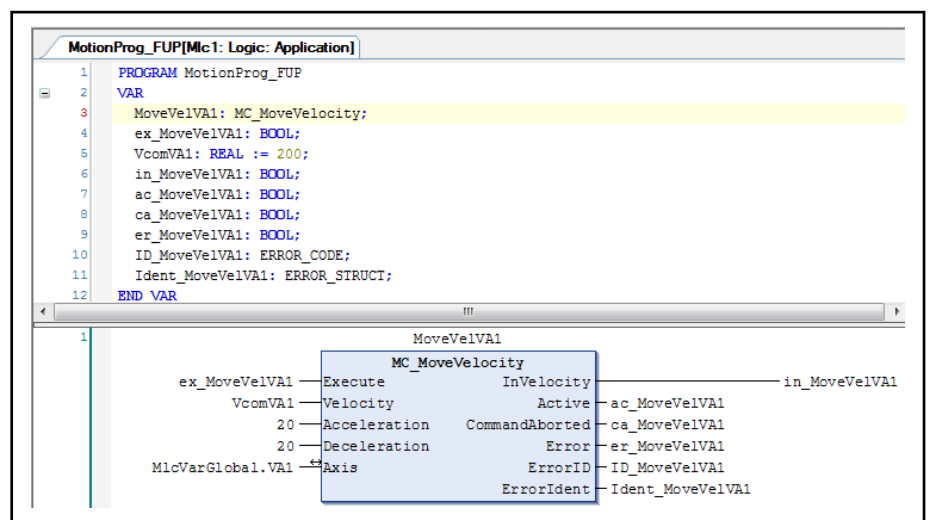


Fig.3-28: PLCopen FB, "MC_MoveVelocity" type is inserted and assigned



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

First Steps - Motion

MC_Stop

Generate an instance of the PLCopen FB MC_Stop.

Insert MC_Stop and declare a "StopVA1" instance

To insert a new network, click on network 1 with the right mouse button,

Insert Network (below)

Insert MC_Stop and declare a "StopVA1" instance

Assign inputs and declare input variables if necessary:

"Execute" → "ex_StopVA1" of type BOOL

"Deceleration" → 50

"Axis" → "MlcVarGlobal.VA1" of type "AXIS_REF")

Outputs:

"Done" → "do_StopVA1" of type BOOL

"Active" → "ac_StopVA1" of type BOOL

"CommandAborted" → "ca_StopVA1" of type BOOL

"Error" → "er_StopVA1" of type BOOL

"ErrorID" → "ID_StopVA1" of type ERROR_CODE

"ErrorIdent" → "Ident_StopVA1" of type ERROR_STRUCT

Actual velocity "VactVA1"

The final step is the implementation of a network in which the current velocity of axis VA1 "VactVA1" (data type REAL) can be read.

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

The data from all axes that can control an MLC is located in the array

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

where the axis number is used as the index.

The variable is provided with the name

GVL_Base.AxisData[2].rActualVelocity_i

for the virtual axis, axis number 2, (axis number 1 has RA1, since scanning RA1 was executed before declaring VA1).

- To insert a new network, click on network 2 with the right mouse button
Insert Network (below)
- **Insert Assignment**
- Insert "VactVA1" on the right **???** and confirm with <Enter>, type "REAL" in the declaration window and confirm with <Enter>.
- Go to the left **???** and click on  using the input assistance to navigate to **Variables ► ML_BASE ► Global Variable List ► GVL_Base ► AxisData:**

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

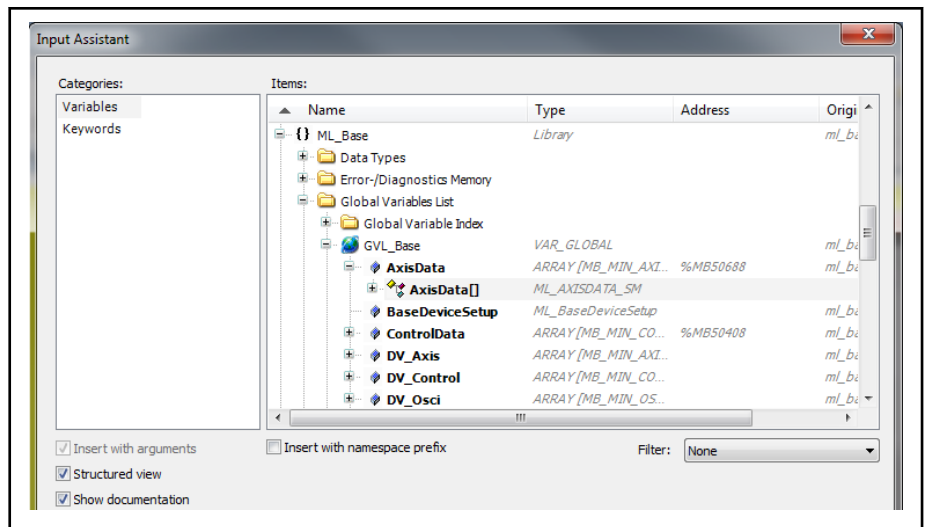


Fig.3-29: Actual velocity value VA1 from the global variable list of the library ML_Base.library

Input is supported by the Intellisense function; i.e. all available elements are opened and can be selected after entering "." for "AxisData[2]."

Confirm "rActualVelocity_i" with <Enter>.

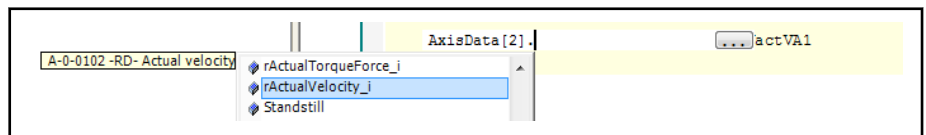


Fig.3-30: Selecting the variables "ActualVelocity"

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

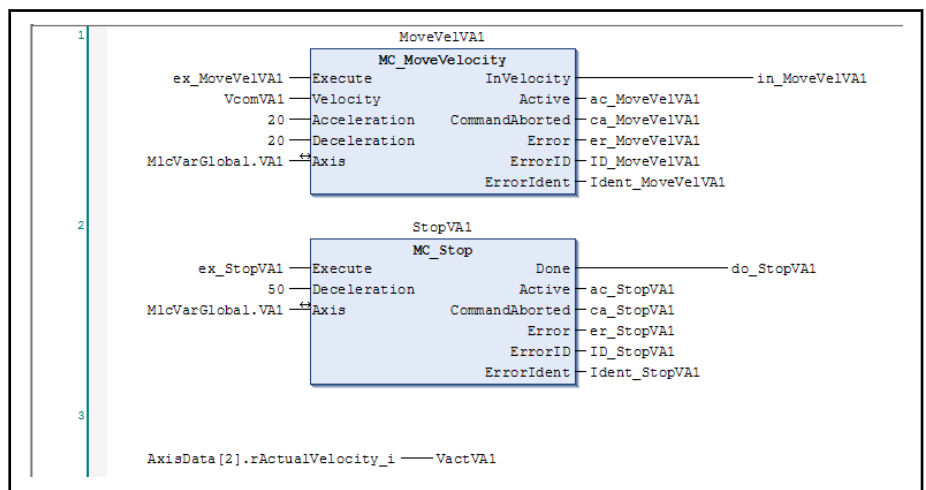


Fig.3-31: Implementing the virtual axis VA1



AxisData[].rActualVelocity_i; and
GVL_Base.AxisData[].rActualVelocity_i;
are equivalently used by the compiler.

3.4.3 Download of the PLC Sub-project Virtual Axis VA1 (FBD)

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

First Steps - Motion

The compilation can be accessed via the IndraWorks menu item

- **Create ► Build**, compilation of the added part of the application
- **Create ► Rebuild**, compilation of the complete application

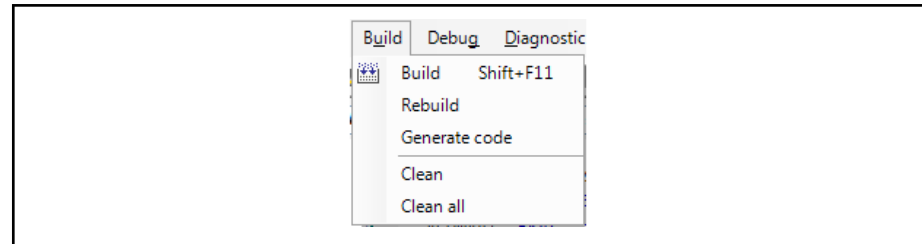


Fig.3-32: Build application

To apply the compiled data to the target control, select the IndraWorks menu item **Debug ► Login** (see the following figure).

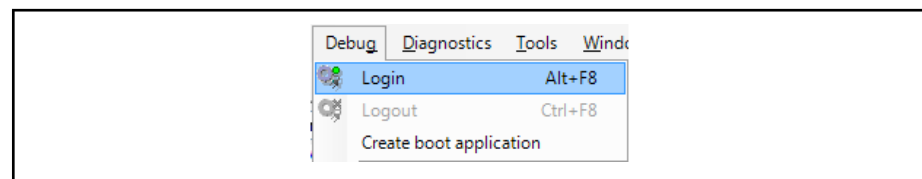


Fig.3-33: Transferring compiled data to the control

A prompt asking for agreement to download is displayed depending on whether an application is already loaded and whether this application is a predecessor of its 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 not valid.

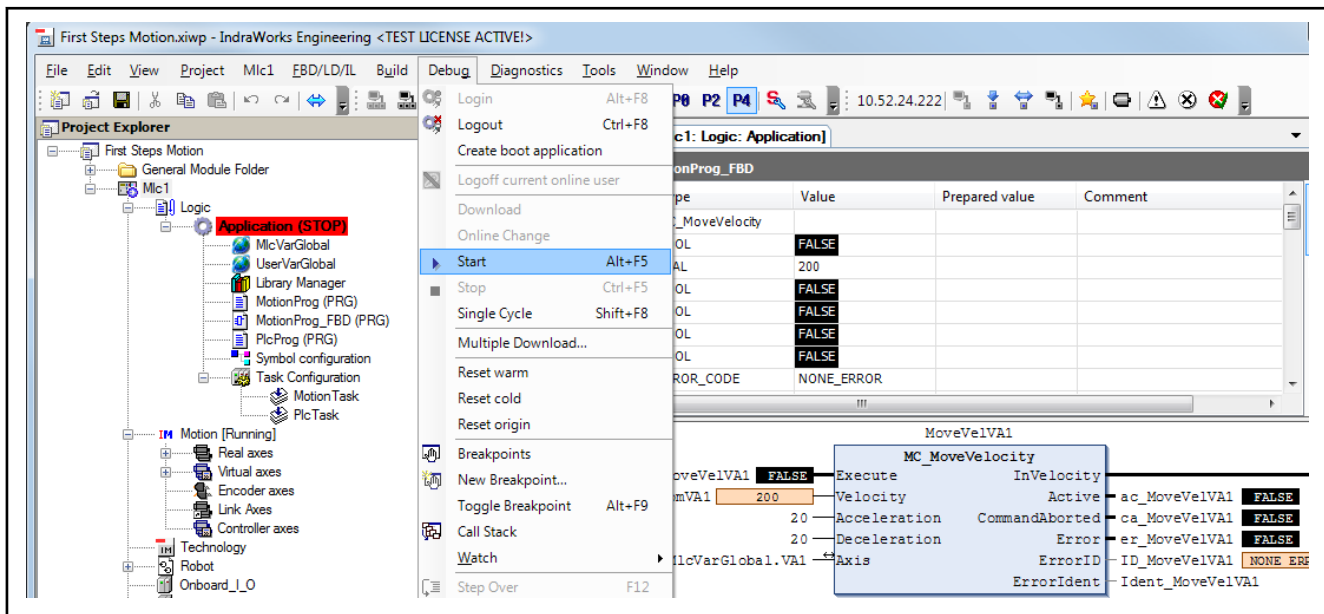


Fig.3-34: Starting the program processing in the control (1)

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

First Steps - Motion

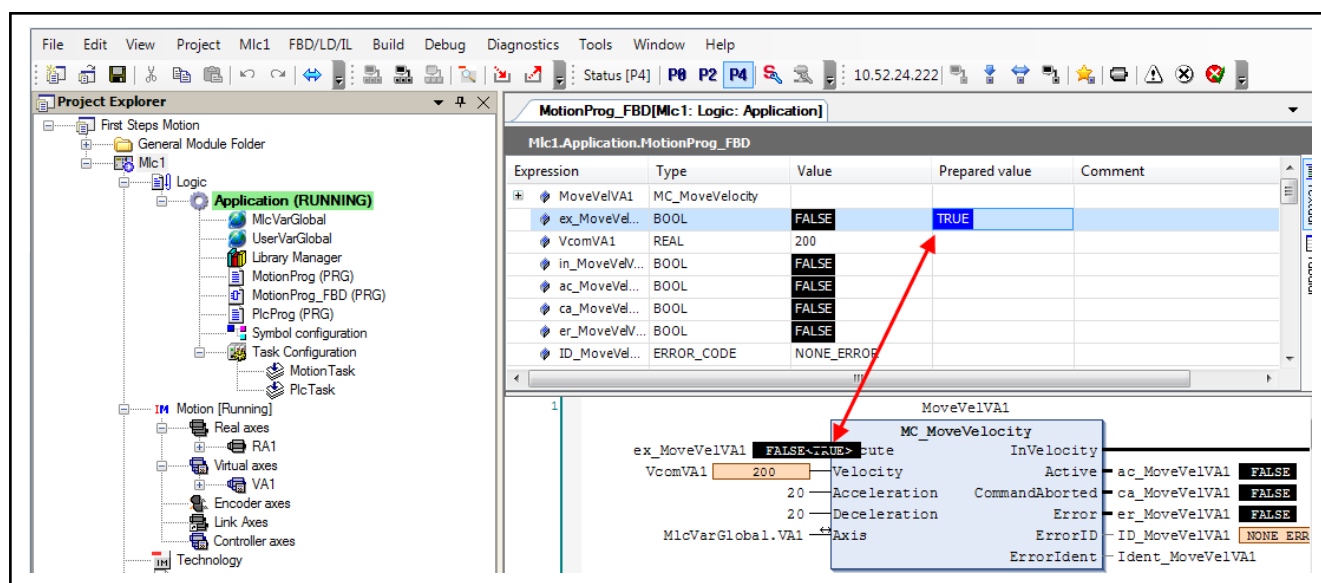


Fig.3-35: Starting the program processing in the control (2)

The motion process starts automatically since the processing of the programmed function blocks is activated one after the other.

"Write values" of input variables

To enable processing of "MoveVelVA1" of type "MC_MoveVelocity", a positive edge must occur at the "Execute" input of the FB.

Double-click on FALSE behind the input variable "ex_MoveVelVA1" using the left mouse button. There is a highlighted "FALSE" behind "ex_MoveVelVA1" and a "TRUE" in the declaration line at first, then a "FALSE" after the next click and then an empty space (deactivated).

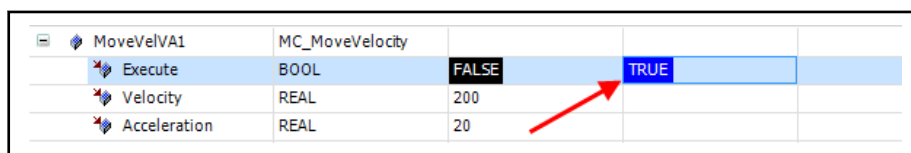


Fig.3-36: "Write values" in the declaration part of the program

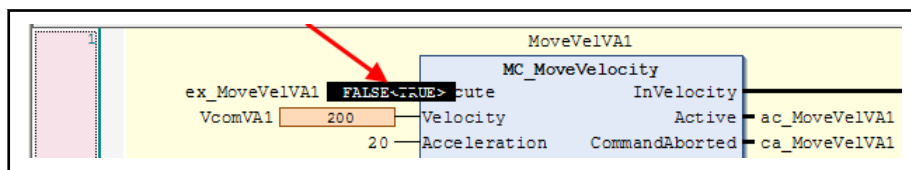


Fig.3-37: "Write values" at the function block interface

Via **Debug** ► **Write Values** or <Ctrl>+<F7>, "ex_MoveVelVA1" in the control is set to "TRUE".



- "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 created sets the outputs "InVelocity" and "Active" of the "MoveVelVA1" function block to "TRUE" (1).

The axis travels at 200 rpm (2).

First Steps - Motion

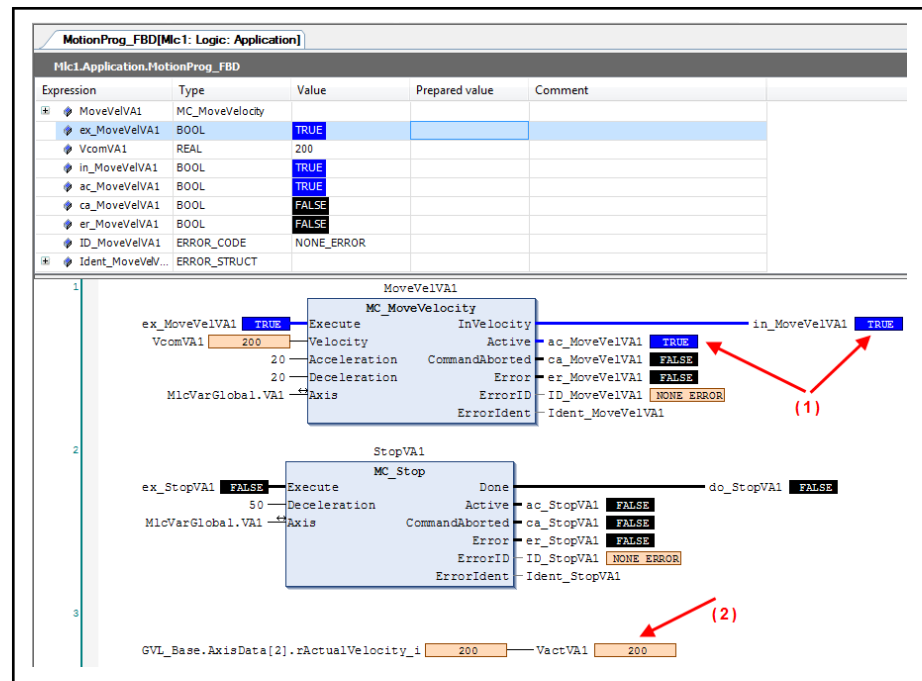


Fig.3-38: After "Write Values" at the "Execute" input

Switch "ex_MoveVelVA1" again to "FALSE". The axis continues traveling at 200 rpm ②.

By setting the variable "x_StopVA1" at the "Execute" input of the "StopVA1" function block of type MC_Stop, the axis motion can be stopped.

Stop the application and log out.

3.4.4 Recording the Motion of Axis VA1 (Trace), FBD

IndraLogic provides a Trace tool to trace motion processes. The following has to be recorded using this tool.

- the axis velocity "VactVA1"
- for "MoveVelVA1" (MC_MoveVelocity)
 - "ex_MoveVelVA1" (Execute)
 - "in_MoveVelVA1" (InVelocity)
 - "ca_MoveVelVA1" (CommandAborted)

and

- for "StopVA1" (MC_Stop)
 - "ex_StopVA1" (Execute)

To do so, the trace recording must first be configured.

Insert the object "Trace_VA1" via the context menu of "Application" **Add ► Trace....** Open the object by double clicking on it.

First Steps - Motion

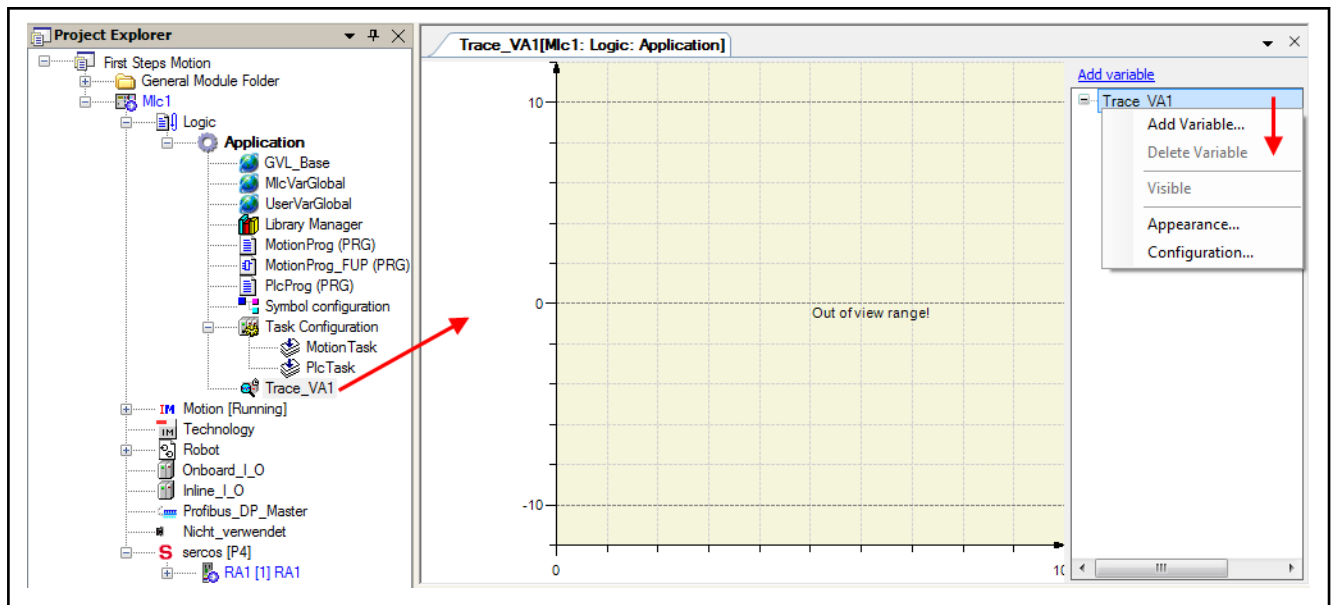


Fig.3-39: Dialog: New Trace object "Trace_VA1" with context menu

The following settings can be selected for the Trace object via the context menu on the right side of the page:

- Configuration
- Display settings (appearance)
- Add new variable

Configuration...

The new Trace object must firstly be configured:

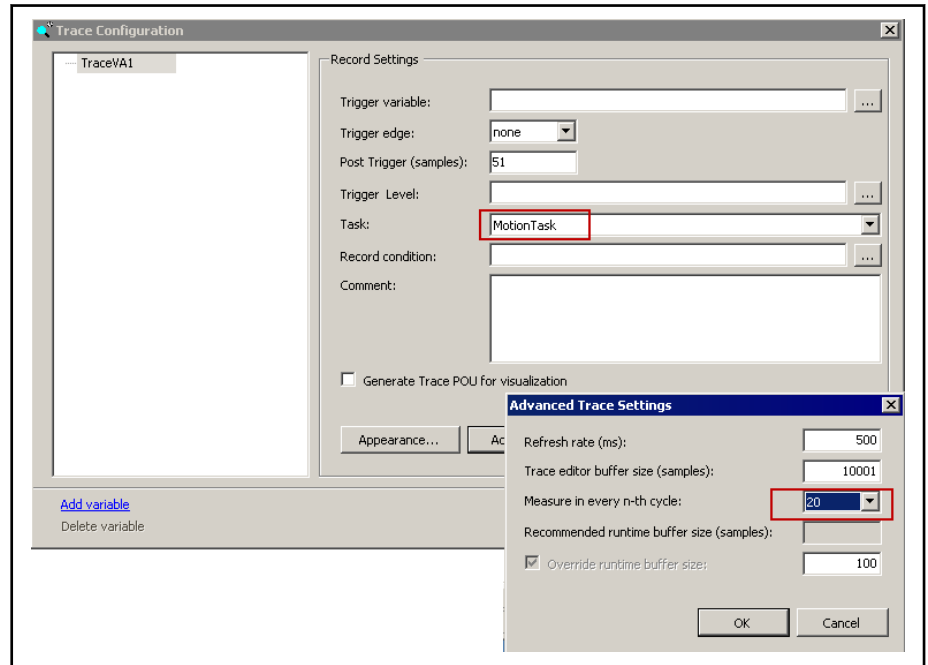


Fig.3-40: Dialog: Trace configuration with extended Trace setting

The required settings are

- Task: **MotionTask**
- Measurement in each **20** cycle, $20 \times 2 \text{ ms} = 40 \text{ ms}$

First Steps - Motion

Task: From the list of available tasks, select the the task for which the values of trace variables are to be read after they are processed. Default: First task in the task configuration tree.

Record condition: A Boolean variable, a value or a Boolean expression can be entered here. As soon as this condition is met, or receives the value TRUE, value recording begins. If nothing is entered here, value recording starts as soon as the trace configuration is loaded into the control and the application is running.

Measurement: Enter the number of cycles that have to elapse before the value of the trace variables is to be read again. Using the formula (Buffer size * Record in every x-th cycle * Task interval), the minimum recording time frame can be estimated.

Recommended runtime buffer size: Size of the ring buffer in bytes, which is currently provided in the runtime system for collecting trace variable values in the currently configured record. The buffer content is then displayed as a value curve or curves in the trace window. Default: 100.



If the configuration changes, the Trace object must be logged into again.

Display settings (appearance)

Settings for the display can now be entered. These settings are also effective even without logging in again.

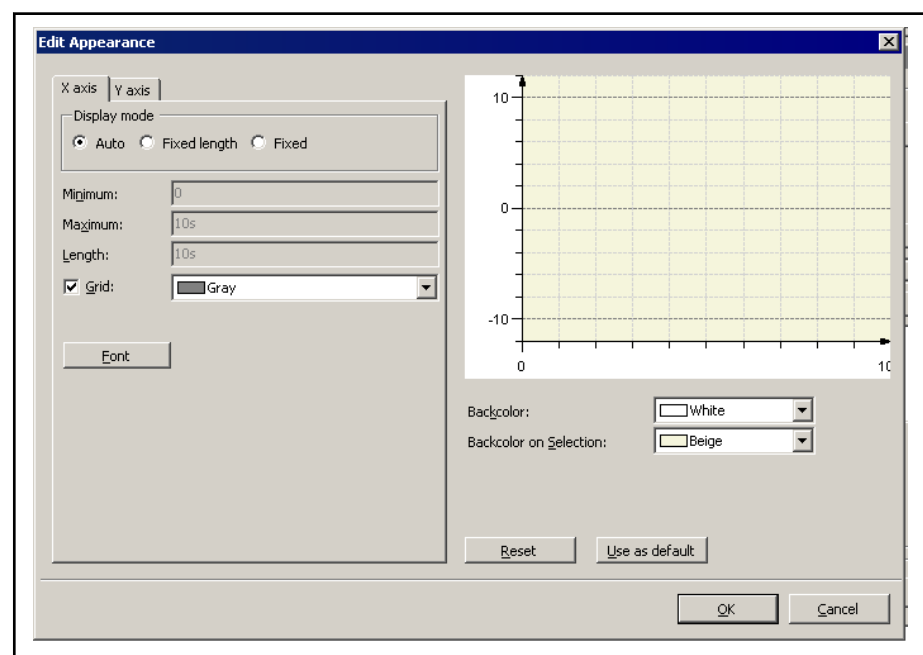


Fig. 3-41: Dialog: Display settings (appearance)

Select **Auto**, positioning a **Length 10s** window over the curve. The **Scale division** is set out in a 1s grid with a **scale division** of 1s.

Selecting the new variables

Now the variables have to be named and their appearance has to be selected.

First Steps - Motion

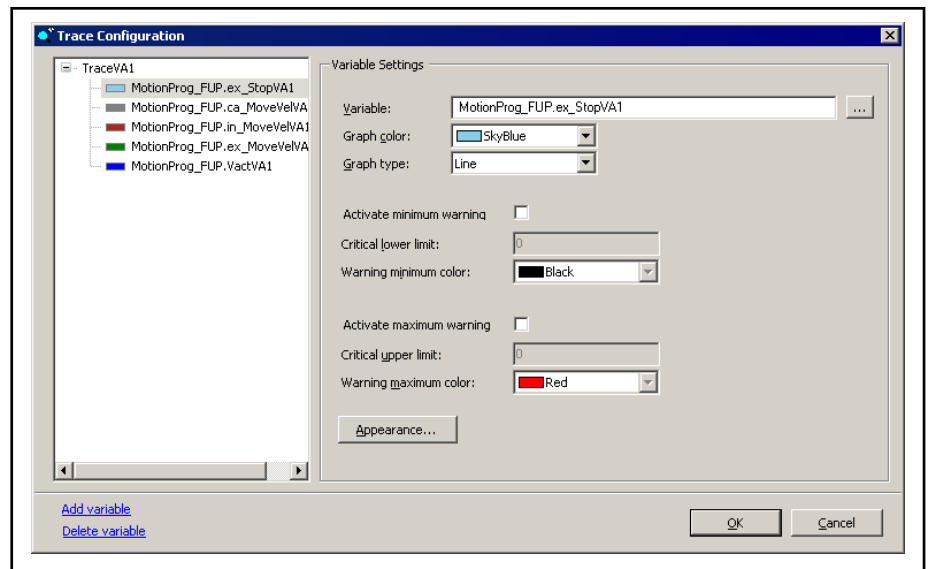


Fig.3-42: Dialog: Creating new variables and adapting appearance

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:

The context menu is displayed by right-clicking into the left section of the Trace window.

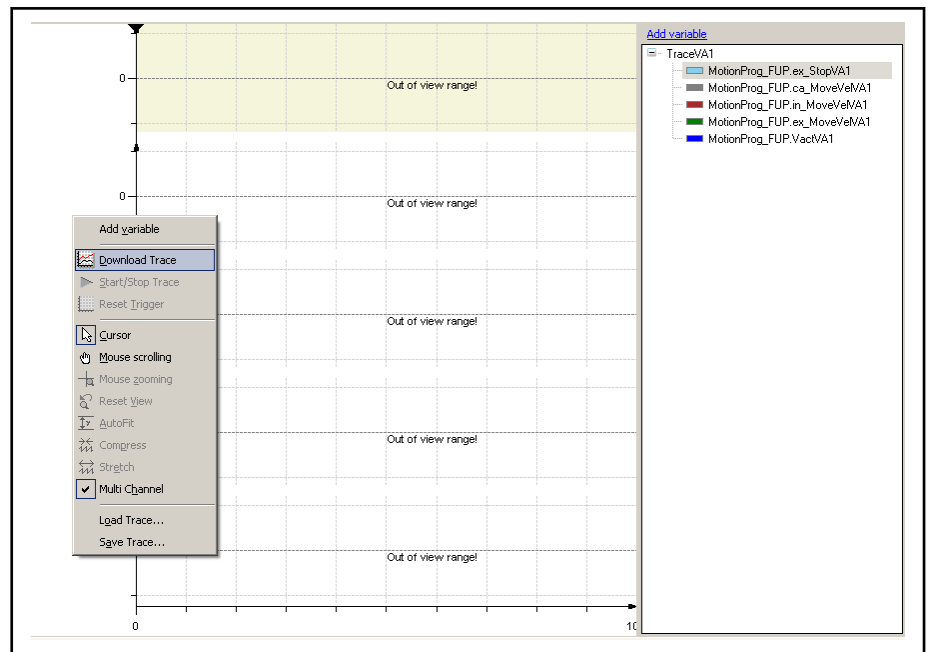


Fig.3-43: Menu to control the Trace recording

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

Start/Stop Trace: This command starts or stops the display of the Trace recording.

First Steps - Motion

If the recording is currently stopped, the  icon is positioned before the command.

If the recording is currently running, the  icon is positioned before 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 enables the view of the Trace curve to be changed. 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 with 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.

Mouse scrolling: In this mode, the 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.

Default appearance: This command can be used to reset the display of the recording(s) to the default values after they have been changed by zooming.

Performing the Trace:

1. Logging 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" trace context menu.
The Trace recording then starts up automatically.
3. Write values.
The following values are set identically to ["Write Values" of input variables](#), [page 87](#),.
 - ex_MoveVelVA1 to TRUE
 - ex_StopVA1 to TRUE
4. End the recording.
Stop the Trace via the Trace context menu item "Start/Stop Trace".
5. Evaluate the Trace recording.
Set the window data (refer to the screen below).

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

First Steps - Motion

Shortcuts	Action in the trace diagram	Mouse control
<↑/↓> 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

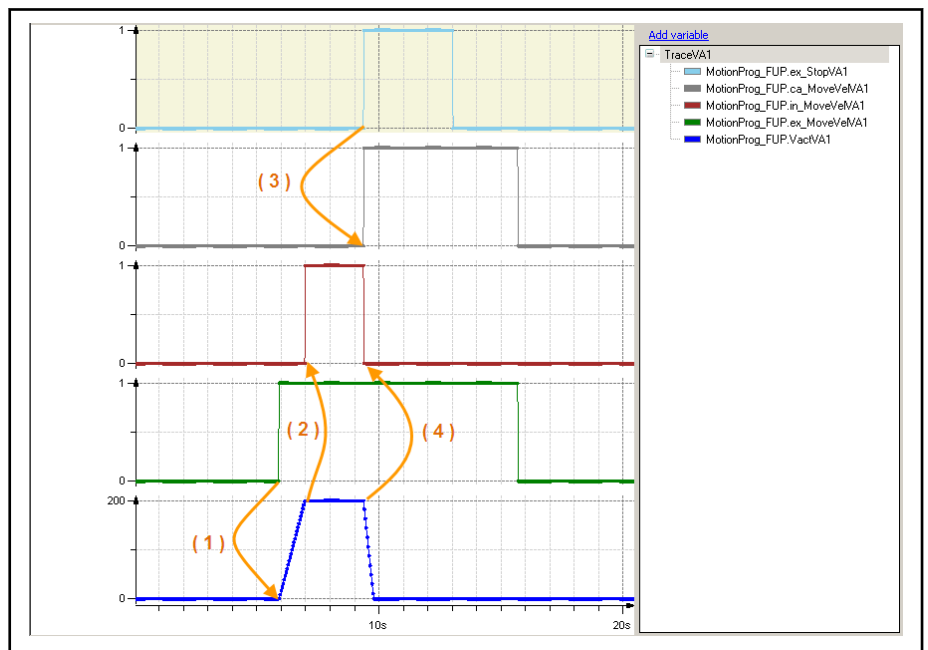


Fig.3-44: Trace recording for the virtual axis VA1

Evaluate the Trace recording

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

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

First Steps - Motion

3.4.5 Programming the Real Axis RA1 (FBD)



From here onwards, the "real" drive technology is required
or

Select the second virtual axis **without** MC_Power as an alternative!

The use of a **real** axis firstly requires an instance of the PLCopen function block MC_Power.

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

The real axis is stopped with an instance of MC_Stop.

The programming is executed similarly to section [Programming the Virtual Axis VA1, Programming Language ST, page101](#).

MC_Power

To insert a new network, click on network 3 with the right mouse button

Insert Network (below)

Insert MC_Power and declare a "PowerRA1" instance

Assign inputs and declare input variables if necessary:

"Enable" → "en_PowerRA1" of type BOOL

"Axis" → "MlcVarGlobal.RA1" of type "AXIS_REF"

Outputs:

"Status" → "st_PowerRA1" of type BOOL

"Error" → "er_PowerRA1" of type BOOL

"ErrorID" → "ID_PowerRA1" of type ERROR_CODE

"ErrorIdent" → "Ident_PowerRA1" of type ERROR_STRUCT

MC_GearIn

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

Insert Network (below)

Insert MC_GearIn and declare the "GearInRA1" instance

Assign inputs and declare input variables if necessary:

"Execute" → "ex_GearInRA1" of type BOOL

"RatioNumerator" → 5

"RatioDenominator" → 10

"MasterFineadjust" → 20 (%)

"Master" → "MlcVarGlobal.VA1" of type "AXIS_REF"

"Slave" → "MlcVarGlobal.RA1" of type "AXIS_REF"

Outputs:

"InGear" → "ig_GearInRA1" of type BOOL

"Active" → "ac_GearInRA1" of type BOOL

"CommandAborted" → "ca_GearInRA1" of type BOOL

"Error" → "er_GearInRA1" of type BOOL

"ErrorID" → "ID_GearInRA1" of type ERROR_CODE

"ErrorIdent" → "Ident_GearInRA1" of type ERROR_STRUCT

MC_GearOut

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

Insert Network (below)

Insert MC_GearOut and declare a "GearOutRA1" instance

Assign inputs and declare input variables if necessary:

"Execute" → "ex_GearOutRA1" of type BOOL

"Slave" → "MlcVarGlobal.RA1" of type "AXIS_REF"

Outputs:

"Done" → "do_GearOutRA1" of type BOOL

"Active" → "ac_GearOutRA1" of type BOOL

"Error" → "er_GearOutRA1" of type BOOL

"ErrorID" → "ID_GearOutRA1" of type ERROR_CODE

"ErrorIdent" → "Ident_GearOutRA1" of type ERROR_STRUCT

MC_Stop

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

Insert Network (below)

Insert MC_Stop and declare a "StopRA1" instance

Assign inputs and declare input variables if necessary:

"Execute" → "ex_StopRA1" of type BOOL

"Deceleration" → 50 (rad/s²)

"Axis" → "MlcVarGlobal.RA1" of type "AXIS_REF"

Outputs:

"Done" → "do_StopRA1" of type BOOL

"Active" → "ac_StopRA1" of type BOOL

"CommandAborted" → "ca_StopVA1" of type BOOL

"Error" → "er_StopRA1" of type BOOL

"ErrorID" → "ID_StopRA1" of type ERROR_CODE

"ErrorIdent" → "Ident_StopRA1" of type ERROR_STRUCT

Actual velocity "VactRA1"

The final step is the implementation of a network in which the current velocity of axis RA1 "VactRA1" (data type REAL) can be read.

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

The data from all axes that can control an MLC is located in the array

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

where the axis number is used as the index.

2) 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[1].rActualVelocity_i

for the real axis, axis number 1, (RA1 has axis number 1, since scanning RA1 was executed before declaring VA1).

- To insert a new network, right-click on network 7
Insert Network (below)
- **Insert Assignment**
- Insert "VactRA1" on the right ??? and confirm with <Enter>, type "REAL" in the declaration window and confirm with <Enter>.
- Go to the left ??? and click on [] using the input assistance to navigate to **Variables ► ML_Base ► Global Variable List ► GVL_Base ► AxisData**:

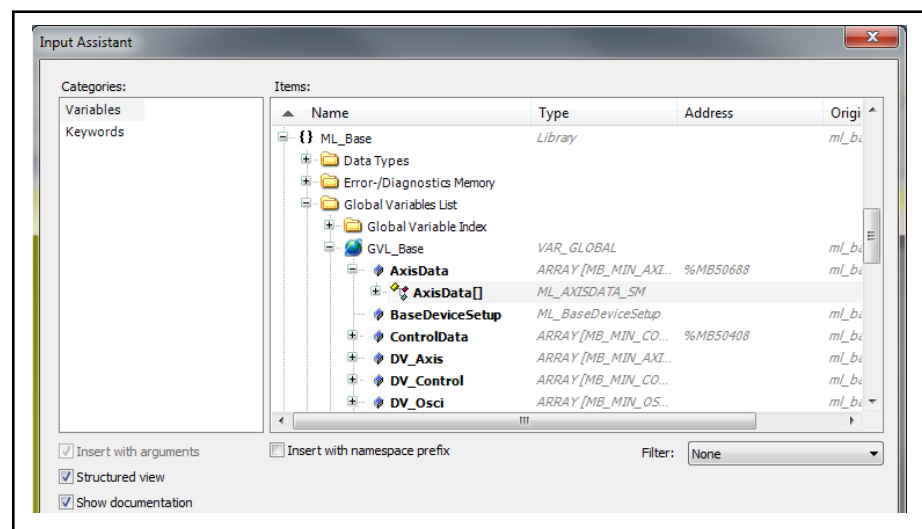


Fig. 3-45: Actual velocity value VA1 from the global variable list of the library ML_Base.library

Input is supported by the Intellisense function; i.e. all available elements are opened and can be selected after entering "." for "AxisData[1]."

Confirm "rActualVelocity_i" with <Enter>.

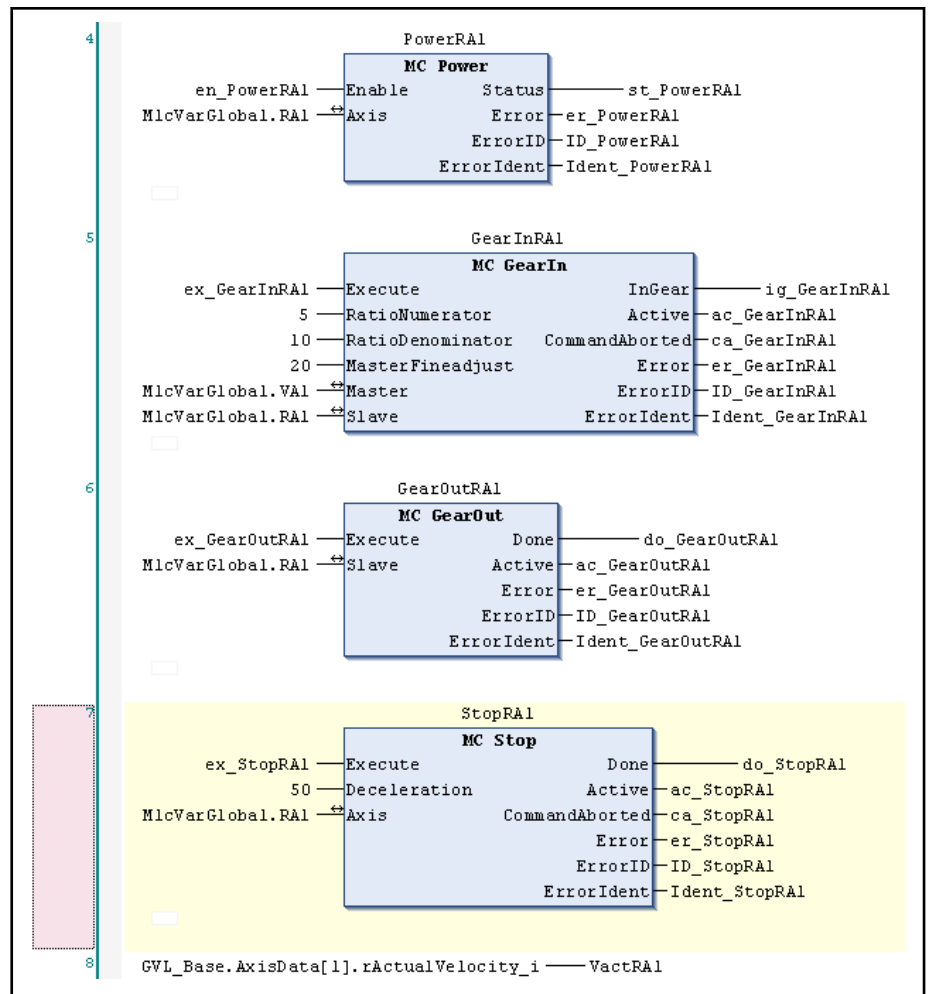


Fig.3-46: Implementing the real axis



AxisData[].rActualVelocity_i; and
GVL_Base.AxisData[].rActualVelocity_i;
are equivalently used by the compiler.

3.4.6 Downloading the Complete PLC Project (FBD)

The project is downloaded identically to the description in section [Download of the PLC Sub-project Virtual Axis VA1 \(FBD\)](#), page 85.

The new project overwrites the old 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_PowerRA1" (MC_Power) by "Write Values".
st_PowerRA1= TRUE (MC_Power).

2. Start the virtual axis.

Enable "ex_MoveVelVA1" (MC_MoveVelocity) by "Write Values".
Start up ends at VactVA1 = 200 rpm.

First Steps - Motion

3. Trace the real axis.

Enable the "ex_GearInRA1" (MC_GearIn) by "Write Values".

The expected gear transmission ratio is:

$$Tr = (\text{RatioNumerator} / \text{RatioDenominator}) * (1 + \text{MasterFineadjust})$$

for VactVA1 = 200 rpm results VactRA1 = 120 rpm.



The MasterFineadjust value is given in "%".

4. If MC_GearIn is "InGear", the velocity "VcomVA1" 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 the "ex_GearInRA1" (MC_GearIn) by "Write Values".
in_GearInRA1= FALSE, ac_GearInRA1= TRUE.
The real axis continues to follow the virtual axis.
6. Terminate synchronicity.
Enable the "ex_GearOutRA1" (MC_GearOut) by "Write Values".
The dependency of the real axis on the virtual axis is suspended. However, the velocity at which both axes continue to rotate remains unchanged.
7. Change the command velocity "VcomVA1".
The changed command velocity only affects the virtual axis. "VactRA1" remains unchanged.
8. Disable "ex_GearOutRA1" by "Write Values".
do_GearOutRA1= FALSE.
9. Stop the virtual axis.
Enable "ex_StopVA1" (MC_Stop).
The virtual axis is decelerated to VactVA1 = 0 rpm.
10. Disable "ex_StopVA1" (MC_Stop).
11. Stop the real axis.
Enable "ex_StopRA1" (MC_Stop).
The real axis is decelerated to VactRA1= 0 rpm. After the axis stops, it is actively held in this position.
12. Disable "ex_StopRA1" (MC_Stop).
The active holding of the real axis ends.
13. Power off the real axis.
Disable "en_PowerRA1" (MC_Power) by "Write Values".



Only possible in the "standstill" mode after executing the stop at RA1.

3.4.8 Recording the Motion of Both Axes (Trace), FBD

The Trace tool was used, in the same way as in section [Recording the Motion of Axis VA1 \(Trace\), FBD, page 88](#), to visualize the startup of the virtual axis and its synchronization.

First Steps - Motion

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_PowerRA1" (MC_Power).
2. Start the virtual axis.
Enable "ex_MoveVelVA1" (MC_MoveVelocity).
3. Follow the real axis.
Enable "ex_GearInRA1" (MC_GearIn).
4. Speed command value jump of the virtual axis.
Increase the speed command value.
Switch on by "ex_MoveVelVA1" TRUE $\Rightarrow$ FALSE $\Rightarrow$ TRUE.

The following figure shows the trace result:

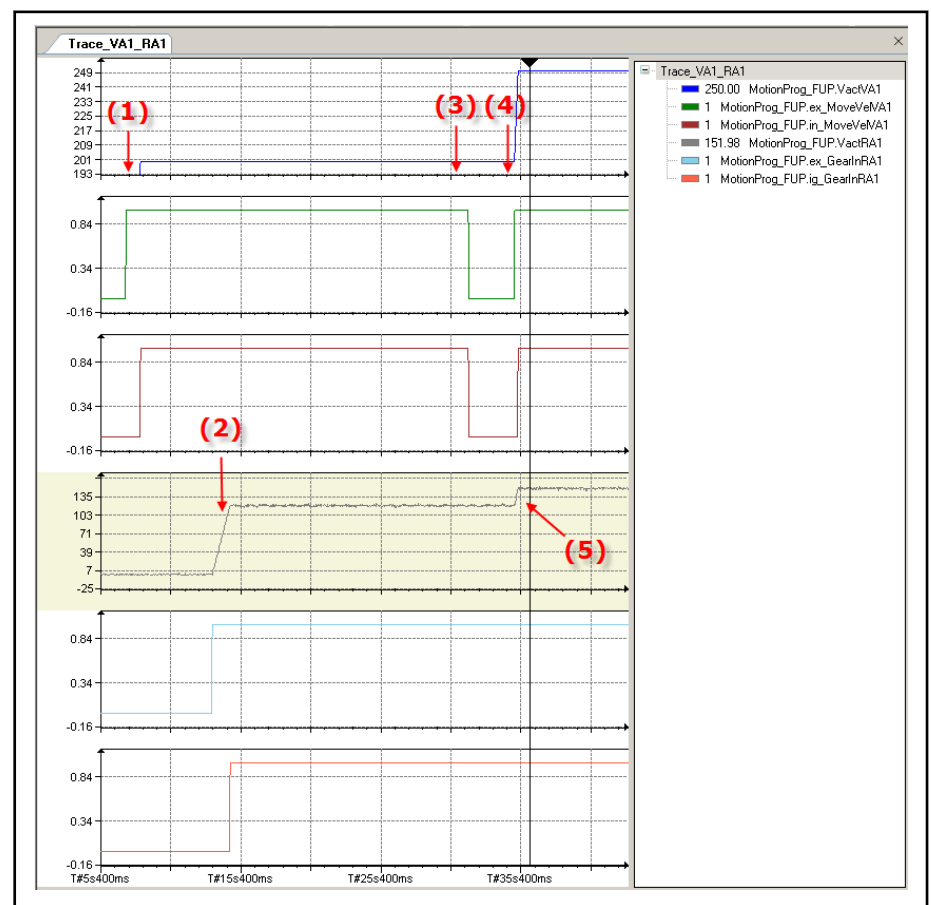


Fig.3-47: Trace recording for both axes

Evaluate the Trace recording

- (1) - Start of virtual axis startup
- (2) - Enable gear function, real axis startup
- (3) - Increase command speed of virtual axis, "ex_MoveVelVA1" TRUE $\Rightarrow$ FALSE
- (4) - Enable command speed of virtual axis, "ex_MoveVelVA1" FALSE $\Rightarrow$ TRUE
- (5) - Jump response on real axis

First Steps - Motion

3.5 Programming a 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).

During a project creation with the control Mlc1, the real axis RA1 and the virtual axis VA1, first programming activities of the motion sequence have been started below the "Logic" folder:

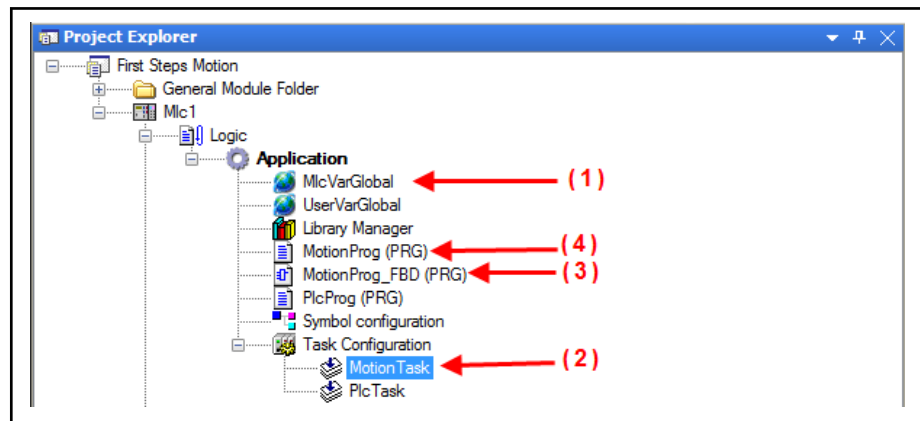


Fig.3-48: Project Explorer after expanding the "Logic" node

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 mouse button, **Add** ► **POU**, Entering the name and selecting FBD as programming language.

The programs are currently still empty.

Under "Task Configuration", the following tasks were created:

- **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 part of the sercos cycle. If the tasks are not processed within the specified part of the sercos cycle, 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 for:

- **UserVarGlobal**, reserved for the user, not used in "First Steps - Motion"
- **MlcVarGlobal (1)**, contains the address constants for the axes RA1 and VA1.

Program:

```
(* Automated generated code by MLC. *)
(* Please don't edit, the code will be overwritten...*)
VAR_GLOBAL CONSTANT
(*virtual axis "VA1" with axis number 2*)
VA1: AXIS_REF :=(CntrlNo:=LOCAL_CNTRL,AxisNo:=AXIS_2);
(*real axis "RA1" with axis number 1*)
RA1: AXIS_REF :=(CntrlNo:=LOCAL_CNTRL,AxisNo:=AXIS_1);
```

(*...*)
END_VAR



Only the Motion task is used for the "First Steps Motion" project. According to the programming language required, either **MotionProg** or **MotionProg_FBD** is assigned.

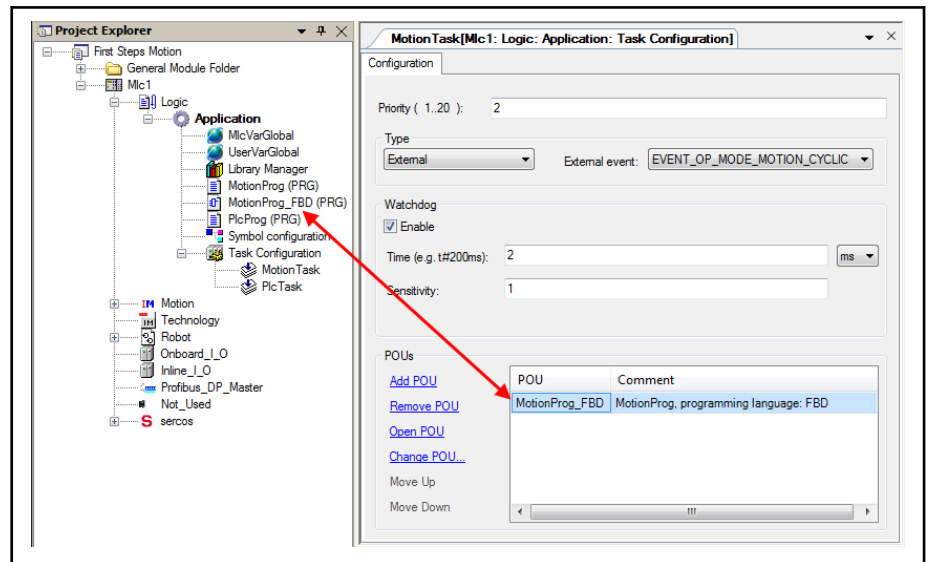


Fig.3-49: Delete call: MotionProg_FBD; add call: MotionProg

3.5.2 Programming the Virtual Axis VA1, Programming Language ST



This section uses ST (Structured Text) as the programming language. From the functional point of view, the same result as in the sections with the programming language FBD occurs.

Similarly to [Project Goal First Steps - Motion, page 67](#), instances of the PLCopen blocks MC_MoveVelocity and MC_Stop are called using "VA1".



The MC_Power instance is not required for virtual axes.

Proceed as follows:

1. Assign the MotionTask to the "MotionProg" program.
2. Open the "MotionProg" program (by double-clicking on it).

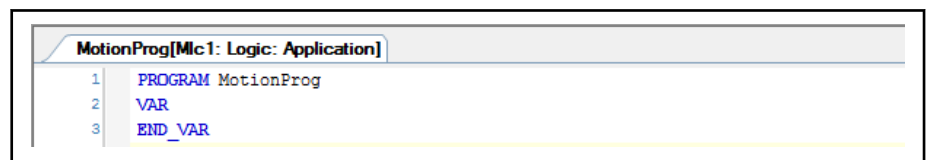


Fig.3-50: New program in ST (Structured Text)

Unlike when programming in FBD, the instances of the function blocks in the declaration section VAR ... END_VAR, must firstly be declared in ST (Structured Text).

Only then can they be used in the implementation.

First Steps - Motion

Firstly in section VAR ... END_VAR, enter the name of the instance of the function block MC_MoveVelocity "MoveVelVA1". Call up the context menu by right-clicking on it and select Input Assistant (alternatively <F2> window).

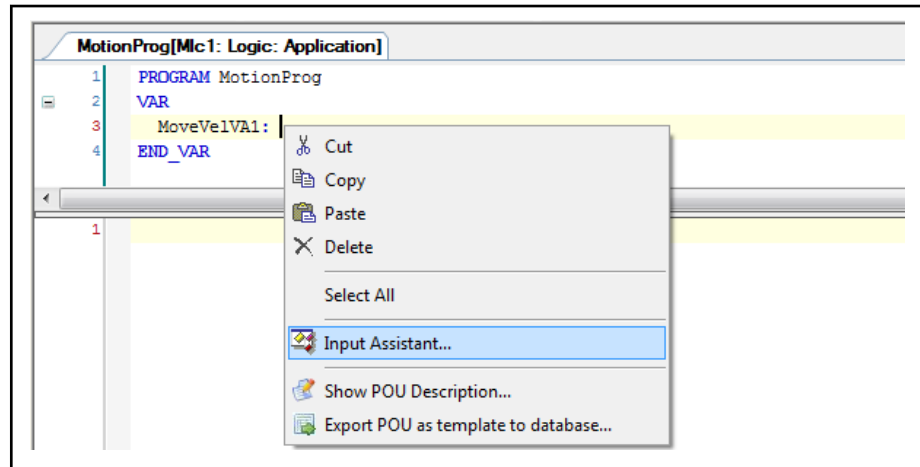


Fig.3-51: Declaration of the instance MoveVelVA1, MC_MoveVelocity type

To do so, select the following in the open selection window:

User-defined Types

► ML_PLCopen ► POU ► Motion ► MC_MoveVelocity.

Select the option "Add with arguments".

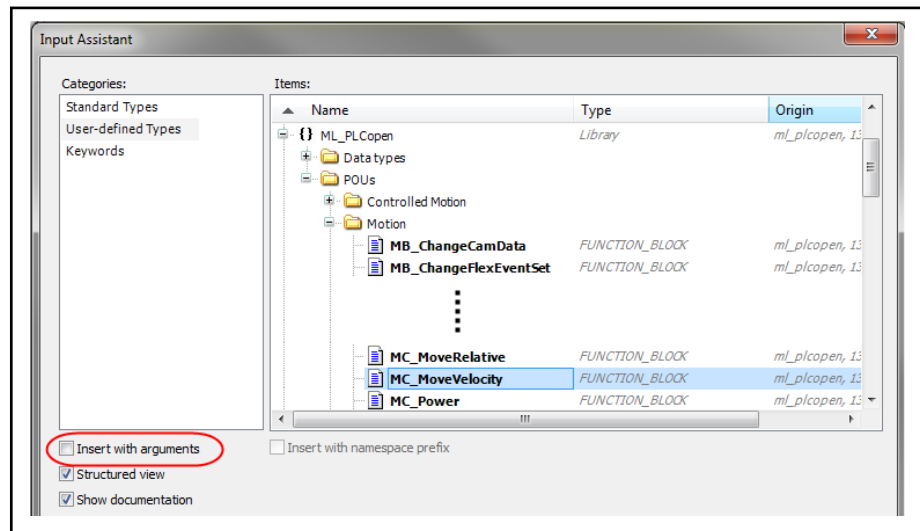


Fig.3-52: Selection window "Function Blocks"

After confirming with **OK**, complete the line with a ";".

Go to line 1 in the implementation section of the Editor and open the Input Assistant with <F2>.

Select the following in the opened window **Instance Calls** ► **MoveVelVA1**.

Select the option "Add with arguments".

First Steps - Motion

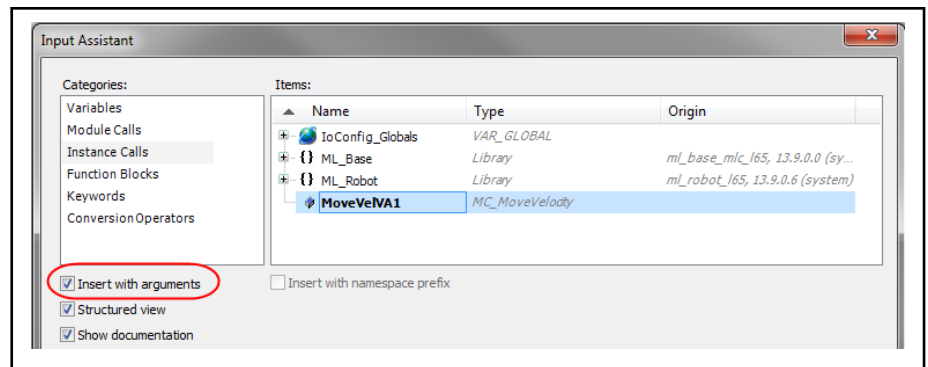


Fig.3-53: Transfer of the instance into the implementation

Confirm the insertion of the instance with OK.

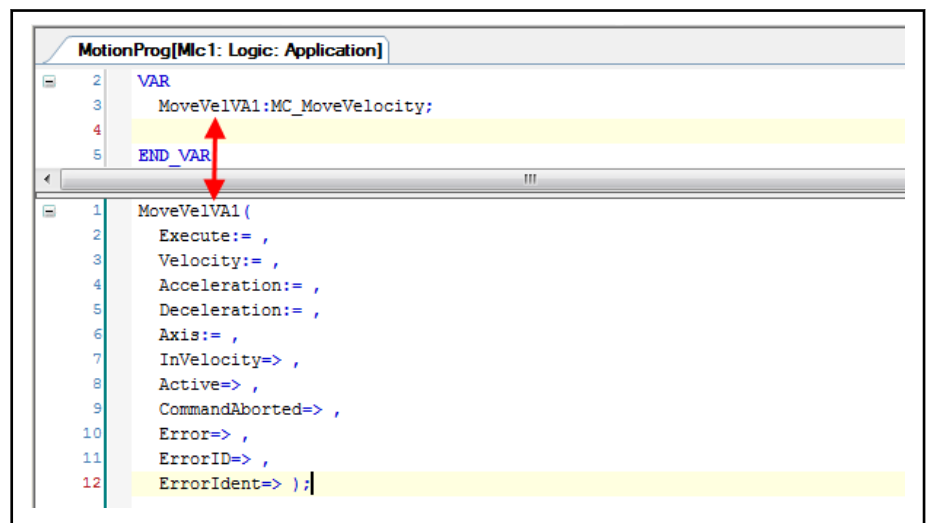


Fig.3-54: Broad design of a function block instance

Assign the FB inputs.

Begin with the input "Execute":

- Enter the name "ex_MoveVelVA1" and complete with <Enter>
- The open window "Variable Declaration" can be confirmed immediately with <Enter>

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

- Enter the name "VcomVA1" and confirm using <Enter>
- Enter REAL as type and 200 (rpm) as initial value. Press <Enter> to confirm

For the "Acceleration" inputs, please enter 20 (rad/s²) and for "Deceleration" 20 (rad/s²).



Enter the numbers directly onto the function block inputs.

Alternatively a variable with the respective value can be used.

To assign the "Axis" input, execute the following:

- Right-click to open the input assistant:
Select **Variables** ► **Application** ► **MlcVarGlobal** ► **VA1**.

First Steps - Motion

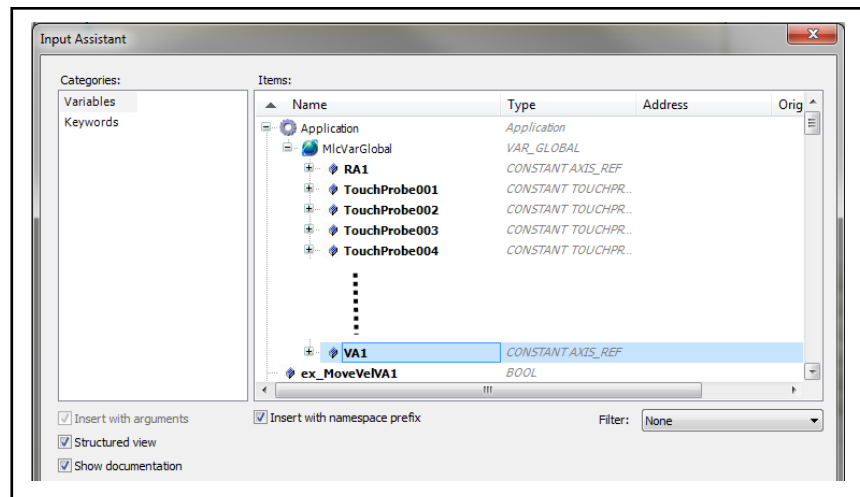


Fig.3-55: Selecting the axis instance when assigning a function block of the type "MC_MoveVelocity"

- Confirm using **OK**.



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

`MlcVarGlobal.VA1`

To obtain feedback about whether the function block has worked correctly, its outputs are assigned.

- Assign the "InVelocity" output of the function block using the "iv_MoveVelVA1" variable, type BOOL
- Enter the name "iv_MoveVelVA1" and confirm using <Enter>
- The open window "Variable Declaration" can be confirmed immediately with <Enter>
- Assign "ca_MoveVelVA1" to the CommandAborted output
- Assign "er_MoveVelVA1" to the Error output
- Assign "ID_MoveVelVA1" to the ErrorID output
- Assign "Ident_MoveVelVA1" to the "ErrorIdent" output.

The PLCopen FB, type MC_MoveVelocity, is thus assigned according to its task. The declaration section was automatically completed.

```

2  VAR
3  MoveVelVal: MC_MoveVelocity;
4  ex_MoveVelVal: BOOL;
5  VcomVal: REAL := 200;
6  iv_MoveVelVal: BOOL;
7  ac_MoveVelVal: BOOL;
8  ca_MoveVelVal: BOOL;
9  er_MoveVelVal: BOOL;
10 ID_MoveVelVal: ERROR_CODE;
11 Ident_MoveVelVal: ERROR_STRUCT;
12 END VAR

1  MoveVelVal(
2  Execute:= ex_MoveVelVal,
3  Velocity:= VcomVal,
4  Acceleration:= 20,
5  Deceleration:= 20,
6  Axis:= MlcVarGlobal.VA1,
7  InVelocity=> iv_MoveVelVal,
8  Active=> ac_MoveVelVal,
9  CommandAborted=> ca_MoveVelVal,
10 Error=> er_MoveVelVal,
11 ErrorID=> ID_MoveVelVal,
12 ErrorIdent=> Ident_MoveVelVal);

```

Fig.3-56: 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 FB MC_Stop.

MC_Stop

Declare and insert instance "StopVA1" of MC_Stop.

Assign inputs and declare input variables if necessary:

"Execute" → "ex_StopVA1" of type BOOL

"Deceleration" → 50

"Axis" → "MlcVarGlobal.VA1" of type "AXIS_REF"

Outputs:

"Done" → "do_StopVA1" of type BOOL

"Active" → "ac_StopVA1" of type BOOL

"CommandAborted" → "ca_StopVA1" of type BOOL

"Error" → "er_StopVA1" of type BOOL

"ErrorID" → "ID_StopVA1" of type ERROR_CODE

"ErrorIdent" → "Ident_StopVA1" of type ERROR_STRUCT

Actual velocity "VactVA1"

The final step is the implementation of a network in which the current velocity of axis VA1 "VactVA1" (data type REAL) can be read.

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

The data from all axes that can control a control is located in the array

GVL_Base.AxisData: ARRAY [1..xx³⁾] OF ML_AXISDATA_SM

where the axis number is used as the index.

The variable is provided with the name

³⁾ The number of controllable axes depends on the control type and its firmware, currently for the L65: 64 axes

First Steps - Motion

GVL_Base.AxisData[2].rActualVelocity_i

for the virtual axis, axis number 2, (axis number 1 has RA1, since scanning RA1 was executed before declaring VA1).

- Insert new line
- Insert VactVA1:= , complete declaration
- using the input assistance to navigate to **Variables** ► **ML_BASE** ► **Global Variable List** ► **GVL_Base** ► **AxisData**:

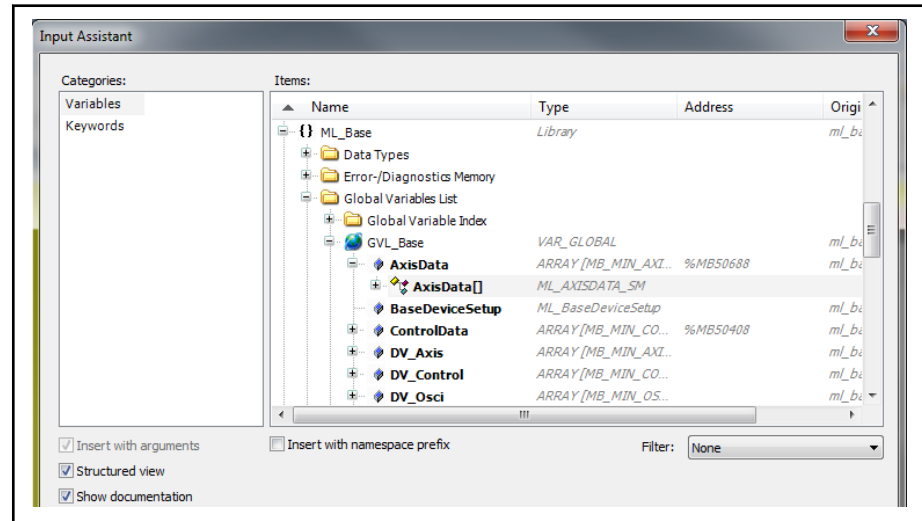


Fig.3-57: Actual velocity value VA1 from the global variable list of the library ML_Base.library

Input is supported by the Intellisense function; i.e. all available elements are opened and can be selected after entering "." for "AxisData[2]."

Confirm "rActualVelocity_i" with <Enter>.

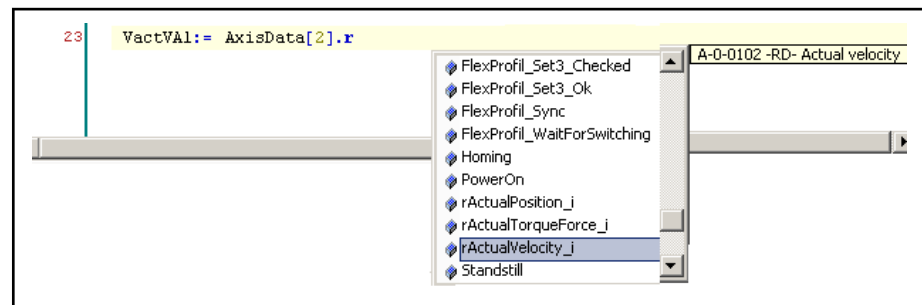


Fig.3-58: Selecting the variables "ActualVelocity"

- Do not forget to complete the line with ";".

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

```

1  PROGRAM MotionProg
2  VAR
3  MoveVelVal: MC_MoveVelocity;
4  ex_MoveVelVal: BOOL;
5  VcomVal: REAL := 200;
6  iv_MoveVelVal: BOOL;
7  ac_MoveVelVal: BOOL;
8
9  MoveVelVal(
10     Execute:= ex_MoveVelVal,
11     Velocity:= VcomVal,
12     Acceleration:= 20,
13     Deceleration:= 20,
14     Axis:= MlcVarGlobal.Val,
15     InVelocity=> iv_MoveVelVal,
16     Active=> ac_MoveVelVal,
17     CommandAborted=> ca_MoveVelVal,
18     Error=> er_MoveVelVal,
19     ErrorID=> ID_MoveVelVal,
20     ErrorIdent=> Ident_MoveVelVal);
21
22 StopVal(
23     Execute:= ex_StopVal,
24     Deceleration:= 50,
25     Axis:= MlcVarGlobal.Val,
26     Done=> do_StopVal,
27     Active=> ac_StopVal,
28     CommandAborted=> ca_StopVal,
29     Error=> er_StopVal,
30     ErrorID=> ID_StopVal,
31     ErrorIdent=> Ident_StopVal);
32
33 VactVal:= AxisData[2].rActualVelocity_i;

```

Fig.3-59: Implementing the virtual axis VA1



AxisData[].rActualVelocity_i; and
GVL_Base.AxisData[].rActualVelocity_i;
are equivalently used by the compiler.

3.5.3 Downloading the PLC Sub-project Virtual Axis VA1 (ST)

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

The build can be accessed via the IndraWorks menu bar

- **Create ► Build**
Build of the added part of the application
- **Create ► Rebuild**
Build of the complete application

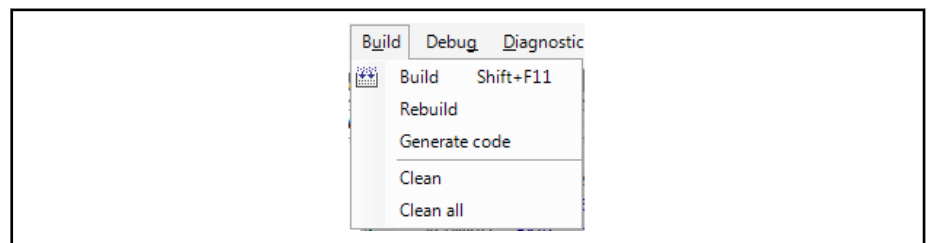


Fig.3-60: Build application

To apply the compiled data to the target control, select the IndraWorks menu item **Debug ► Login** (see the following figure).

First Steps - Motion

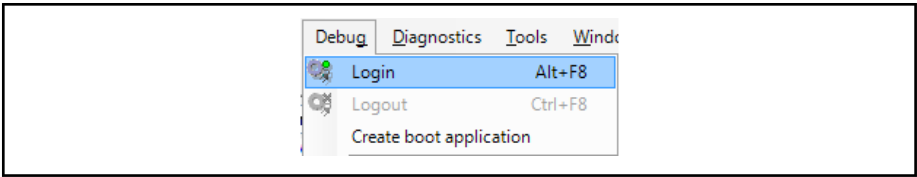


Fig.3-61: Transferring compiled data to the control

A prompt asking for agreement to download is displayed depending on whether an application is already loaded and whether this application is a predecessor of its 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 not valid.

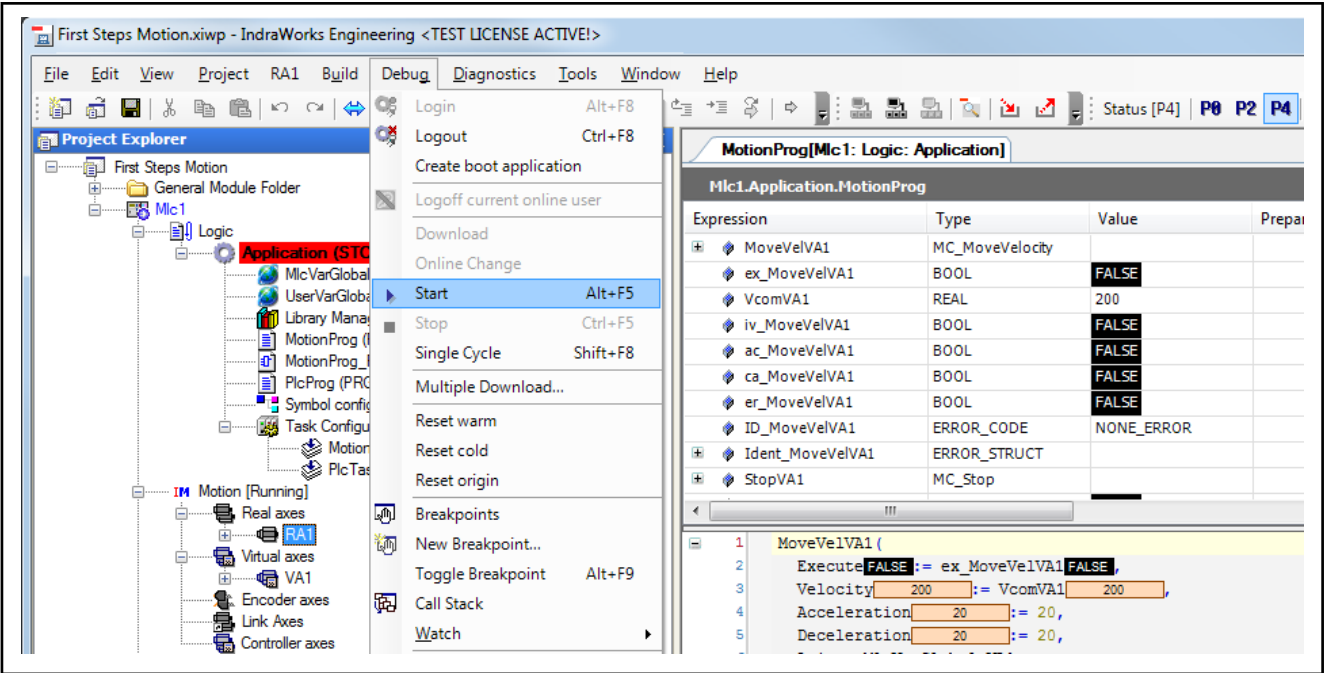


Fig.3-62: Starting the program processing in the control (1)

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

First Steps - Motion

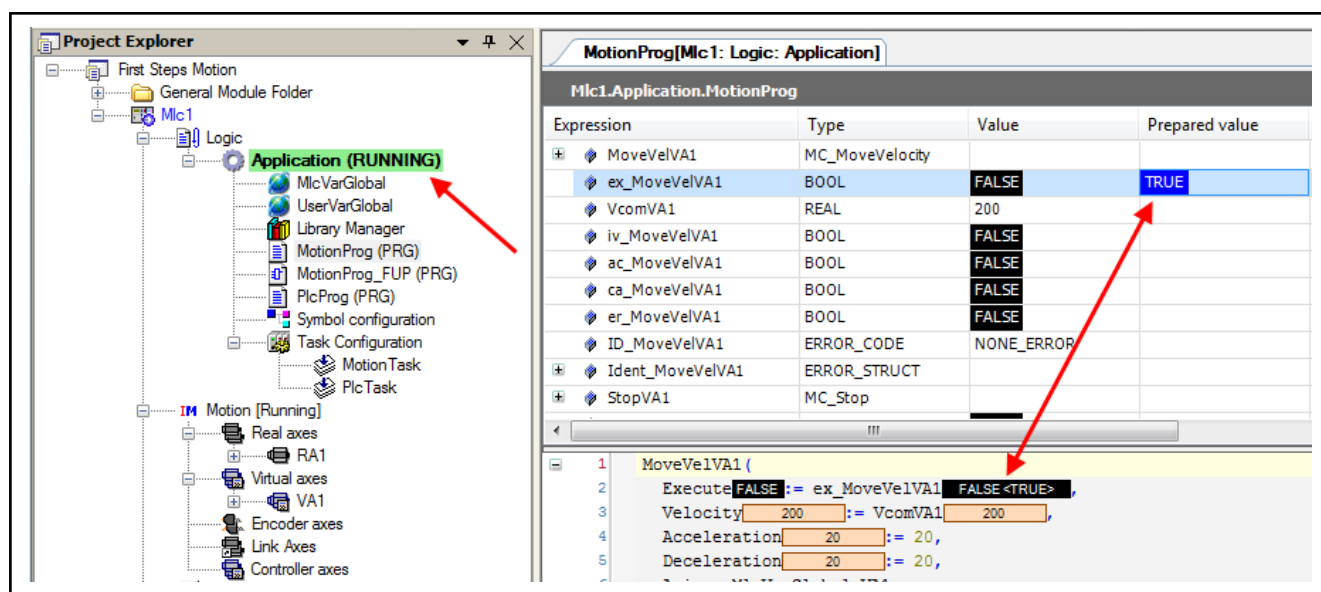


Fig.3-63: Starting the program processing in the control (2)

The motion process starts automatically since the processing of the programmed function blocks is activated one after the other.

"Write values", input variables

To enable processing of "MoveVelVA1" of type "MC_MoveVelocity", a positive edge must occur at the "Execute" input of the FB.

Click on FALSE with the left mouse button behind the input variable "ex_MoveVelVA1". The "Prepare Value" window opens. Enter "TRUE" and confirm.

Now, a highlighted "FALSE<TRUE>" is behind "ex_MoveVelVA1" and "TRUE" is displayed in the declaration line.

First Steps - Motion

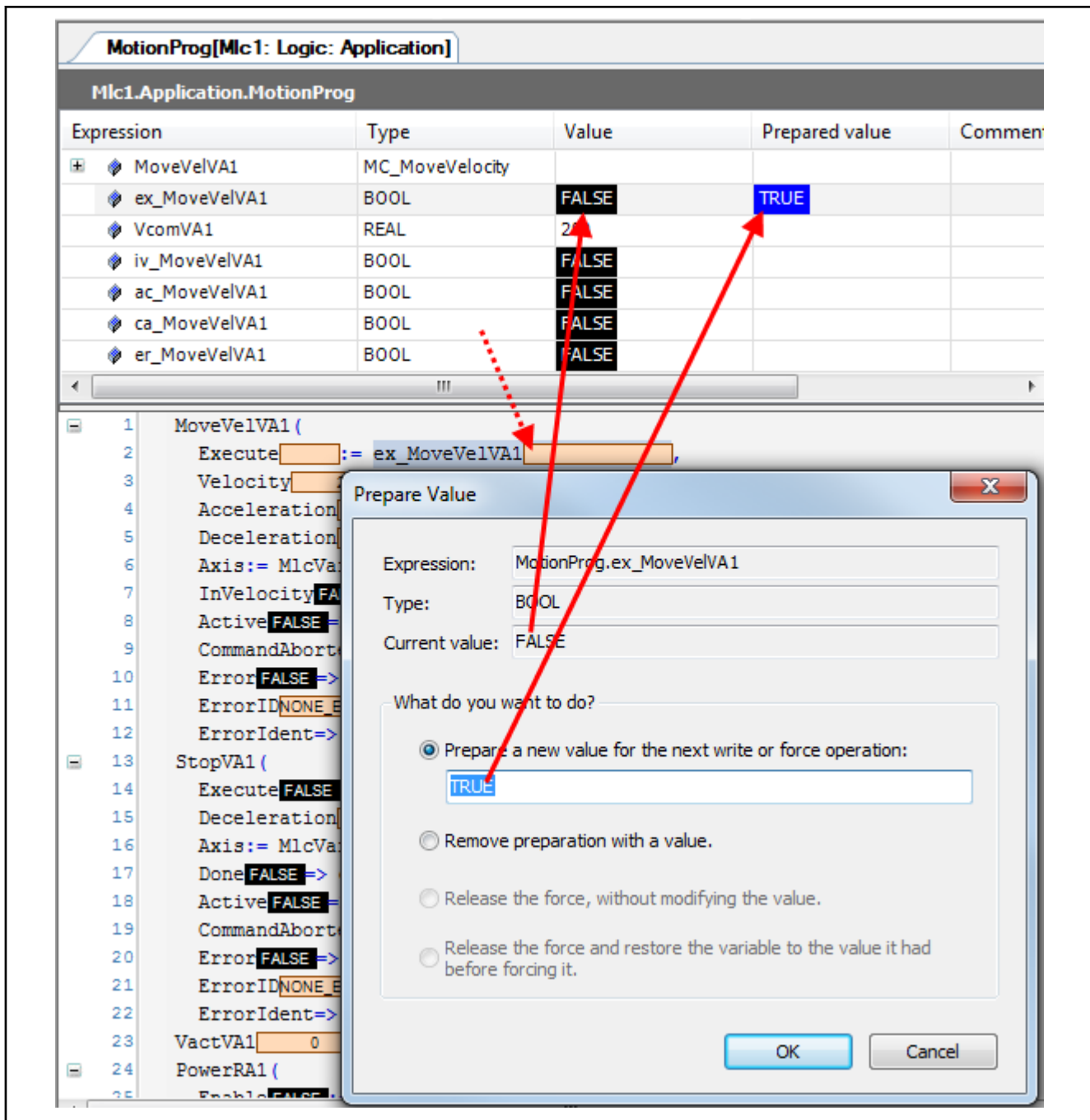


Fig.3-64: Preparation of "Write values"

Via **Debug ► Write Values** <Ctrl>+<F7>, "ex_MoveVelVA1" in the control is set to "TRUE".



- "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 created sets the outputs "InVelocity" and "Active" of the "MoveVelVA1" function block to "TRUE" (1) (see the following image).

The axis travels at 200 rpm (2).

First Steps - Motion

MotionProg[MLC1: Logic: Application]

MLC1.Application.MotionProg

Expression	Type	Value	Prepared value	Comment
MoveVelVA1	MC_MoveVelocity			
ex_MoveVelVA1	BOOL	TRUE		
VcomVA1	REAL	200		
iv_MoveVelVA1	BOOL	TRUE		
ac_MoveVelVA1	BOOL	TRUE		
ca_MoveVelVA1	BOOL	FALSE		
er_MoveVelVA1	BOOL	FALSE		

```

1  MoveVelVA1 (
2    Execute TRUE := ex_MoveVelVA1 TRUE,
3    Velocity 200 := VcomVA1 200,
4    Acceleration 20 := 20,
5    Deceleration 20 := 20,
6    Axis:= MlcVarGlobal.VA1,
7    InVelocity TRUE => iv_MoveVelVA1 TRUE,
8    Active TRUE => ac_MoveVelVA1 TRUE,
9    CommandAborted FALSE => ca_MoveVelVA1 FALSE,
10   Error FALSE => er_MoveVelVA1 FALSE,
11   ErrorID NONE_ERROR => ID_MoveVelVA1 NONE_ERROR,
12   ErrorIdent=> Ident_MoveVelVA1);
13 StopVA1 (
14   Execute FALSE := ex_StopVA1 FALSE,
15   Deceleration 50 := 50,
16   Axis:= MlcVarGlobal.VA1,
17   Done FALSE => do_StopVA1 FALSE,
18   Active FALSE => ac_StopVA1 FALSE,
19   CommandAborted FALSE => ca_StopVA1 FALSE,
20   Error FALSE => er_StopVA1 FALSE,
21   ErrorID NONE_ERROR => ID_StopVA1 NONE_ERROR,
22   ErrorIdent=> Ident_StopVA1);
23 VactVA1 200 := GVL_Base.AxisData[2].rActualVelocity_i 200;
  
```

Fig.3-65: After "Write Values" at the "Execute" input

Switch "ex_MoveVelVA1" again to "FALSE". The axis continues traveling at 200 rpm ②.

By setting the variable "x_StopVA1" at the "Execute" input of the "StopVA1" function block of type MC_Stop, the axis motion can be stopped–.

Stop the PLC and log out.

3.5.4 Recording the Motion of Axis VA1 (Trace), ST

IndraLogic provides a Trace tool to trace motion processes. The following has to be recorded using this tool:

- the axis velocity "VactVA1"
- for "MoveVelVA1" (MC_MoveVelocity)
 - "ex_MoveVelVA1" (Execute)
 - "in_MoveVelVA1" (InVelocity)
 - "ca_MoveVelVA1" (CommandAborted)

and

- for "StopVA1" (MC_Stop)
 - "ex_StopVA1" (Execute)

To do so, the trace recording must first be configured.

Insert the object "Trace_VA1" via the context menu of "Application" **Add ► Trace....** Open the object by double clicking on it.

First Steps - Motion

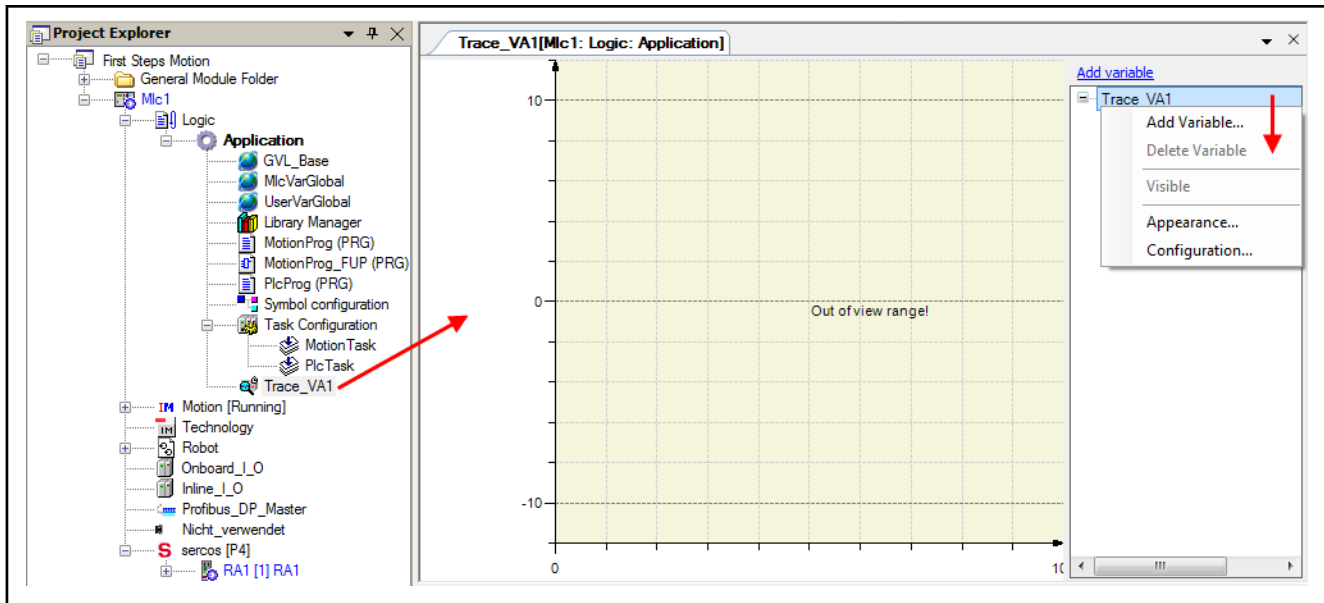


Fig.3-66: Dialog: New Trace object "Trace_VA1" with context menu

The following settings can be selected for the Trace object via the context menu on the right side of the page:

- Configuration
- Display settings (appearance)
- Add new variable

Configuration...

The new Trace object must firstly be configured:

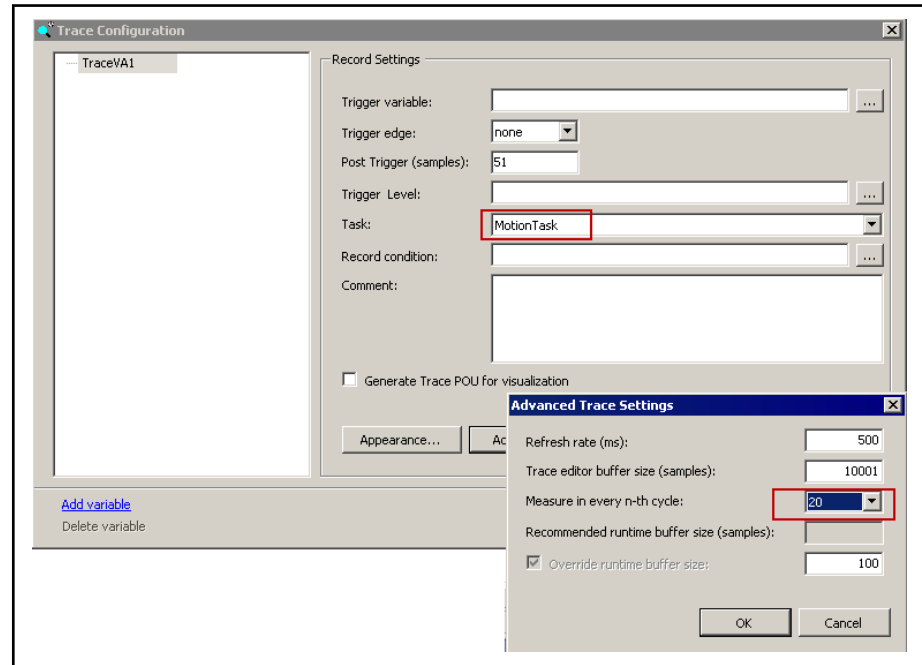


Fig.3-67: Dialog: Trace configuration with extended Trace setting

The required settings are

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

Task: From the list of available tasks, select the the task for which the values of trace variables are to be read after they are processed. Default: First task in the task configuration tree.

Record condition: A Boolean variable, a value or a Boolean expression can be entered here. As soon as this condition is met, or receives the value TRUE, value recording begins. If nothing is entered here, value recording starts as soon as the trace configuration is loaded into the control and the application is running.

Measurement: Enter the number of cycles that are to elapse before the value of the trace variables is to be read again. Using the formula (Buffer size * Record in every x-th cycle * Task interval), the minimum recording time frame can be estimated.

Recommended runtime buffer size: Size of the ring buffer in bytes, which is currently provided in the runtime system for collecting trace variable values in the currently configured record. The buffer content is then displayed as a value curve or curves in the trace window. Default: 100.



If the configuration changes, the Trace object must be logged into again.

Display settings (appearance)

Settings for the display can now be entered. These settings are also effective even without logging in again.

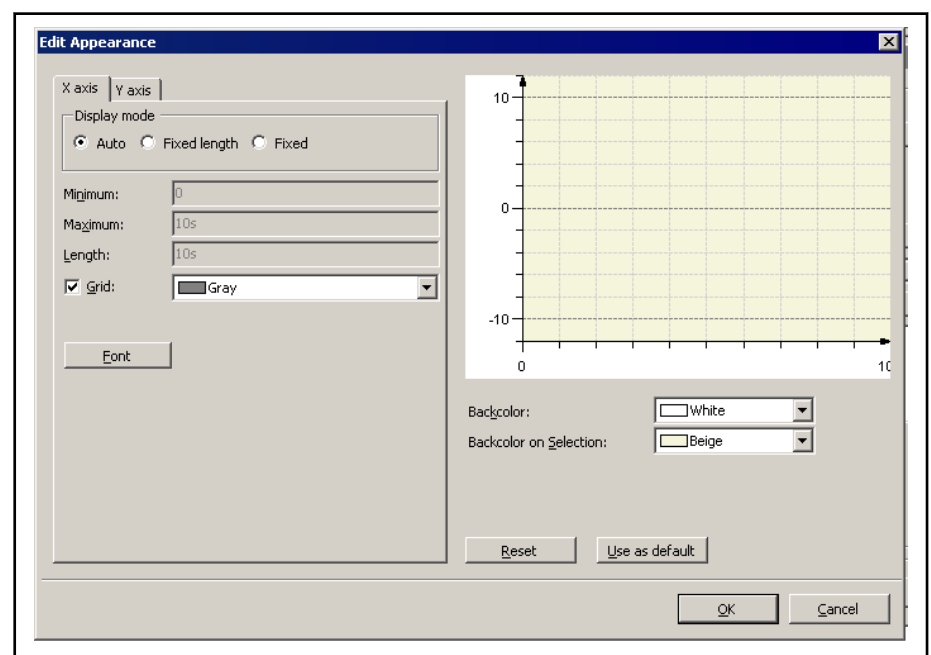


Fig. 3-68: Dialog: Display settings (appearance)

Select **Auto**, positioning a **Length** 10s window over the curve. The **Scale division** is set out in a 1s grid with a **scale division** of 1s.

Selecting the new variables

Now the variables have to be named and their appearance has to be selected.

First Steps - Motion

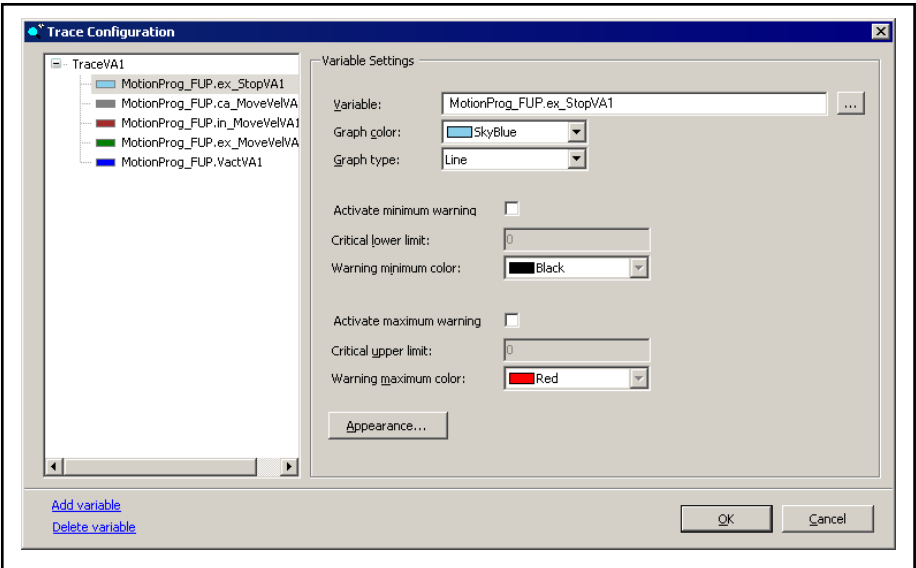


Fig.3-69: Dialog: Creating new variables and adapting appearance

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:

The context menu is displayed by right-clicking into the left section of the Trace window.

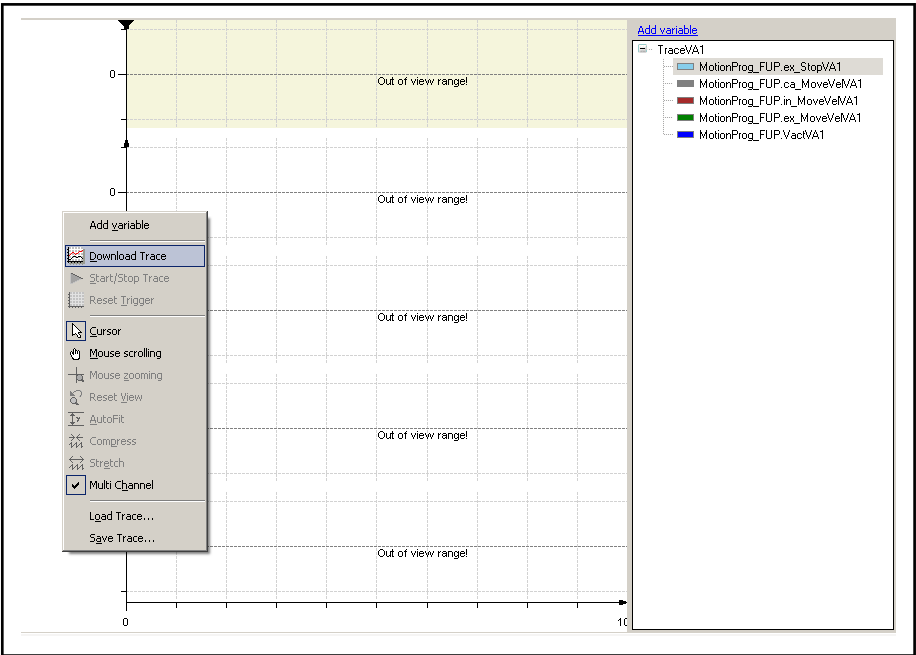


Fig.3-70: Menu to control the Trace recording

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

Start/Stop Trace: This command starts or stops the display of the Trace recording.

If the recording is currently stopped, the  icon is positioned before the command.

If the recording is currently running, the  icon is positioned before 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 enables the view of the Trace curve to be changed. 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 with 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.

Mouse scrolling: In this mode, the 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.

Default appearance: This command can be used to reset the display of the recording(s) to the default values after they have been changed by zooming.

Performing the Trace:

1. Logging 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" trace context menu.
The Trace recording then starts up automatically.
3. Write values.
The following values are set identically to ["Write Values" of input variables", page 109](#).
 - ex_MoveVelVA1 to TRUE
 - ex_StopVA1 to TRUE
4. End the recording.
Stop the Trace via the Trace context menu item "Start/Stop Trace".
5. Evaluate the Trace recording.
Set the window data (refer to the screen below).

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

First Steps - Motion

Shortcuts	Action in the trace diagram	Mouse control
<↑/↓> 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

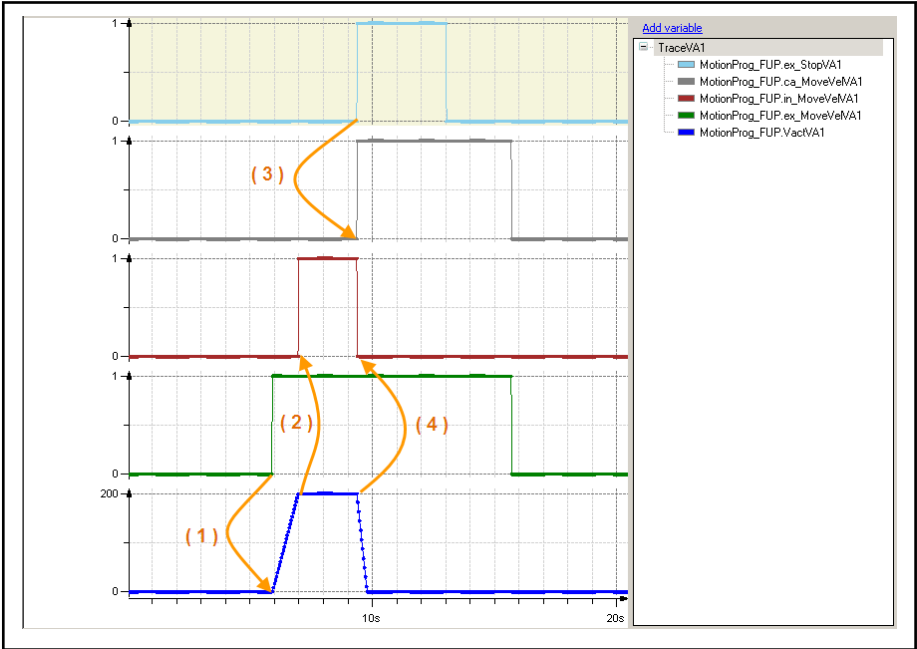


Fig.3-71: Trace recording for the virtual axis VA1

Evaluate the Trace recording

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

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

3.5.5 Programming the Real Axis RA1 (ST)



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

The use of a **real** axis firstly requires an instance of the PLCopen function block MC_Power.

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

The real axis is stopped with an instance of MC_Stop.

The programming is executed similarly to in section [Programming the Virtual Axis VA1, Programming Language FBD, page 79](#).

MC_Power

Adding new line

Insert MC_Power and declare a "PowerRA1" instance

Assign inputs and declare input variables if necessary:

"Enable" → "en_PowerRA1" of type BOOL

"Axis" → "MlcVarGlobal.RA1" of type "AXIS_REF"

Outputs:

"Status" → "st_PowerRA1" of type BOOL

"Error" → "er_PowerRA1" of type BOOL

"ErrorID" → "ID_PowerRA1" of type ERROR_CODE

"ErrorIdent" → "Ident_PowerRA1" of type ERROR_STRUCT

MC_GearIn

Adding new line

Insert MC_GearIn and declare the "GearInRA1" instance

Assign inputs and declare input variables if necessary:

"Execute" → "ex_GearInRA1" of type BOOL

"RatioNumerator" → 5

"RatioDenominator" → 10

"MasterFineadjust" → 20 (%)

"Master" → "MlcVarGlobal.VA1" of type "AXIS_REF"

"Slave" → "MlcVarGlobal.RA1" of type "AXIS_REF"

Outputs:

"InGear" → "ig_GearInRA1" of type BOOL

"Active" → "ac_GearInRA1" of type BOOL

"CommandAborted" → "ca_GearInRA1" of type BOOL

"Error" → "er_GearInRA1" of type BOOL

First Steps - Motion

"ErrorID" → "ID_GearInRA1" of type ERROR_CODE

"ErrorIdent" → "Ident_GearInRA1" of type ERROR_STRUCT

MC_GearOut

Adding new line

Insert MC_GearOut and declare a "GearOutRA1" instance

Assign inputs and declare input variables if necessary:

"Execute" → "ex_GearOutRA1" of type BOOL

"Slave" → "MlcVarGlobal.RA1" of type "AXIS_REF"

Outputs:

"Done" → "do_GearOutRA1" of type BOOL

"Active" → "ac_GearOutRA1" of type BOOL

"Error" → "er_GearOutRA1" of type BOOL

"ErrorID" → "ID_GearOutRA1" of type ERROR_CODE

"ErrorIdent" → "Ident_GearOutRA1" of type ERROR_STRUCT

MC_Stop

Adding new line

Insert MC_Stop and declare a "StopRA1" instance

Assign inputs and declare input variables if necessary:

"Execute" → "ex_StopRA1" of type BOOL

"Deceleration" → 50 (rad/s²)

"Axis" → "MlcVarGlobal.RA1" of type "AXIS_REF"

Outputs:

"Done" → "do_StopRA1" of type BOOL

"Active" → "ac_StopRA1" of type BOOL

"CommandAborted" → "ca_StopVA1" of type BOOL

"Error" → "er_StopRA1" of type BOOL

"ErrorID" → "ID_StopRA1" of type ERROR_CODE

"ErrorIdent" → "Ident_StopRA1" of type ERROR_STRUCT

Actual velocity "VactRA1"

The final step is the implementation of a network in which the current velocity of axis RA1 "VactRA1" (data type REAL) can be read.

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

The data from all axes that can control an MLC is located in the array

GVL_Base.AxisData: ARRAY [1..xx⁴] OF ML_AXISDATA_SM

where the axis number is used as the index.

The variable is provided with the name

GVL_Base.AxisData[1].rActualVelocity_i

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

First Steps - Motion

for the real axis, axis number 1, (RA1 has axis number 1, since scanning RA1 was executed before declaring VA1).

- Insert new line
- Insert `VactRA1:=` , complete declaration
- using the input assistance to navigate to **Global Variables** ► **ML_BASE** ► **Global Variable List** ► **GVL_Base** ► **AxisData**:

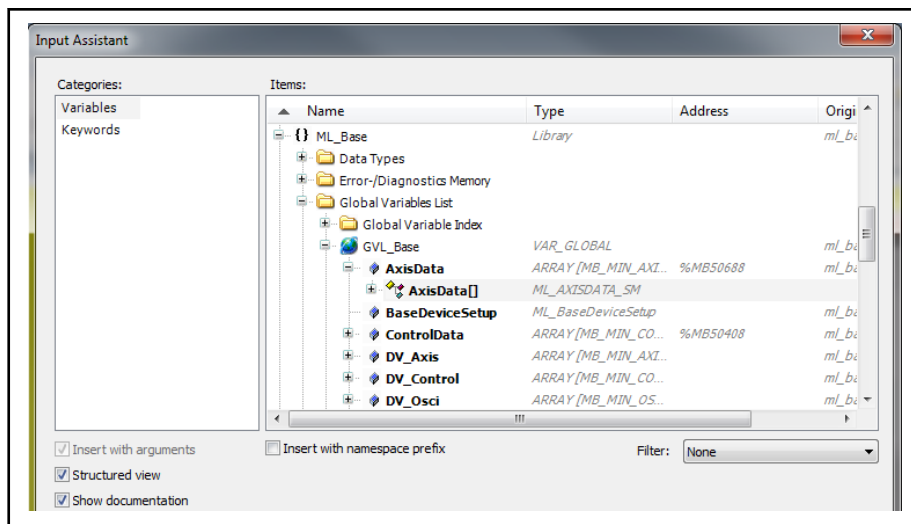


Fig. 3-72: Actual velocity value VA1 from the global variable list of the library *ML_Base.library*

Input is supported by the Intellisense function; i.e. all available elements are opened and can be selected after entering "." for "GVL_Base.AxisData[1]."

Confirm "rActualVelocity_i" with <Enter>.

- Do not forget to complete the line with ";".

First Steps - Motion

```

25   PowerRA1( Enable:= en_PowerRA1,
26           Axis:= MlcVarGlobal.RA1,
27           Status=> st_PowerRA1,
28           Error=> er_PowerRA1,
29           ErrorID=> ID_PowerRA1,
30           ErrorIdent=> Ident_PowerRA1);
31
32   GearInRA1( Execute:= ex_GearInRA1,
33            RatioNumerator:= 5,
34            RatioDenominator:= 10,
35            MasterFineadjust:= 20,
36            Master:= MlcVarGlobal.VA1,
37            Slave:= MlcVarGlobal.RA1,
38            InGear=> ig_GearInRA1,
39            Active=> ac_GearInRA1,
40            CommandAborted=> ca_GearInRA1,
41            Error=> er_GearInRA1,
42            ErrorID=> ID_GearInRA1,
43            ErrorIdent=> Ident_GearInRA1);
44
45   GearOutRA1( Execute:= ex_GearOut,
46            Slave:= MlcVarGlobal.RA1,
47            Done=> do_GearOut,
48            Active=> ac_GearOut,
49            Error=> er_GearOut,
50            ErrorID=> ID_GearOut,
51            ErrorIdent=> Ident_GearOut);
52
53   StopRA1( Execute:= ex_StopRA1,
54           Deceleration:= 50,
55           Axis:= MlcVarGlobal.RA1,
56           Done=> do_StopRA1,
57           Active=> ac_StopRA1,
58           CommandAborted=> ca_StopRA1,
59           Error=> er_StopRA1,
60           ErrorID=> ID_StopRA1,
61           ErrorIdent=> Ident_StopRA1);
62
63   VactRA1:= GVL_Base.AxisData[1].rActualVelocity_i;

```

Fig.3-73: Implementing the real axis



AxisData[].rActualVelocity_i; and
GVL_Base.AxisData[].rActualVelocity_i;
are equivalently used by the compiler.

3.5.6 Downloading the Complete PLC Project (ST)

The project is downloaded identically to the description in section [Downloading the PLC Sub-project Virtual Axis VA1 \(ST\)](#), page 107.

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_PowerRA1" (MC_Power) by "Write Values".
st_PowerRA1= TRUE (MC_Power).
2. Start the virtual axis.
Enable "ex_MoveVelVA1" (MC_MoveVelocity) by "Write Values".
Start up ends at VactVA1 = 200 rpm.
3. Trace the real axis.
Enable the "ex_GearInRA1" (MC_GearIn) by "Write Values".
The expected gear transmission ratio is:
 $Tr = (\text{RatioNumerator} / \text{RatioDenominator}) * (1 + \text{MasterFineadjust})$
for VactVA1 = 200 rpm results VactRA1 = 120 rpm.



The MasterFineadjust value is given in "%".

4. If MC_GearIn is "InGear", the velocity "VcomVA1" 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 the "ex_GearInRA1" (MC_GearIn) by "Write Values".
in_GearInRA1= FALSE, ac_GearInRA1= TRUE.
The real axis continues to follow the virtual axis.
6. Terminate synchronicity.
Enable the "ex_GearOutRA1" (MC_GearOut) by "Write Values".
The dependency of the real axis on the virtual axis is suspended. However, the velocity at which both axes continue to rotate remains unchanged.
7. Change the command velocity "VcomVA1".
The changed command velocity only affects the virtual axis. "VactRA1" remains unchanged.
8. Disable "ex_GearOutRA1" by "Write Values".
do_GearOutRA1= FALSE
9. Stop the virtual axis.
Enable "ex_StopVA1" (MC_Stop).
The virtual axis is decelerated to VactVA1 = 0 rpm.
10. Disable "ex_StopVA1" (MC_Stop).
11. Stop the real axis.
Enable "ex_StopRA1" (MC_Stop).
The real axis is decelerated to VactRA1= 0 rpm. After the axis stops, it is actively held in this position.
12. Disable "ex_StopRA1" (MC_Stop).

First Steps - Motion

The active holding of the real axis ends.

13. Power off the real axis.

Disable "en_PowerRA1" (MC_Power) by "Write Values".



Only possible in the "standstill" mode after executing the stop at RA1.

3.5.8 Recording the Motion of Both Axes (Trace), ST

The Trace tool was used, in the same way as in section [Recording the Motion of Axis VA1 \(Trace\), ST, page 111](#), to visualize the startup of the virtual axis and its 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_PowerRA1" (MC_Power).
2. Start the virtual axis.
Enable "ex_MoveVelVA1" (MC_MoveVelocity).
3. Follow the real axis.
Enable "ex_GearInRA1" (MC_GearIn).
4. Speed command value jump of the virtual axis.
Increase the speed command value.
Switch on by "ex_MoveVelVA1" TRUE ⇒ FALSE ⇒ TRUE.

The following figure shows the trace result:

First Steps - Motion

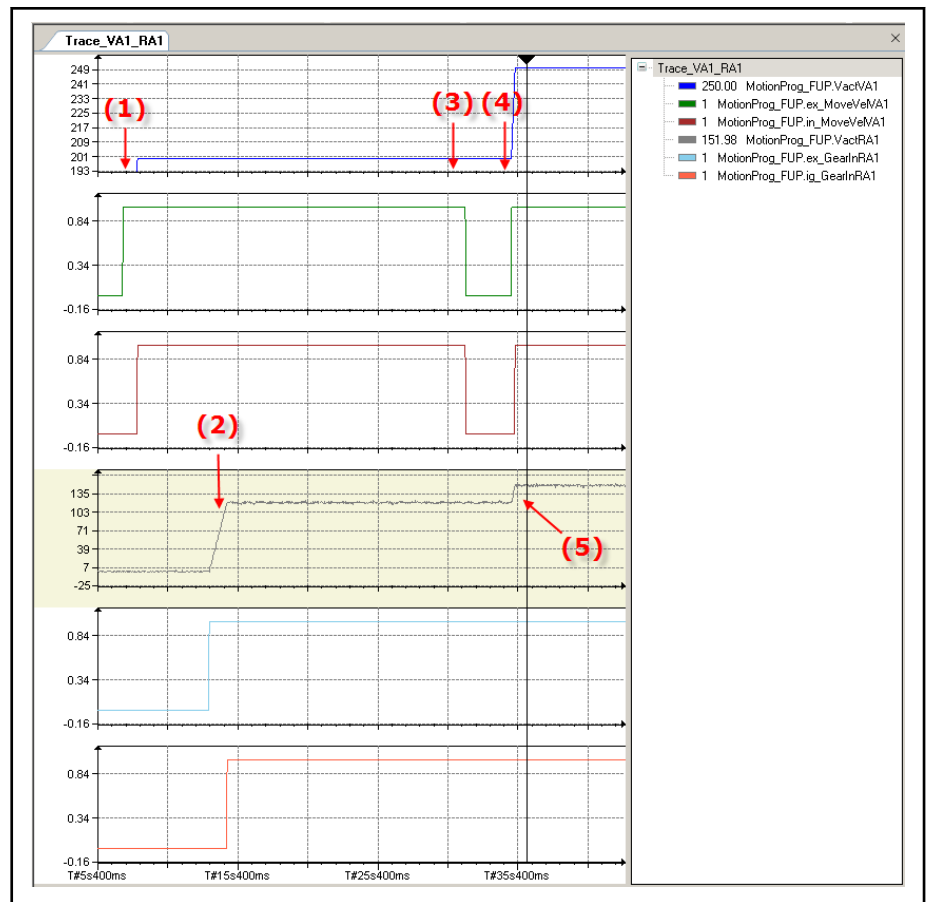


Fig.3-74: Trace recording for both axes

Evaluate the Trace recording

- (1) - Start of virtual axis startup
- (2) - Enable gear function, real axis startup
- (3) - Increase command speed of virtual axis, "ex_MoveVelVA1" TRUE ⇒ FALSE
- (4) - Enable command speed of virtual axis, "ex_MoveVelVA1" FALSE ⇒ TRUE
- (5) - Jump response on real axis

3.6 Restoring the Project

Location Based on the project created, an export file was generated enabling parts of the project to be restored irrespective of its control hardware e.g. IndraLogic XLC L25, L45, L65 (with sercos III bus) or IndraMotion MLC L25, L45, L65 and the drive engineering.

The export file has been created for the "Application" node and contains the following components:

First Steps - Motion

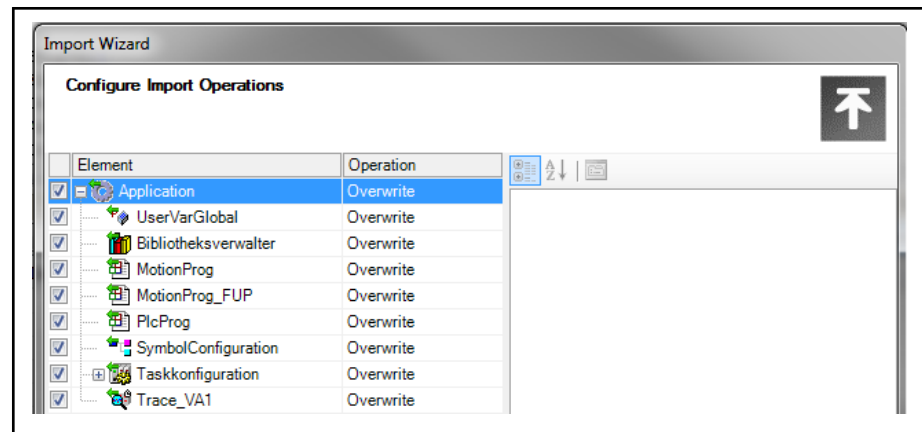


Fig.3-75: Components of the export file

The given components are overwritten during import.



The export file can be used as the basis for "First Steps - Flex-Profile".

Restore

To ensure that the project runs on a control (IndraLogic XLC L25, L45, L65 (with or without sercos III bus) or IndraMotion MLC L25, L45, L65) proceed as follows:

1. Clear old data in the control.
Clear the old memory content of the control.
Details are described under [Resetting the Control \(CMLx5\)](#), page 68,.
The memory of the control is cleared and any possible Motion components present and the PLC components are ready for operation.
2. Create a new project.
Create a new empty project (**File ► New ► Project**). The project name can be chosen at random, although "First Steps Motion" is recommended.
A new empty project has been created.
3. Creating the control.
Drag a control, e.g. IndraLogic XLC L25, L45, L65 (with sercos III bus) or IndraMotion MLC L25, L45, L65 from the device library and drop it to the empty project. The control name can be chosen at random, although "Mlc1" is recommended.
Complete device type and IP address.
Perform a connection test.
End adding as usual.
The desired control is now contained in the project.
4. Creating 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 renamed in "RA1".
5. Declare a virtual axis. The axis is assigned the axis number "2" and the name "VA1".

First Steps - Motion

In the Global Variable List "MlcVarGlobal", the axis numbers and names are provided as axis references of the "AXIS_REF" type.
The motion subobject cannot be downloaded (see [chapter 3.3.8 "Downloading the Motion Subproject" on page 75](#)).

6. Import the export file "Application_FirstSteps_Motion.iwx".

In the Project Explorer, expand the control node to **Logic ► Application**.

Select **Import...** in the context menu of the "Application" node.

Select the export file "Application_FirstSteps_Motion.iwx" in "<Drive>:\All Programs\Rexroth\IndraWorks\Projects". In the context menu opens the "Configure import actions" dialog. Select "Overwrite" for each element in the import dialog. The import is started using **Finish**.

The required files were copied. The project can now be run on the control.

4 First Steps - FlexProfile

4.1 Project Goal "First steps - FlexProfile"

A virtual axis that is supposed 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 traveled on a master axis revolution of 360° . The following figure shows the respective motion diagram.

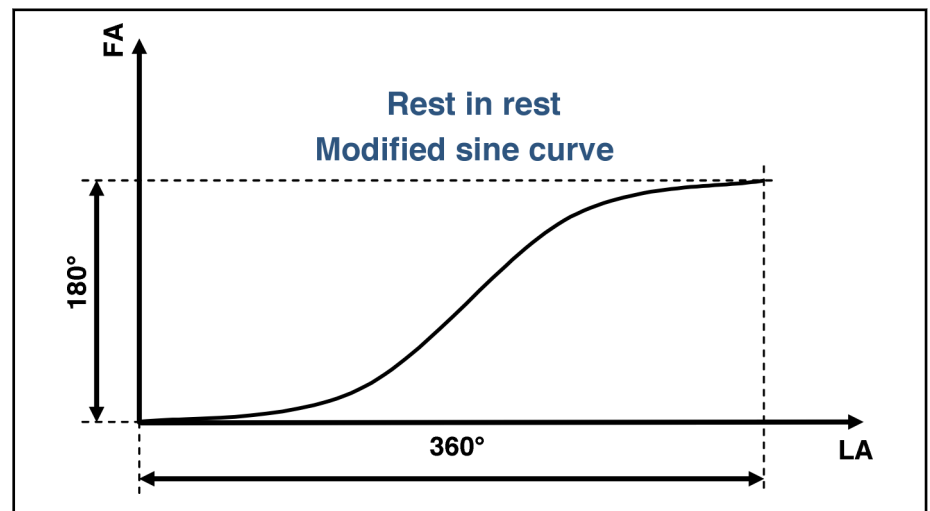


Fig. 4-1: Coupling profile between master axis (VA1) and slave axis (RA1)

The slave axis be assigned up to four MotionProfiles, each with a maximum of 16 (64) steps.¹⁾

The FlexProfile 0 in RA1 is used for "First Steps - FlexProfile".

A FlexProfile can be created in two different ways:

- IndraWorks, "FlexProfile" dialog of the slave axis, see [Creating the FlexProfile via IndraWorks Dialogs, page 129](#)
- or
- IndraLogic, PLC program code with a MB_ChangeFlexProfileSet function block for data transmission from the PLC program to the slave axis, see [Creating the FlexProfile via PLC, page 132](#)
 - For programming in ST, see [FlexProfile, data transfer in ST, page 132](#)
 - For programming in LD, see [FlexProfile, data transfer in in FBD, page 134](#)

The actual processing is started via the ML_FlexProfile function block (ML_FlexProfile assumes the task (connection slave axis to master axis, that was assumed by MC_GearIn in the "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, page 136](#)

¹⁾ The number of steps within a FlexProfile depends on the firmware version and the type of control.

- For programming in LD, see [ML_FlexProfile in FBD, page 137](#)

4.2 Technical Requirements

- Software:
 - PC with installed IndraWorks Engineering for IndraMotion MLC 13VRS
- Control:
 - IndraMotion MLC L25
 - IndraMotion MLC L45
 - IndraMotion MLC L65

A control, e.g. type "**IndraMotion MLC L65.1**" or "**IndraLogic XLC L65.1**", is **possible with sercos III interface**, and current firmware 13VRS has to be available. 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 the current firmware is loaded:
- 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 FlexProfile, refer to the documentation "Rexroth IndraMotion MLC13VRS Functional Description", chapter "FlexProfile"

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 VA1 as master axis and the real axis RA1, IndraDrive **with interpolation on the control**, as slave axis.

4.3 Creating the Motion Subproject

To ensure that the project runs on a control (IndraLogic XLC L25, L45, L65 (with or without sercos III bus) or IndraMotion MLC L25, L45, L65) proceed as follows:

1. Clear old data in the control

Clear the old memory content of the control.

Details are described under [Resetting the Control \(CMLx5\), on page 68](#).

The memory of the control is cleared and any possible Motion components present and the PLC components are ready for operation.

2. Create a new project

Create a new empty project (**File ► New ► Project**). The project name can be chosen at random, although "First Steps FlexProfile" is recommended.

A new empty project has been created.

3. Create the control

Drag a control, e.g. IndraLogic XLC L25, L45, L65 (with sercos III bus) or IndraMotion MLC L25, L45, L65 from the device library and drop it to the empty project. The control name can be chosen at random, although "DCC_Control" is recommended.

Complete device type and IP address.

Perform a connection test.

End adding as usual.

The desired control is now contained in the project.

4. Create the motion subproject

Scan the sercos bus (context menu sercos node **Scan Bus Configuration**).



Verify that the "Ipo drive" column is not ticked. Untick the column if Ipo drive is ticked!

The axis number "1" is assigned to the drive. Rename the drive in "RA1".

5. Declare a virtual axis. Assign the axis number "2" to the axis and change the name to "VA1".

In the Global Variable List "MlcVarGlobal", the axis numbers and names are provided as axis references of the "AXIS_REF" type.

The download (see [Download of the Motion project, page 75](#)) can be executed (the MOtion component is in state "Ready for operation (BB)").

6. Import the export file "Application_FirstSteps_Motion.iwx"

In the Project Explorer, expand the control node to **Logic ► Application**.

Select **Import...** in the context menu of the "Application" node.

Select the export file "Application_FirstSteps_Motion.iwx" in "Drive:\All Programs\Rexroth\IndraWorks\Projects". Select "Overwrite" for each element in the import dialog. The import is started using **Finish**.

The required files were copied. The project can now be run on the control.

4.4 Creating a FlexProfile

4.4.1 Creating a FlexProfile, General Information

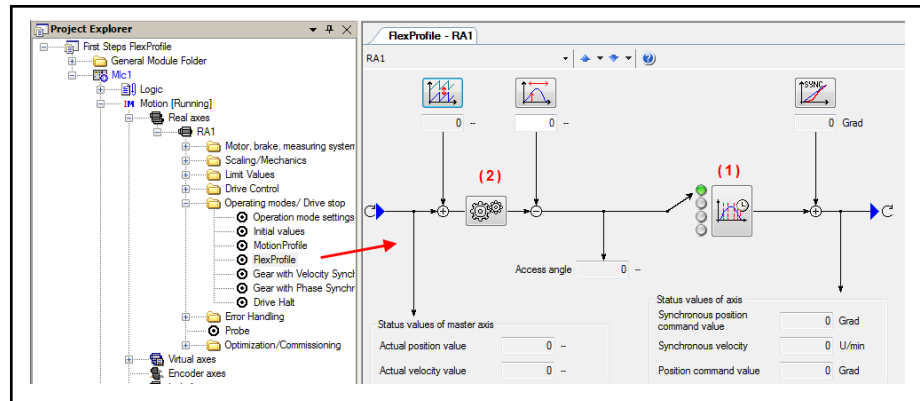


Creating or modifying a FlexProfile requires the control and PC to be coupled **online**.

4.4.2 Creating the FlexProfile via IndraWorks Dialogs

When clicking on the "FlexProfile" entry in the Project Explorer under "Real Axes" → "RA1", a dialog opens in which the general profile configuration can be selected.

First Steps - FlexProfile

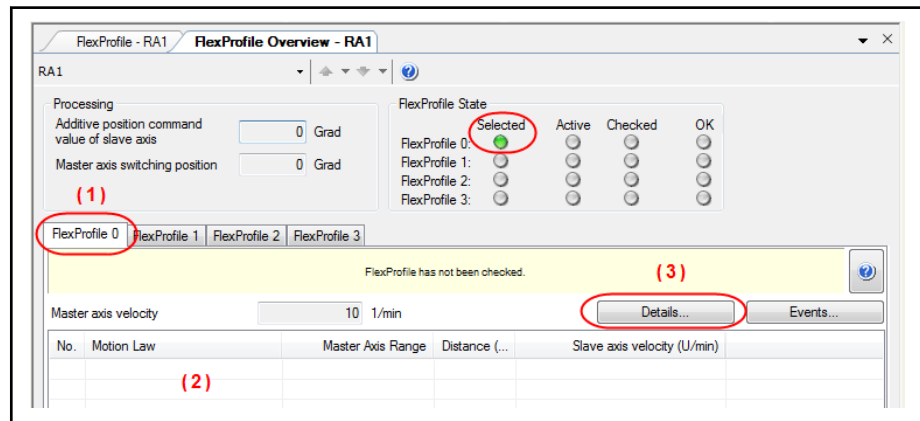


- (1) Transition to "FlexProfile 0"... "FlexProfile 3" dialogs
 (2) Transition to "Gear Settings" dialog

Fig.4-2: FlexProfile dialog with action chain between master and slave axis

Open the "FlexProfile Overview" by clicking on (1).

The following dialog shows the processing status for the four FlexProfiles assigned to the axis, registers "FlexProfile 0" to "FlexProfile 3".



- (1) "FlexProfile 0" to "FlexProfile 3", "FlexProfile 0" is selected
 (2) Overview on the motion steps 0...15 (63) of the selected FlexProfile (write-protected)
 (3) Transition to the details of the selected FlexProfile

Fig.4-3: Dialog: "FlexProfile Overview"; FlexProfile 0 is selected

A new MotionProfile can be created via "Details" (1).

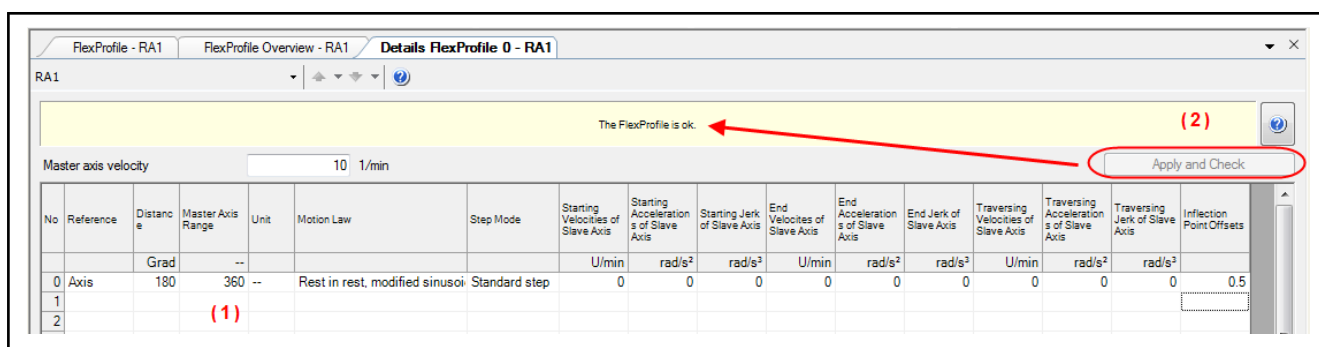
The FlexProfile Editor opens automatically.

An empty configuration table is shown in which each line defines one profile step [0...15 (63)].

The master axis sources, motion laws and motion step types required can be selected from the drop down menu.

Enter the data as shown in the figure below and configure the motion sequence.

First Steps - FlexProfile



- (1) Configuration of the individual motion steps [0...15 (63)] of the selected FlexProfile
- (2) Applying the data of the MotionProfile and check whether the profile is correct

Fig. 4-4: Dialog: FlexProfile configuration

For FlexProfile 0, the following data was entered into the dialog above:

Reference	Axis
Distance	180 degrees
Master axis section	360
Unit	-
Motion law	Rest-in-rest, modified sine curve
Step type	Standard step
... more...	0
Turning point displacement	0,5

After the FlexProfile configuration has been completed, the changes have to be applied to the control. This is done by clicking on **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.

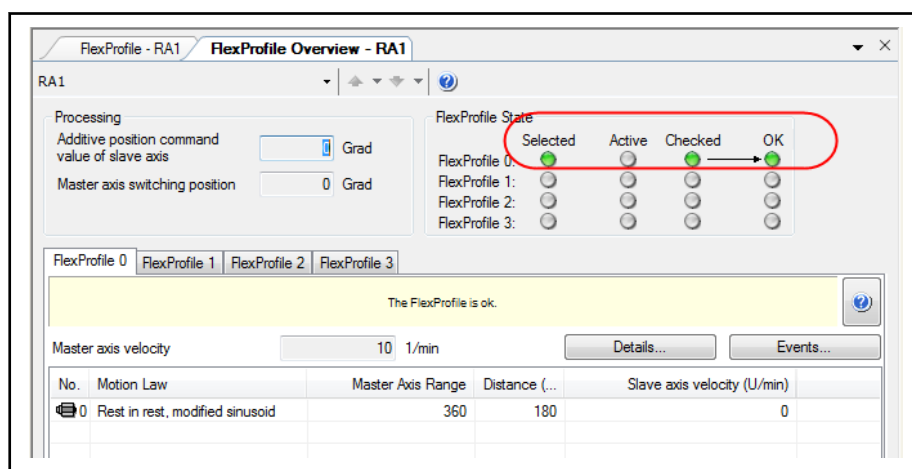


Fig. 4-5: Dialog: FlexProfile overview window, FlexProfile 0, checked → ok

First Steps - FlexProfile

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 can also be configured via PLC.



Please note that the execution time of the "MB_ChangeFlexProfileSet" function block can be somewhat longer depending on the size and the complexity of a profile to be written.

Therefore, the function block should only be used in PLC processes running freely.

Programming of the "First steps - FlexProfile" is therefore executed in the programs [PlcProg](#), on [page 132](#), or [PlcProg_FBD](#), [page 134](#). Both programs are alternatively controlled via the PlcTask.

Use 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 (63) 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.

The data set corresponds to [fig. 4-4 "Dialog: FlexProfile configuration" on page 131](#).

2. Transferring FlexProfile data from the PLC program to the control (axis RA1).

This task is executed by the instance "ChangeFlexProfileSetRA1" of type "MB_ChangeFlexProfileSet".

The instance corresponds to [fig. 4-5 "Dialog: FlexProfile overview window, FlexProfile 0, checked → ok" on page 131](#).

FlexProfile, data transfer in ST

The variable "profileAxis" is first declared when providing the data set.

Subsequently, its structure elements are assigned with the desired values.

The structure elements are listed in the library manager (ML_Base_xxx_yyy ► Data Types ► Flex Structs ► MB_PROFILE_STEP²⁾.

²⁾ ML_Base_xxx_yyy: xxx stands for XLC or MLC; yyy stands for L25, L45 or L65

First Steps - FlexProfile

Inputs/Outputs Documentation			
TYPE MB_PROFILE_STEP			
Name	Type	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	Segment type: standard (fixed) or Flex
StartVel	LREAL	0	Slave axis start velocities
StartAcc	LREAL	0	Slave axis start acceleration
StartJrk	LREAL	0	Slave axis start jerk
EndVel	LREAL	0	Slave axis final velocity
EndAcc	LREAL	0	Slave axis final acceleration
EndJrk	LREAL	0	Slave axis final jerk
TravelVel	LREAL	0	Slave axis traveling velocity, not taken into account by all motion laws
LimitAcc	LREAL	0	Slave axis traveling acceleration, not taken into account by all motion laws
LimitJrk	LREAL	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.4-6: Description of the structure MB_PROFILE_STEP in the library manager

Use the "." operator with the following Intellisense function for the input.

```

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;

```

Fig.4-7: "profileAxis" data set with value assignment in ST

From line 17 onwards, the instance "ChangeFlexProfileSetRA1" of the function block "MB_ChangeFlexProfileSet" is usually entered to transfer the data array to the control.

First Steps - FlexProfile

```

15  profileAxis[0].TurningPointDisplacement:= 0.5;
16
17  ChangeFlexProfileSetRA1(
18      Execute:= ex_ChangeFlexProfileSetRA1,
19      SetNumber:= SetNum,
20      NumberOfSteps:= 1,
21      MasterAxisVelocity:= 10,
22      ProfileStepAdr:= ADR(profileAxis),
23      Axis:= MlcVarGlobal.RA1,
24      Done=> do_ChangeFlexProfileSetRA1,
25      Active=> ac_ChangeFlexProfileSetRA1,
26      Error=> er_ChangeFlexProfileSetRA1,
27      ErrorID=> ID_ChangeFlexProfileSetRA1,
28      ErrorIdent=> Ident_ChangeFlexProfileSetRA1);

```

Fig. 4-8: Function block to transfer the data array to the control (SetNum = 0 for FlexProfile 0)

The data array is transferred by "Write Values" at the input "ex_ChangeFlexProfileSetRA1", 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.

FlexProfile, data transfer in in FBD

The variable "profileAxis" is first declared when providing the data set.

Subsequently, its structure elements are assigned with the desired values.

The structure elements are listed in the library manager (ML_Base_xxx_yyy ► Data Types ► Flex Structs ► MB_PROFILE_STEP)

TYPE MB_PROFILE_STEP			
Name	Type	Initial	Comment
Master	MB_MASTER_TYPE	MASTER_AX150	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	Segment type: standard (fixed) or Flex
StartVel	LREAL	0	Slave axis start velocities
StartAcc	LREAL	0	Slave axis start acceleration
StartJrk	LREAL	0	Slave axis start jerk
EndVel	LREAL	0	Slave axis final velocity
EndAcc	LREAL	0	Slave axis final acceleration
EndJrk	LREAL	0	Slave axis final jerk
TravelVel	LREAL	0	Slave axis traveling velocity, not taken into account by all motion laws
LimitAcc	LREAL	0	Slave axis traveling acceleration, not taken into account by all motion laws
LimitJrk	LREAL	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. 4-9: Description of the structure MB_PROFILE_STEP in the library manager

Use the "." operator with the following Intellisense function to input the assignments.

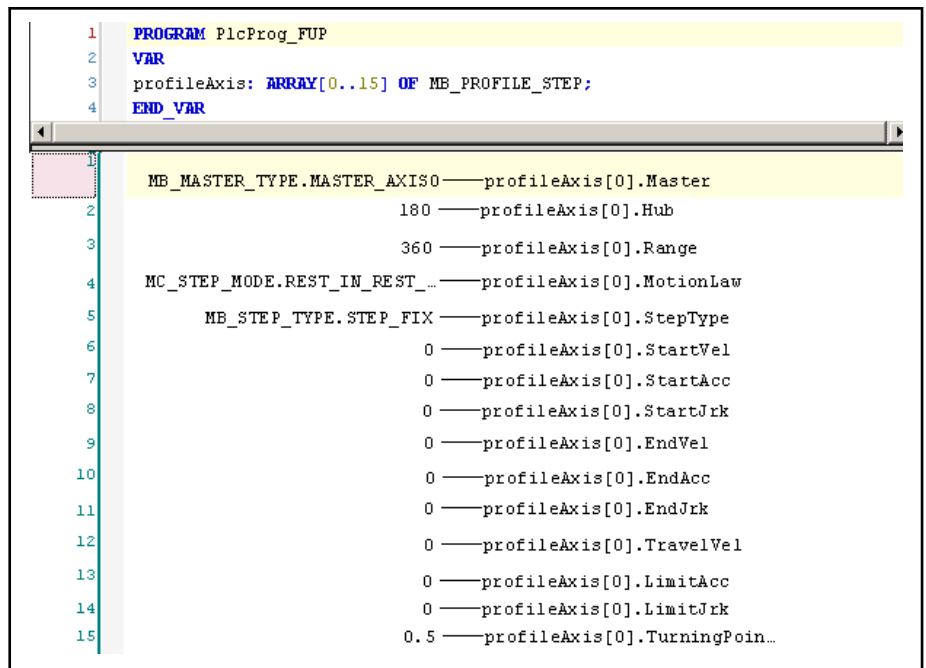


Fig.4-10: "profileAxis" data set with value assignment in FBD.

The instance "ChangeFlexProfileSetRA1" of the function block "MB_ChangeFlexProfileSet" is usually input as network number 16 to transfer the data array to the control.

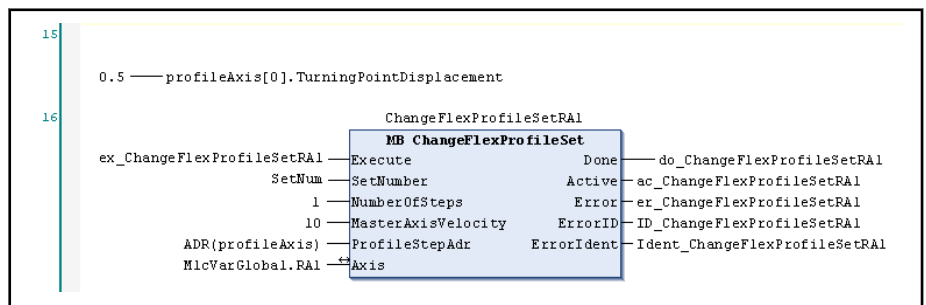


Fig.4-11: Function block to transfer the data array to the control (SetNum = 0 for FlexProfile 0)

The data array is transferred by "Write Values" at the input "ex_ChangeFlexProfileSetRA1", 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.

4.5 Completing the PLC Subproject, Executing a FlexProfile Motion Step

As soon as the MotionProfile is written on the axis, it can be started by executing the "ML_FlexProfile" function block. This function block establishes the coupling between the master axis and the slave axis by means of the MotionProfile previously specified and selected and synchronizes the slave axis.

Starting from this point in time, a slave axis motion follows the master axis motion. In the simplest case, the master axis rotates at constant velocity, for

First Steps - FlexProfile

example by calling the PLCopen function block "MC_MoveVelocity". The slave axis travels the desired MotionProfile.

Import the export file "Application_FirstSteps_IndraLogic_Motion.iwx" from the directory "Drive:\All Programs\Rexroth\IndraWorks\Projects".

Open the control node **Logic ► Application** for the import in the Project Navigator and select **Import...** in the context menu of the "Application" node.

This file contains in Structured Text (MotionProg) and in FBD (MotionProg_FBD) the programs already tested in "First Steps Motion".

The following modifications are required:

1. Change **VcomVA1** from 200 rpm to 10 rpm.
2. **MoveVelVA1:**
 - "Acceleration" → 100 [rad/s²]
 - "Deceleration" → 100 [rad/s²]
3. Delete the "GearInRA1" instance of MC_GearIn. This instance is replaced by the "FlexProfileRA1" instance of 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 "FlexProfileRA1" instance:

Declaration of "MotionProg":

1. Insert an empty line in front of END_VAR.
2. FlexProfileRA1: open the input assistant with <F2> and under **user-defined types**
 - ML_PLCopen ► POU ► Motion ► ML_FlexProfile**
 - deselect the "Add with arguments".
3. Complete line:


```
FlexProfileRA1: ML_FlexProfile;
```

Add empty line instead of "GearInRA1"

Open the input assistant with <F2> and select the option "Add with arguments" under **Instance calls ► FlexProfileRA1**

Assign inputs and declare input variables if necessary:

```
"Execute" → "ex_FlexProfileRA1" 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
```

First Steps - FlexProfile

"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" → "VA1" of type AXIS_REF

(already declared variable from the area of "Variables", Application, MlcVarGlobal)

"Slave" → "RA1" of type AXIS_REF

(already declared variable from the area of "Variables", Application, MlcVarGlobal)

Outputs:

"InSync" → "is_FlexProfileRA1" of type BOOL

"Active" → "ac_FlexProfileRA1" of type BOOL

"CommandAborted" → "ca_FlexProfileRA1" of type BOOL

"Error" → "er_FlexProfileRA1" of type BOOL

"ErrorID" → "ID_FlexProfileRA1" of type ERROR_CODE

"ErrorIdent" → "dent_FlexProfileRA1" of type ERROR_STRUCT

```

30      ErrorIdent=> Ident_PowerRA1);
31  FlexProfileRA1(
32      Execute:= ex_FlexProfileRA1,
33      RatioNumerator:= 1,
34      RatioDenominator:= 1,
35      MasterFineadjust:= 0,
36      SetSelection:= 0,
37      UseSwitchingPosition:= FALSE,
38      SwitchingPosition:= 0,
39      ProfileEntry:= MB_PROFILE_START_POINT.SLAVE_CURRENT_MASTER_CURRENT,
40      MasterOffset:= 0,
41      SlaveOffset:= 0,
42      SyncType:= MB_SYNC_TYPE.SYNC_RAMPIN_CATCH_UP,
43      SyncVelocity:= 100,
44      SyncAcceleration:= 100,
45      ExecutionMode:= MB_EXECUTION_MODE.EXECUTE_CYCLIC,
46      Master:= MlcVarGlobal.VA1,
47      Slave:= MlcVarGlobal.RA1,
48      InSync=> is_FlexProfileRA1,
49      Active=> ac_FlexProfileRA1,
50      CommandAborted=> ca_FlexProfileRA1,
51      Error=> er_FlexProfileRA1,
52      ErrorID=> ID_FlexProfileRA1,
53      ErrorIdent=> Ident_FlexProfileRA1);
54  GearOutRA1(
55      Execute:= ex_GearOutRA1,

```

Fig.4-12: "ML_FlexProfile" is inserted and assigned (ST)

ML_FlexProfile in FBD

As described above, add a function block of type "ML_FlexProfile" and assign the function block as described in the following.

Add ML_FlexProfile and declare a "FlexProfileRA1" instance:

First Steps - FlexProfile

Assign inputs and declare input variables if necessary:

"Execute" → "ex_FlexProfileRA1" 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" → "VA1" of type AXIS_REF
 (already declared variable from the area of "Variables", Application, MlcVarGlobal)
 "Slave" → "RA1" of type AXIS_REF
 (already declared variable from the area of "Variables", Application, MlcVarGlobal)

Outputs:

"InSync" → "is_FlexProfileRA1" of type BOOL
 "Active" → "ac_FlexProfileRA1" of type BOOL
 "CommandAborted" → "ca_FlexProfileRA1" of type BOOL
 "Error" → "er_FlexProfileRA1" of type BOOL
 "ErrorID" → "ID_FlexProfileRA1" of type ERROR_CODE
 "ErrorIdent" → "dent_FlexProfileRA1" of type ERROR_STRUCT

First Steps - FlexProfile

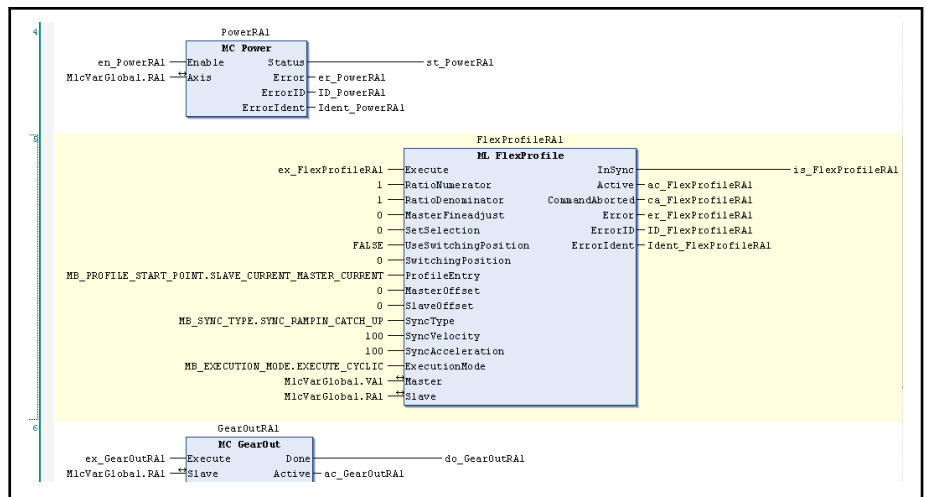


Fig.4-13: "ML_FlexProfile" is inserted and assigned (FBD)

4.6 Downloading and Testing the PLC Subproject

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

The compilation can be accessed via the IndraWorks menu item

- **Create ► Build**, compilation of the added part of the application
- **Create ► Rebuild**, compilation of the complete application

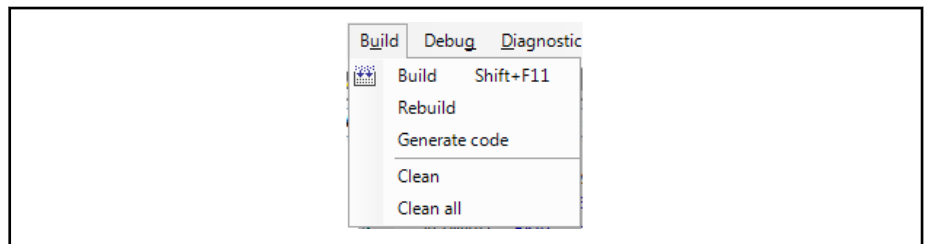


Fig.4-14: Build application

To apply the compiled data to the target control, select the IndraWorks menu item **Debug ► Login** (see the following figure).

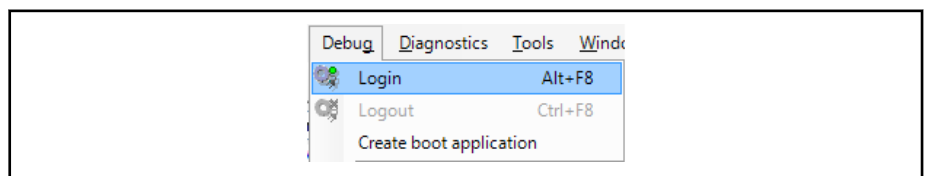


Fig.4-15: Transferring compiled data to the control

A prompt asking for agreement to download is displayed depending on whether an application is already loaded and whether this application is a predecessor of its 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 not valid.

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

4.7 Workflow "Virtual Axis → FlexProfile 0 → Real Axis "

The following procedure shows how to start and stop the axes and the FlexProfile:

1. Power on for the real axis
Enable "en_PowerRA1" (MC_Power) by "Write Values".
st_PowerRA1= TRUE (MC_Power).
2. Start the virtual axis
Enable "ex_MoveVelVA1" (MC_MoveVelocity) by "Write Values".
Start up ends at VactVA1 = 10 rpm.
3. Follow the real axis
Enable "ex_FlexProfileRA1" (ML_FlexProfile) by "Write Values".
The expected coupling behavior is:

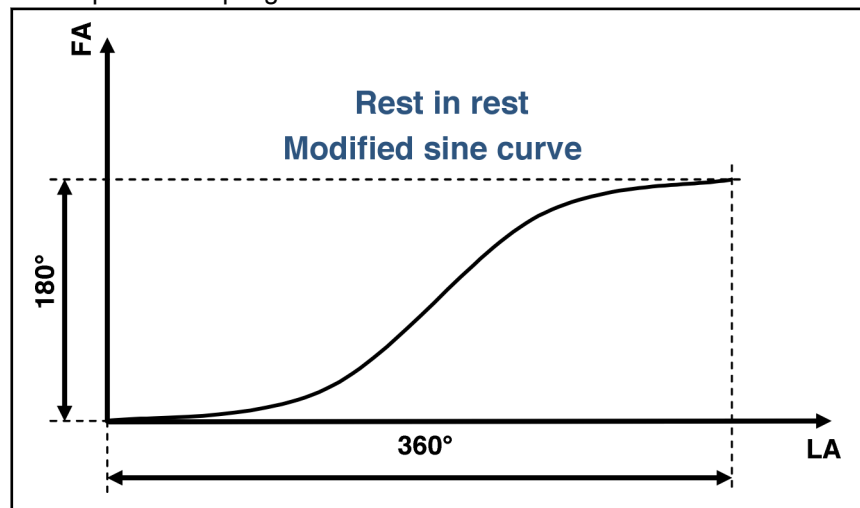


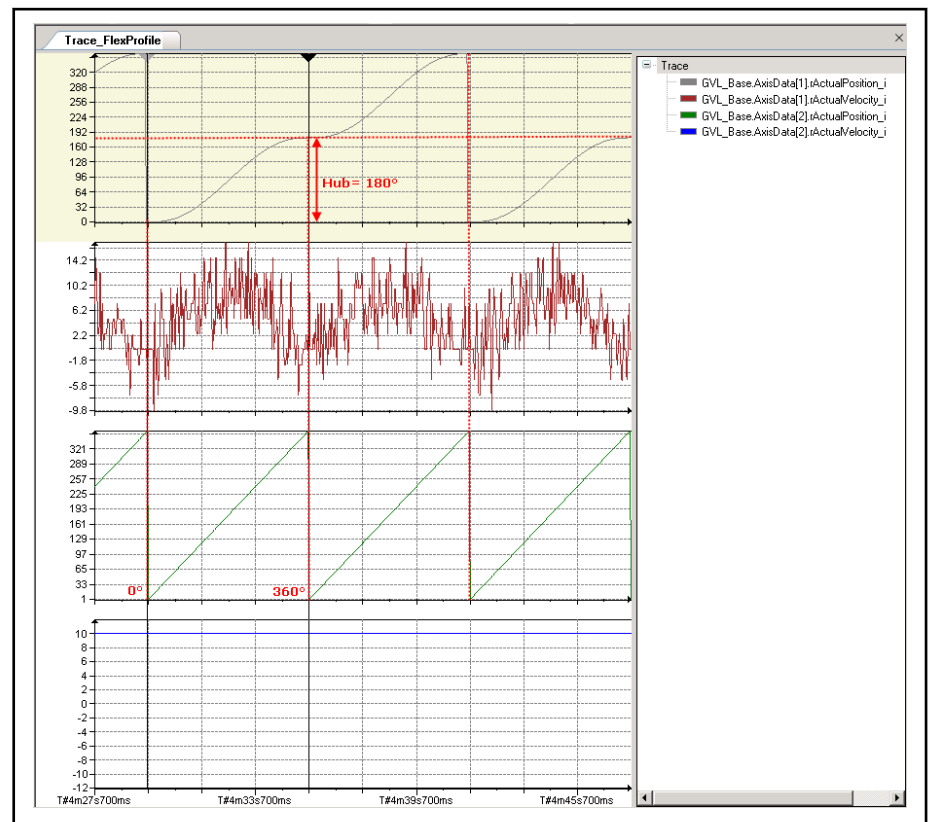
Fig.4-16: Coupling profile between master axis (VA1) and slave axis (RA1)

4. Disable "ex_FlexProfileRA1" (ML_FlexProfile) by "Write Values".
is_FlexProfileRA1= FALSE, ac_FlexProfileRA1= TRUE.
The real axis continues to follow the virtual axis.
5. Terminate synchronicity
Enable the "ex_GearOutRA1" (MC_GearOut) by "Write Values".
The dependency of the real axis on the virtual axis is suspended. However, the velocity at which both axes continue to rotate remains unchanged.
6. Change the command velocity "VcomVA1"
The changed command velocity only affects the virtual axis. "VactRA1" remains unchanged.
7. Disable "ex_GearOutRA1" by "Write Values" True/False.
do_GearOutRA1= FALSE
The real axis stops. The axis is actively retained at this position after reaching standstill.

8. Stop the virtual axis
Enable "ex_StopVA1" (MC_Stop)
The virtual axis is decelerated to $V_{act}VA1 = 0$ rpm.
9. Power off for the real axis
Disable "en_PowerRA1" (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. Result:



Grey Slave axis (RA1), position
Red Slave axis (RA1), velocity
Green Master axis (VA1), position
Blue Master axis (VA1), velocity

Fig. 4-17: Recording the motion of both axes

The trace recording shows the velocity characteristics of the master axis.

The master axis rotates at a constant velocity of 10 [rpm].

Above is the position of the master axis.

The uppermost signal shows the position of the slave axis.

It can be seen that the FlexProfile parameterized previously is traced correctly.

4.9 Restoring the Project

Based on the project created, an export file was generated enabling parts of the project to be restored irrespective of its control hardware (IndraLogic XLC

First Steps - FlexProfile

L25, L45, L65 (with sercos III bus) or IndraMotion MLC L25, L45, L65) and the drive engineering.

The export file has been created for the "Application" node and contains the following components:

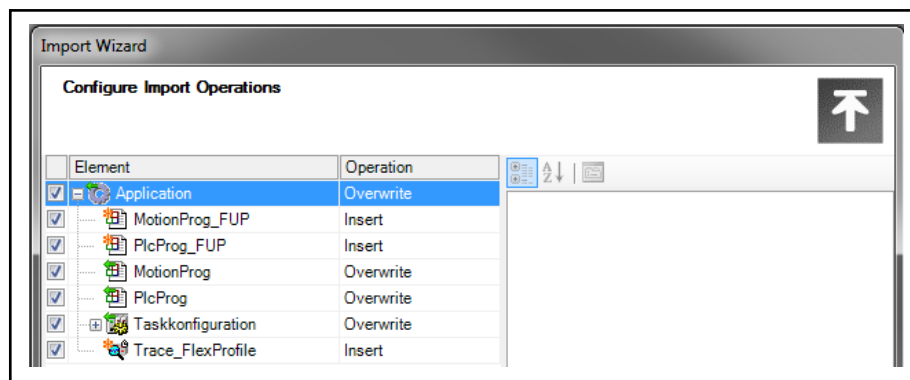


Fig.4-18: Components of the export file

The given components are overwritten during import.

The export file "Application_FirstSteps_FlexProfile.iwx" is located in the directory "<Drive>:\All Programs\Rexroth\IndraWorks\Projects".

5 First Steps - Robot Control V1

5.1 Project Goal "First Steps - Robot Control V1"

A simple example shows the basic procedure when creating a **RobotControl** project (called RoCo project in the following) using an IndraMotion MLC. The keywords of the minimum configuration required in IndraWorks and the PLC programming are addressed. It covers the procedure from the project creation up to the execution of the RoCo source code.

In the example project "RoCoTestV13", it is assumed that only virtual axes are available.

The procedure is also described briefly in case of real axes.

Therefore, the virtual axes in the project are substituted by real axes.

It is to be considered that

1. real axes have the same axis designation Axis1, Axis2 and Axis3
2. the axis numbers 1, 2 and 3 have been assigned in the same order

Thus, it is guaranteed that the PLC program runs without any adaptations.



The user can either restore the example project from the IndraWorks installation DVD from the directory "\\AddOns\\Mlc\\Example-Project\\RobotControl" which contains a .zip file with the following content:

- the parameter export file (*.par),
- the RCL program (*.qll)¹⁾
- PLC program
- the visualization data

- or -

he can create the project step by step executing the following steps. 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",
- Current firmware is loaded
- IP address is known,
- Programming device (PC / laptop)
- Control is connected to the programming device via Ethernet

Optional:

- 3 real axes (e.g. IndraDrive, EcoDriveCS,...)
- Current firmware (starting from MPx07) is loaded
- SRAM Modul (CFL01.1 Y1), only required for controls without Onboard SRAM

Required software IndraWorks installation 13VRS using MLC as target system.

¹⁾ RCL means **RobotControlLanguage**

5.3 Procedure when Creating an RoCo Project

1. Start IndraWorks and create a new project

After starting IndraWorks, create the new project via **File ► New ► Project...**

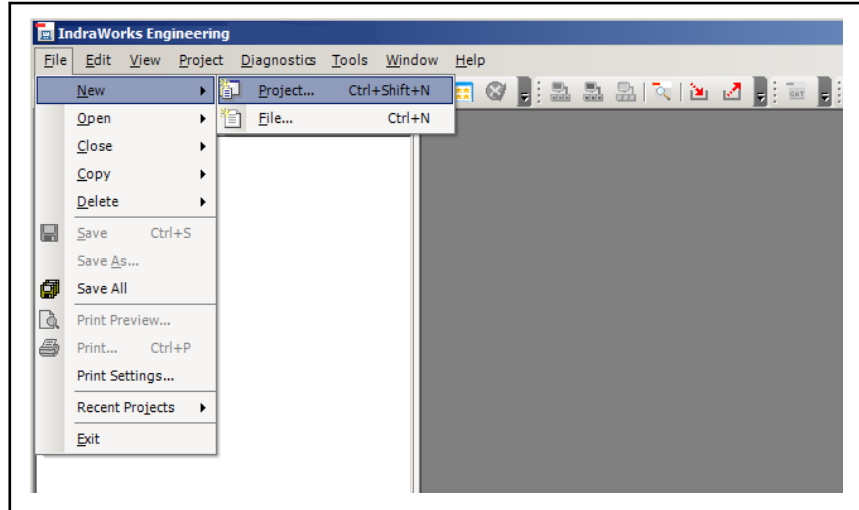


Fig.5-1: Creating a new IndraWorks project

2. Create and configure the IndraMotion MLC control

Therefore, an IndraMotion MLC Lx5 from the library is inserted into the project and its configuration data such as the device name, IP address, hardware configuration etc. are specified.

In the dialog "Interfaces", the function module SRAM module (CFL01.1-Y1) - memory extension for ndraMotion MLC can be created. Optionally, the function module can be added using Drag&Drop from the device library under **Function module**.



The "SRAM Modul (CFL01.1 Y1)" function module is only required for controls without Onboard SRAM.

First Steps - Robot Control V1

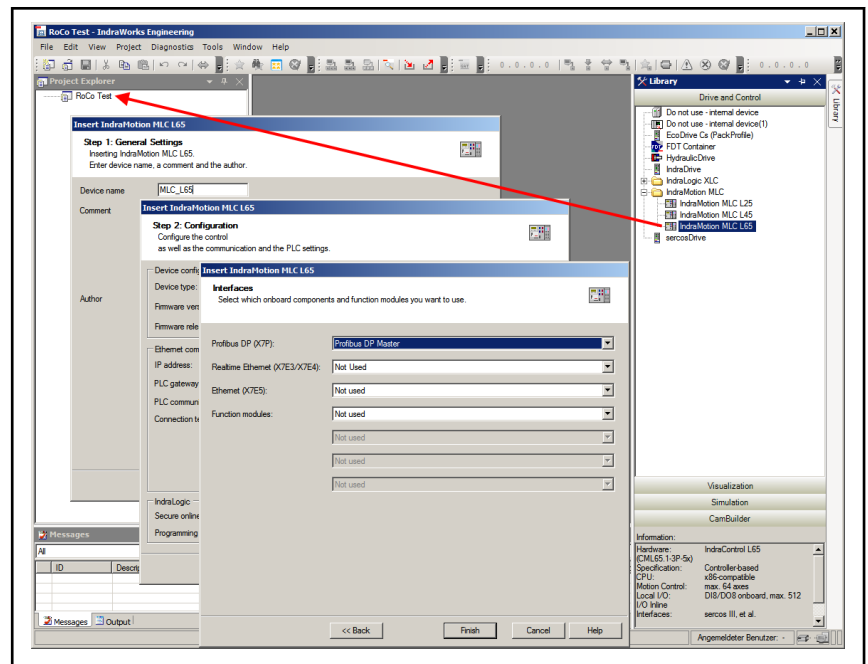


Fig.5-2: Creating and configuring an IndraMotion MLC

3. Creating and configuring **three** axes.

Two possible scenarios are described:

Scenario 1 with virtual axes:

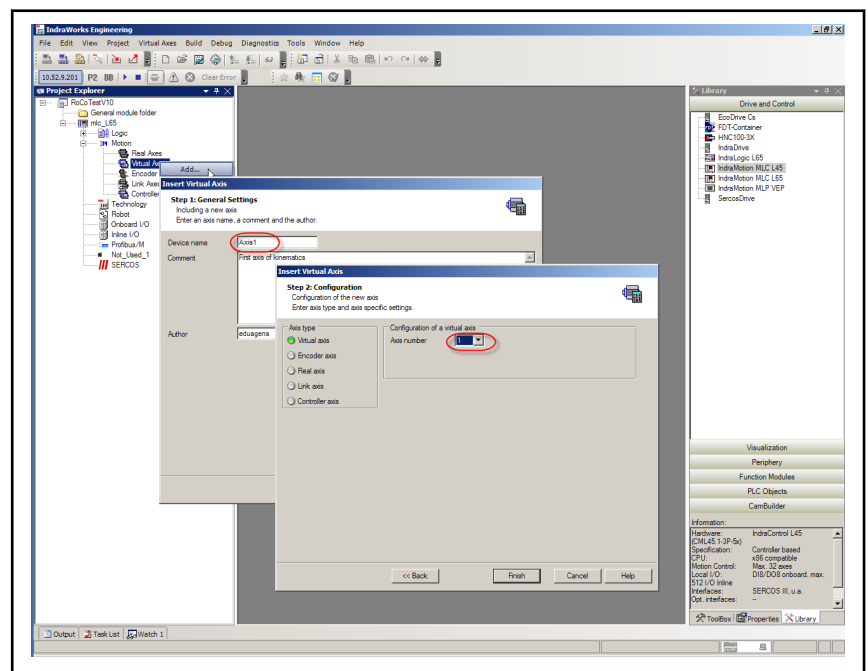


Fig.5-3: Project with virtual axes

Scenario 2 with real axes:

Creating of real axes have been described in detail in "First Steps Motion":

- Adding servos III Drives (Real Axes) to the Project on page 73

First Steps - Robot Control V1

- [Adding a Virtual Axis to the Project on page 74](#)
- [Downloading the Motion Subproject on page 75](#)



Parameter sets that can be imported are available for both the scenarios described. The parameter sets are saved in the project directory "RoCoTestV13\RoCoTestV13virtuell.par" (for virtual axes) or in the project directory "RoCoTestV13\RoCoTestV13real.par" (for real axes).

4. Adding and configuring a kinematics.

A new kinematics is added at the kinematic node of the IndraMotion MLC project.

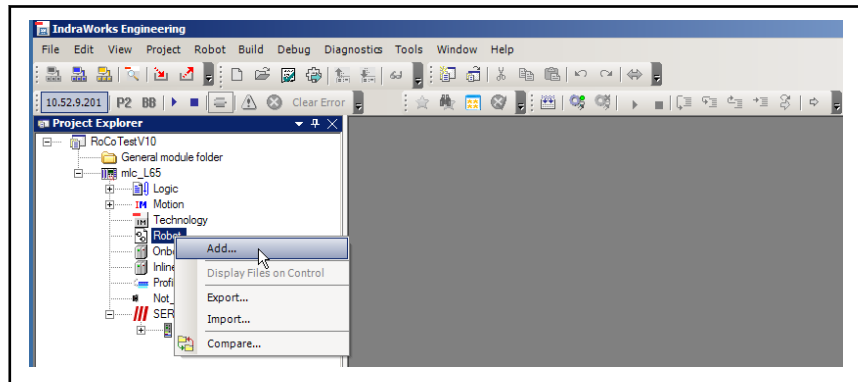


Fig.5-4: Adding new kinematics

Configuration according to the following dialogs:

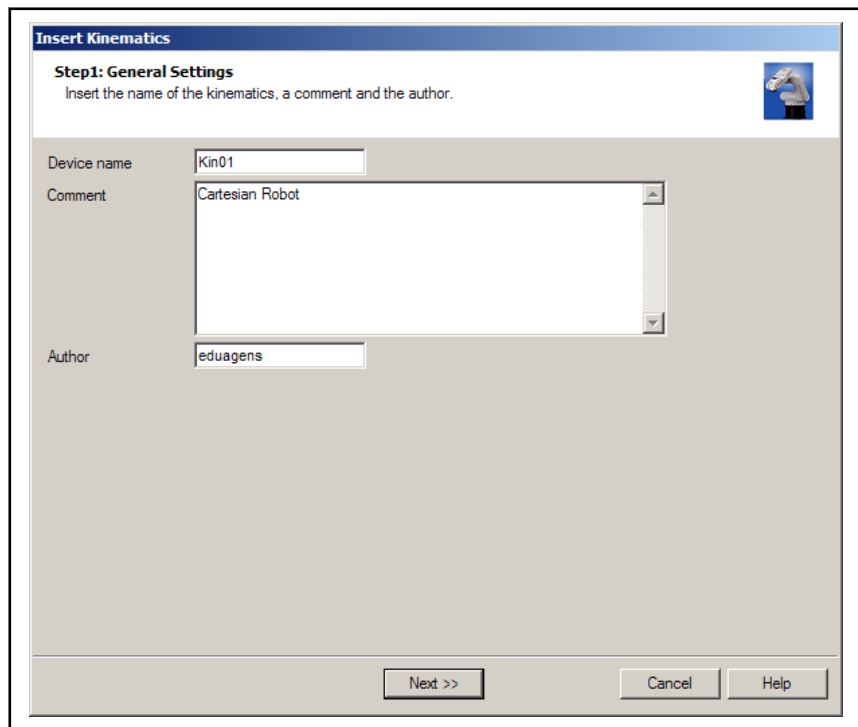


Fig.5-5: Specifying the kinematic names



A global MLC variable with the same name of the type AX-IS_GROUP_REF is created later in the PLC.

All kinematic function blocks need this variable as input variable.

First Steps - Robot Control V1

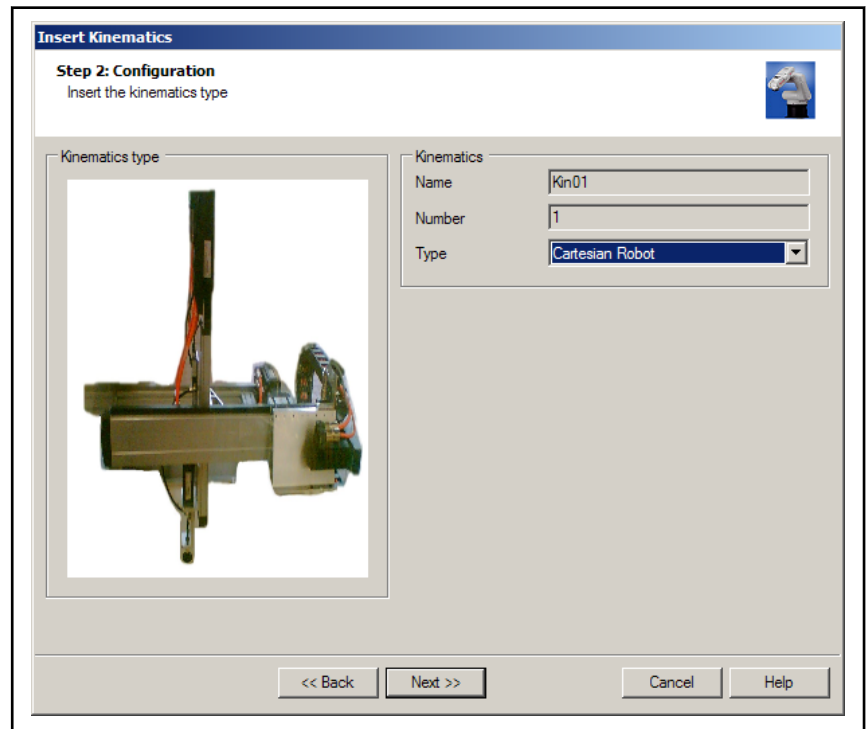


Fig.5-6: Specifying the kinematic type

In the dialog described above, all kinematics that can be interpreted by the control are listed.

In this example, the kinematic type "Cartesian Robot" is required.

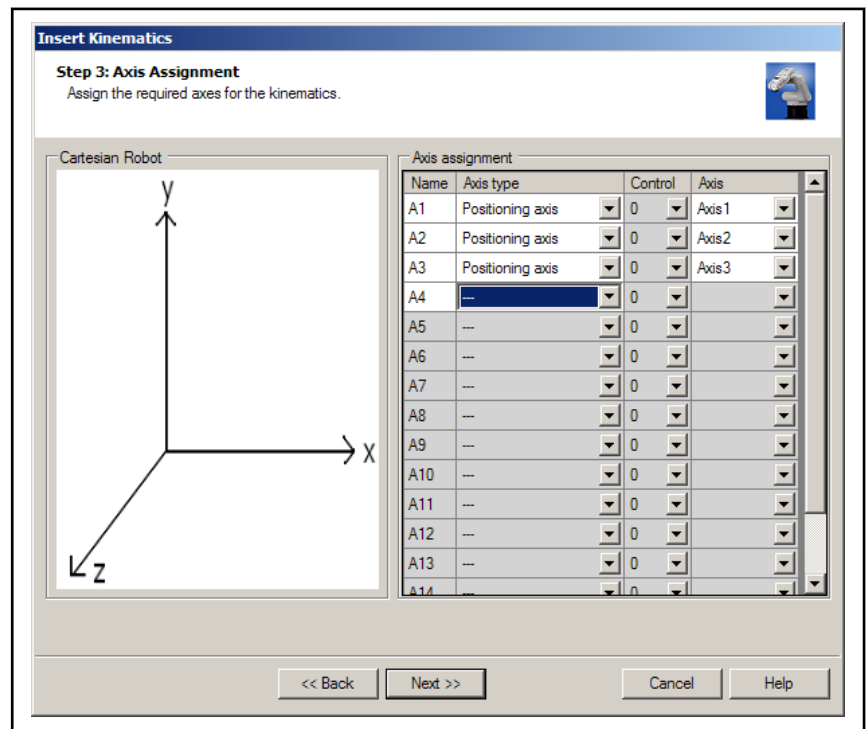


Fig.5-7: Assigning kinematic axes

First Steps - Robot Control V1

All axes belonging to the kinematics (real or virtual) have to be defined and assigned here.

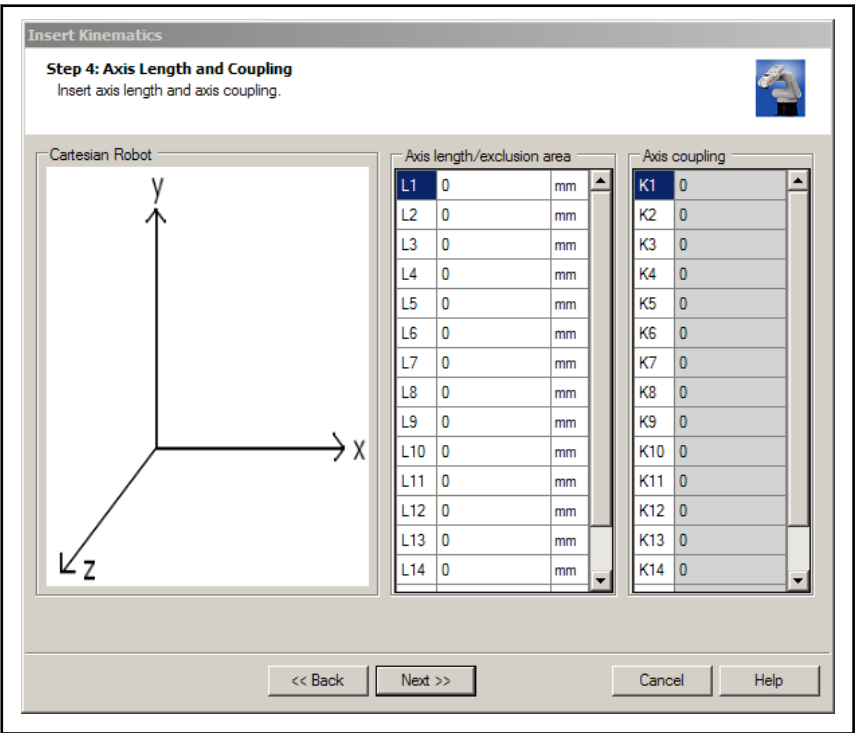


Fig.5-8: Axis length and axis coupling

The typical axis couplings and axis lengths for the robots are entered here.

These specifications are not required for the discussed example.

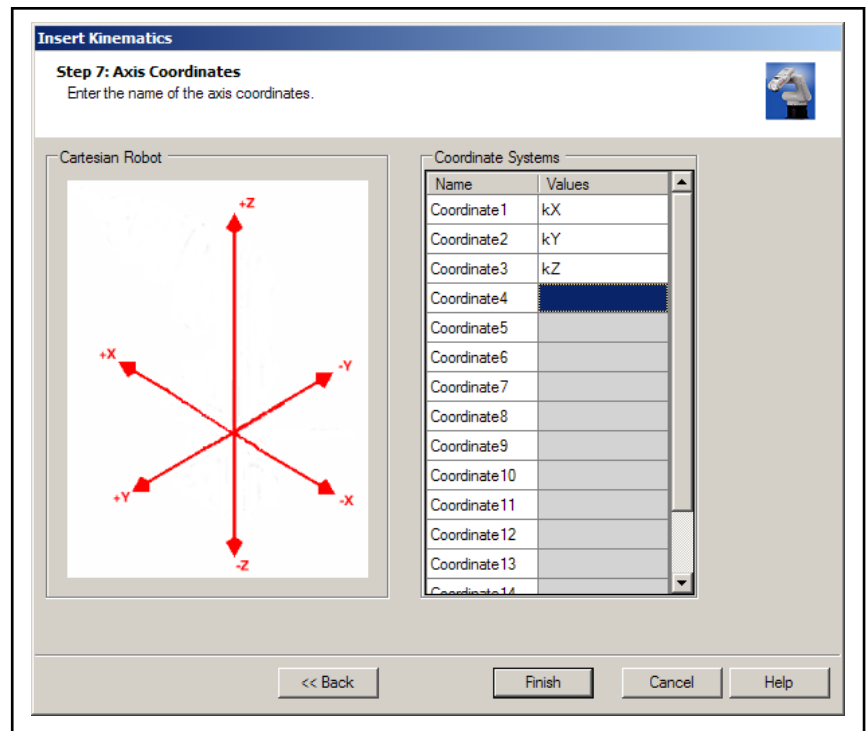


Fig.5-9: Specifying coordinate names

The coordinate names are defined in this step.



The coordinate designations defined are used in the RoCo programming carried out later.

5. Going online.

When going online, the control is switched to P2 and the settings carried out are applied to the control.

6. Import parameters.

For the example project, the C-, A- and K-parameters can be imported.

Selection:

Project directory "RoCoTestV13\RoCoTestV13virtuell.par" for virtual axes or project directory "RoCoTestV13\RoCoTestV13real.par" for real axes.

After a successful parameter import, it should be possible to switch the control to BB (P4).

First Steps - Robot Control V1

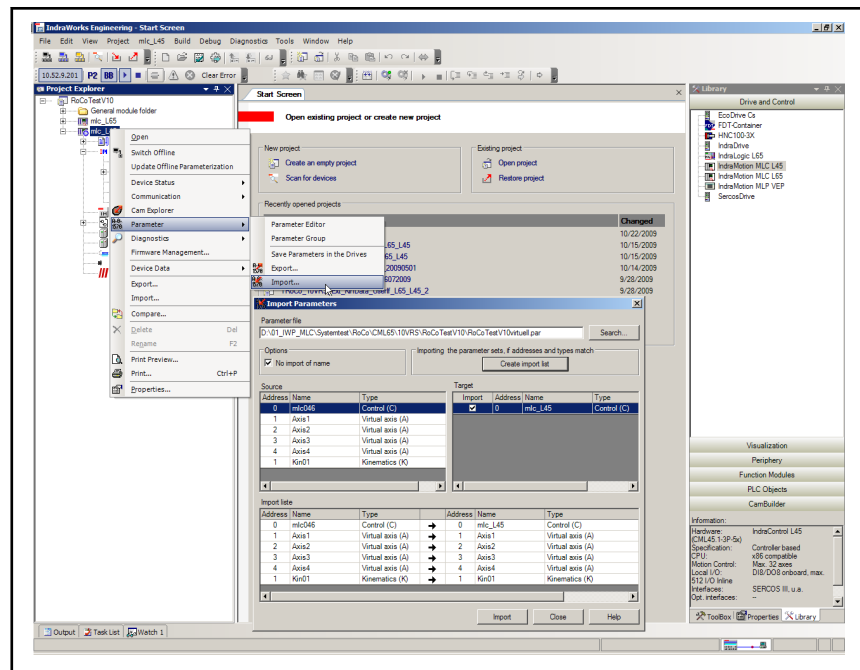


Fig.5-10: Import parameters

7. Create a Motion program.

A new RoCo Motion program "Prog1.qll" is created for the kinematics.

Optionally, the "Prog1.qll" program can be imported from the "RoCo-TestV13\mlc_Lx_\Emb_Prog1.qll" project.

Lx = Hardware variant (L25, L45, L65)

The Motion program contains the following program components:

- Prog1.QLL, (⇒ Source code in text-only format)
- Prog1.PKT, (⇒ Point table)
- Prog1.IRD, (⇒ Source code in machine format, compilation)
- Prog1.SYM, (⇒ Symbol file, required for debugging)

First Steps - Robot Control V1

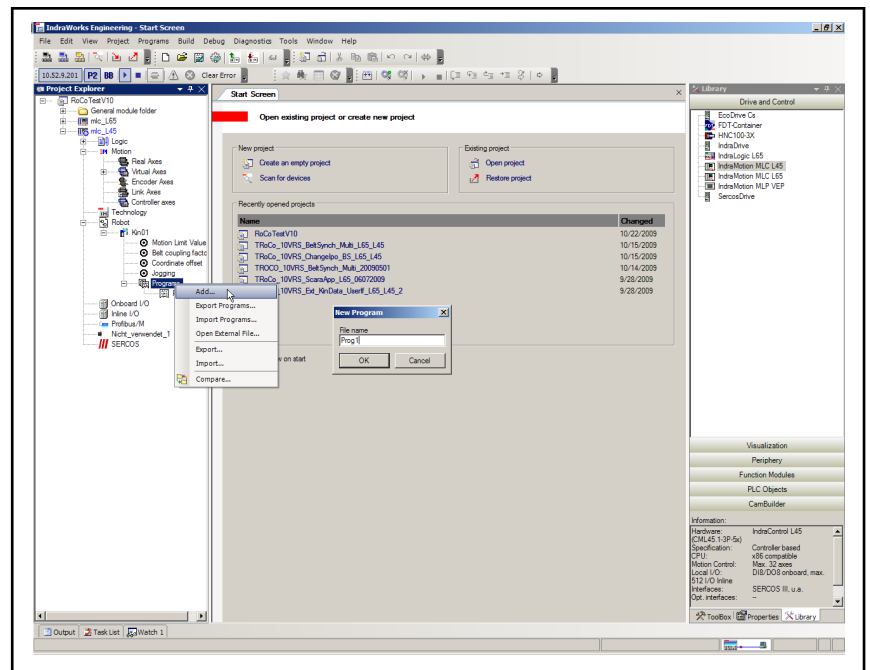


Fig.5-11: Creating a new RoCo program

After editing the RoCo motion program (*.QLL), the program can be compiled.

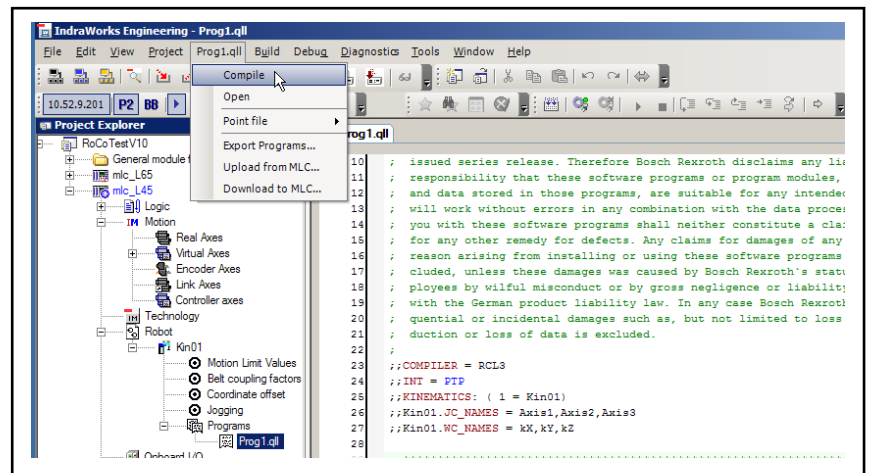


Fig.5-12: Editing the RoCo Motion program

After the compilation, a point table (*.PKT) is created automatically and can now be edited.

First Steps - Robot Control V1

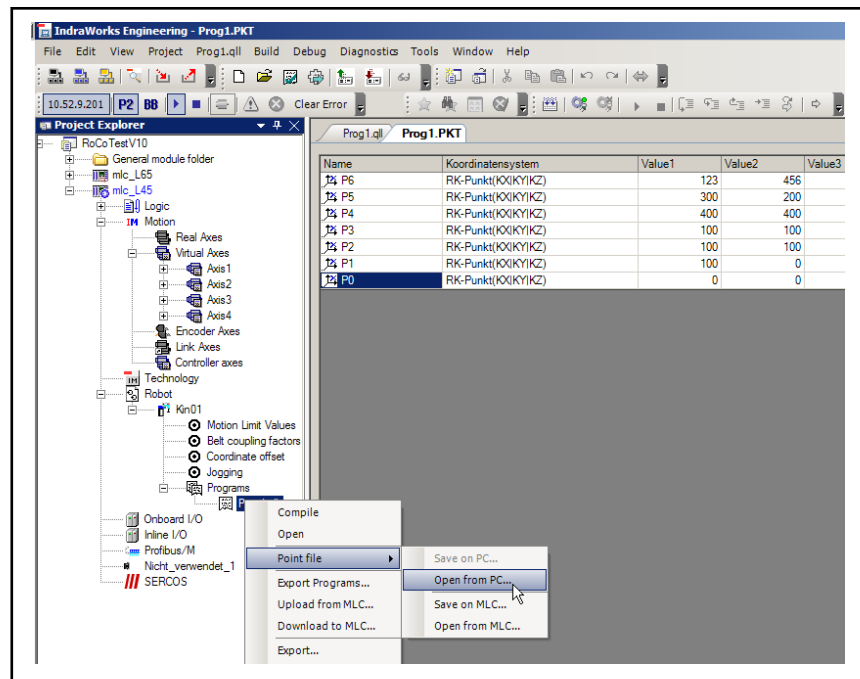


Fig.5-13: Editing the coordinate point table

The Motion program, point file etc. are loaded in the control when operating online.



If no Onboard-SRAM function module exists on the control or no SRAM function module is available, this step is to be repeated after each control reboot since the data is located in the volatile memory of the control.

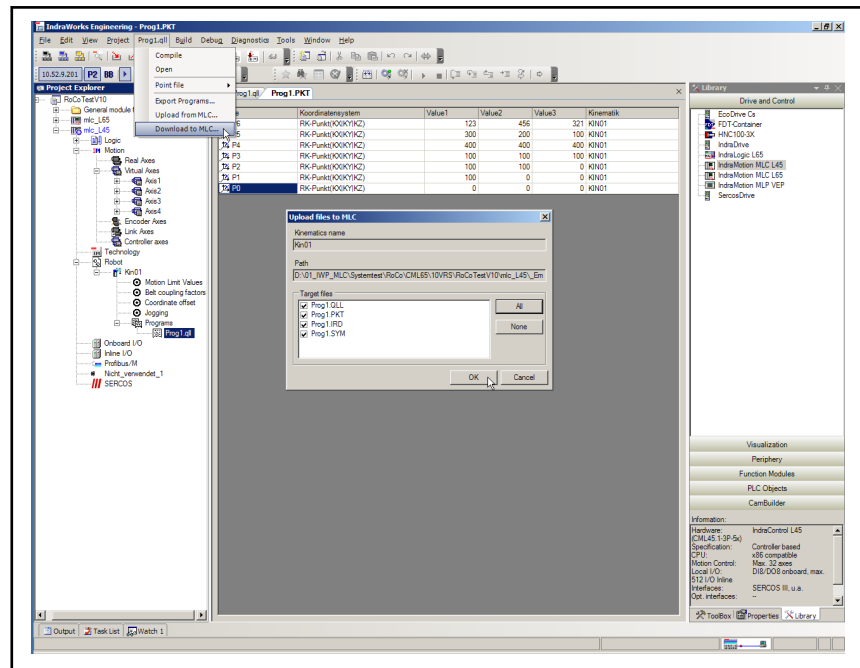


Fig.5-14: Loading the RoCo program in the control

8. RoCo status in the PLC program

There is a visualization in the example project to control the PLC part. Thus, all operating actions required, from enabling the power supply for real axes up to the RoCo program start and kinematic state can be read as well.

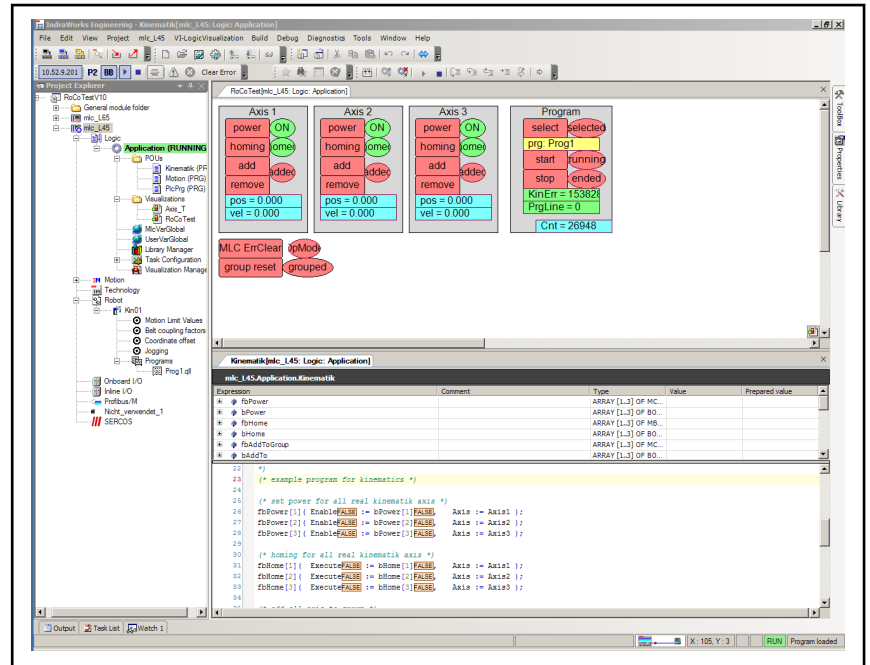


Fig.5-15: RoCo in PLC program

Brief description of the PLC functionality in the programmed sequence.

1. Connecting power.

This command is only to be executed for real axes.

This bit is always set for virtual axes.

Whether power had or had not been connected to an axis, can be read out using the bit "AxisData[n].Axis_Power".

2. Execute homing.

Homing is only required for real axes without absolute encoder evaluation.

The status if an axis is homed can be seen with the bit "AxisData[n].Axis_is_Homed".

3. Assign kinematic axes.

All axes belonging to the defined kinematics are grouped in this step. The axes change from the axis state machine to the kinematics state machine.

The grouped axes cannot travel using the PLCopen motion function blocks, but receive their motion commands directly from the RoCo motion program.

Control via PLC is only possible if all axes belonging to the kinematics are grouped.

First Steps - Robot Control V1

Whether all axes are required in Coordinated Motion, can be seen under the status "AxisData[x].CoordinatedMotion".

4. Removing axes from the kinematics.

Removing the axes from the kinematics is required to be able to traverse individual axes outside of the RoCo Motion program.

5. Selecting the RoCo motion program.

Select the program to execute a RoCo Motion program. This has to be performed after each program end and after each program stop.

6. Starting RoCo motion program.

This command starts the selected program.

7. Stop RoCo motion program

All axis motions are stopped and the selected program is reset.

In case of error

1. Reset the kinematic group (MC_GroupReset)

All kinematic errors are deleted.

2. Deleting MLC control errors and drive errors (C-0-1030)

Deletes all drive errors by the PLC.

5.4 Details on the Project "RoCoTestV13"

Parameter presettings (minimum changes)

C-parameter:

The C-parameter assignment results automatically when IndraMotion MLC is implemented in the Project Navigator and the virtual axes are created.

The following changes in the IndraWorks dialogs "Scaling/Units", "Settings Status Messages" and "Motion Limit Values" are useful for the axes:

- A-0-0053 , Scaling type for acceleration data: translatory
- A-0-0056 , Velocity data scaling type: translatory
- A-0-0059, Scaling type for position data: translatory, absolute
- Limit values and window values respectively

C-0-2484, Kinematic configuration list and all K-parameters result from the kinematic configuration.

Listing of the RoCo program

```
;;COMPILER = RCL3
;;INT = PTP
;;KINEMATICS: ( 1 = Kin01)
;;Kin01.JC_NAMES = Axis1,Axis2,Axis3
;;Kin01.WC_NAMES = kX,kY,kZ

, *****
, This programme was created by:
, Author: rudokohl
, Department: esc3
, Date: 06.12.2006
,
, Programme description:...
, *****

PROGRAM Prog1
; declarations
Def Point: P0, P1, P2, P3, P4, P5, P6

BEGIN
; program
```

First Steps - Robot Control V1

```
Kin01.V_PTP = 0.4      ; velocity for point to point movement
Kin01.A = 1000         ; acceleration
Kin01.V = 1200         ; velocity

;LOOP:
;JUMP LOOP

MOVE TO P0             ; move to P0
Kin01.V = 400          ; velocity
WAIT 3                ; wait for 3 seconds
REPEAT 4 TIMES
    MOVE TO P1          ; move to P1
    MOVE TO P2          ; move to P2
    MOVE TO P3          ; move to P3
    MOVE VIA P4, P5 TO P6 ; move via to P6
    WAIT 3              ; wait for 3 seconds
REPEAT_END

PROGRAM_END
```

Listing of the PLC program (Kinematics)

```
PROGRAM Kinematik
VAR
    fbPower      : ARRAY[1..3] OF MC_Power;
    bPower       : ARRAY[1..3] OF BOOL;
    fbHome       : ARRAY[1..3] OF MB_Home;
    bHome        : ARRAY[1..3] OF BOOL;
    fbAddToGroup : ARRAY[1..3] OF MC_AddAxisToGroup;
    bAddTo       : ARRAY[1..3] OF BOOL;
    fbRemFromGroup : ARRAY[1..3] OF MC_RemAxisFromGroup;
    bRemove      : ARRAY[1..3] OF BOOL;
    fbGroupReset : MC_GroupReset;
    bGroupReset  : BOOL;
    fbPrgSelect  : MB_IRDProgSelect;
    bPrgSelect   : BOOL;
    fbPrgStart   : MB_IRDProgStart;
    bPrgStart    : BOOL;
    fbPrgStop    : MB_IRDProgStop;
    bPrgStop     : BOOL;
    strProg      : STRING(8) := 'Prog1';
END_VAR

(* example program for kinematics *)

(* set power for all real kinematics 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 kinematics axes *)
fbHome[1]( Execute := bHome[1], Axis := Axis1 );
fbHome[2]( Execute := bHome[2], Axis := Axis2 );
fbHome[3]( Execute := bHome[3], Axis := Axis3 );

(* add all axes to group *)
fbAddToGroup[1]( Execute := bAddTo[1], Axis := Axis1, AxisGroup := Kin01 );
fbAddToGroup[2]( Execute := bAddTo[2], Axis := Axis2, AxisGroup := Kin01 );
fbAddToGroup[3]( Execute := bAddTo[3], Axis := Axis3, AxisGroup := Kin01 );

(* remove all axes from group *)
fbRemFromGroup[1]( Execute := bRemove[1], Axis := Axis1, AxisGroup := Kin01 );
fbRemFromGroup[2]( Execute := bRemove[2], Axis := Axis2, AxisGroup := Kin01 );
fbRemFromGroup[3]( Execute := bRemove[3], Axis := Axis3, AxisGroup := Kin01 );

(* select program *)
fbPrgSelect( Execute := bPrgSelect, IRDProgName := strProg, AxisGroup := Kin01 );

(* start program *)
fbPrgStart( Execute := bPrgStart, AxisGroup := Kin01 );

(* stop program *)
fbPrgStop( Execute := bPrgStop, AxisGroup := Kin01 );

(* reset group *)
fbGroupReset( Execute := bGroupReset, AxisGroup := Kin01 );

END_PROGRAM
```

First Steps - Robot Control V1

Listing of the PLC program (PlcPrg) to delete errors

```

PROGRAM PlcPrg
VAR
    fbReset      : MB_WriteParameter;
    bErrClear    : BOOL;
END_VAR

(* error clear MLC *)
fbReset( Execute := bErrClear,
  ParameterNumber := FP_C_0_1030,
  Value := 3,
  Axis := Axis1,
  Done=> , Active=> , Error=> , ErrorID=> , ErrorIdent=> );

END_PROGRAM

```

5.5 Robot Control - Application Examples

5.5.1 RoCo Basics for Pick & Place Applications

General Information

In general, the motion sequence for the Tool Center Point (hereafter referred to as TCP) for simple Pick & Place applications is as follows.

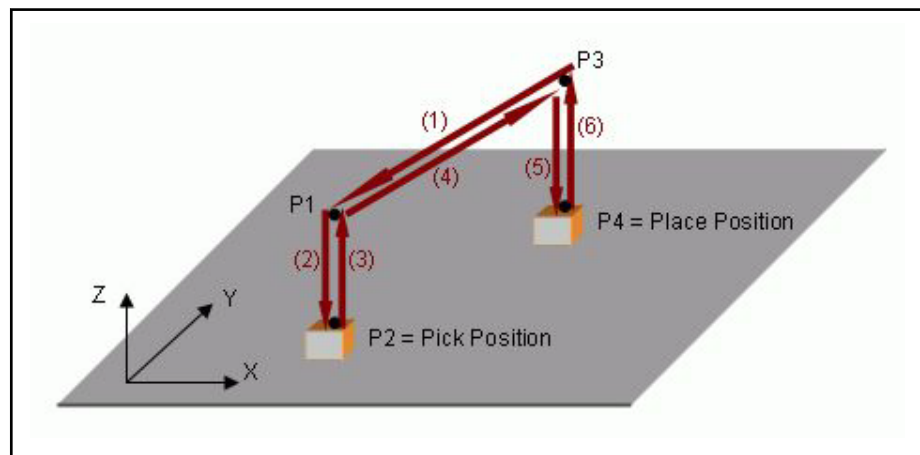


Fig.5-16: Simple Pick & Place application

- Description**
- horizontal motion (1) above the object
 - vertical motion (2) down to the object to grip the object
 - vertical motion (3) up with the object
 - horizontal motion (4) above the location for depositing
 - vertical motion (5) down to the location for depositing to deposit the object
 - vertical motion (6) up without the object
 - Waiting for a new object and then proceeding with (1)

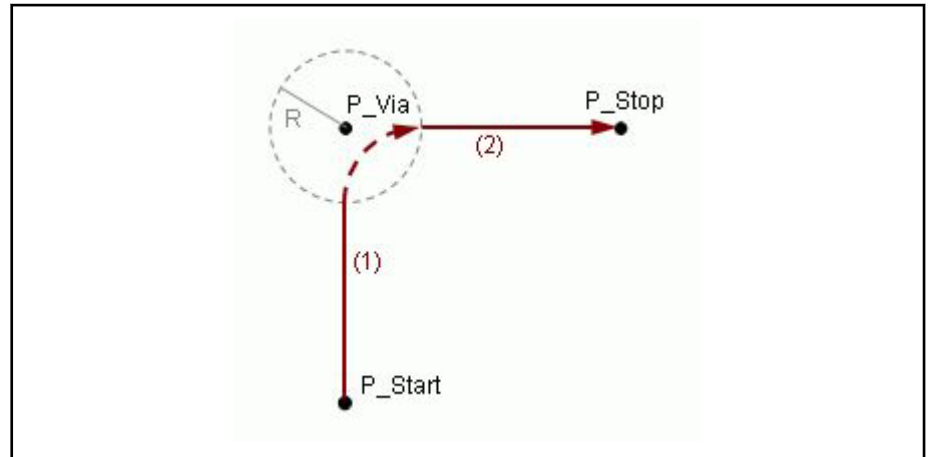
Blending

To carry out the previously traced motion sequence more harmonically and therefore more quickly, it is possible to blend the individual motion blocks between the object pick-up and placement and vice versa.



Blending with BLOCK_SLOPE means that the subsequent motion block starts early when entered into the programmed blending radius R . Within the blending radius, both the motion laws are active. That means that the resulting path curve within the blending sphere results from the superimposing of individual motions.

If the points of the blending motion block are not located on a straight line, the points are not traversed directly.



R Blending radius

Fig. 5-17: General blending

The horizontal motion (2) is started when entering the blending radius.

Both the motion blocks (1) and (2) are active during the blending radius.

The vertical motion (1) ends only when the blending radius is exited.

For the example shown above, blending results in the following motion sequence. Only points P2 and P4 are approached with the "exact stop".

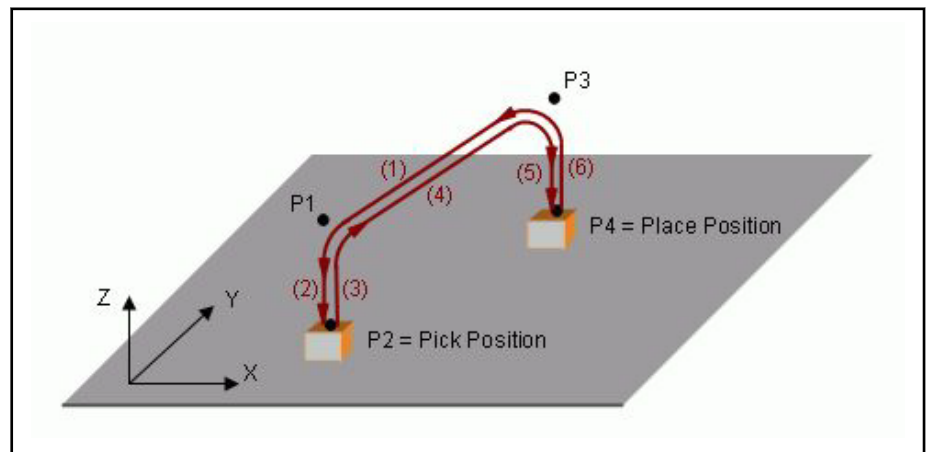


Fig. 5-18: Pick & Place application with blending motions

Description

- Horizontal motion (1) across the object, which changes immediately into a vertical motion (2) downwards to the object when the blending radius is entered.
- Stopping vertical motion (2) down to the object to grip it.
- Vertical motion (3) upwards with the object, which changes immediately into horizontal motion (4) when the blending radius is entered

First Steps - Robot Control V1

- Horizontal motion (4) across the depositing position, which changes immediately into vertical motion (5) down towards the place position when the blending radius is entered
- End of the vertical motion (5) down to the depositing position to deposit the object. Check for new object
- If there is a new object: Vertical motion (6) upwards without the object, which changes immediately into horizontal motion (1) across the object when the blending radius is entered
- If there is no new object: Vertical motion (6) towards P3 and waiting for a new object



To travel to several motion blocks in sequence with blending motion, some basic conditions must be fulfilled.

Blending is interrupted if the following actions are carried out during the blending motion:

- Accessing the synchronous inputs (POINT; BELT; REAL; INTEGER; BINARY)
- Querying the current position (POS; @POS)
- Accessing to the belt counter ("B1")
- Setting or resetting the belt counter (SYNC)
- Waiting times (WAIT)

Belt Synchronization

To remove objects from a moving belt or to deposit them onto a moving belt, belt synchronization must be used.

The following conditions have to be fulfilled to use the belt synchronization:

- Belt axes have to be defined in the kinematic configuration
- Belt coupling factors have to be defined in the kinematic configuration
- The belt counter (channels 501 - 504) has to be declared in the RCL program
- Belt type 4 has to be defined in the RCL program (SPC_FCT 21)
- The belt-synchronous working area has to be defined in the RCL program (SPC_FCT 53)



For synchronization on a moving belt, there must be sufficient distance available for the robot to accelerate to the velocity of the belt. The same applies to the desynchronization.

SYNCHRON

The statement for activating belt synchronization is "SYNCHRON" followed by the designation of the corresponding belt counter.

With the first motion block after the synchronous statement, the robot is synchronized with the belt. At the end of the motion block, the synchronization is complete.

In this motion block, the belt coordinates must contain the value of the belt counter which corresponds to the target position.



If the position of the first motion block is in front of the belt-synchronous workspace, the execution is delayed until the position enters the belt-synchronous workspace.



If the position of the first motion block is already behind the last possible start position of the belt-synchronous working area, the motion is not carried out and the program is interrupted with an error.

SYNCHRON_END

With the final motion block before the statement "SYNCHRON_END" the robot is desynchronized from the belt. In this motion block, the value of the belt coordinate must be the target position "0".

For the example shown above, this results in the following motion sequence:

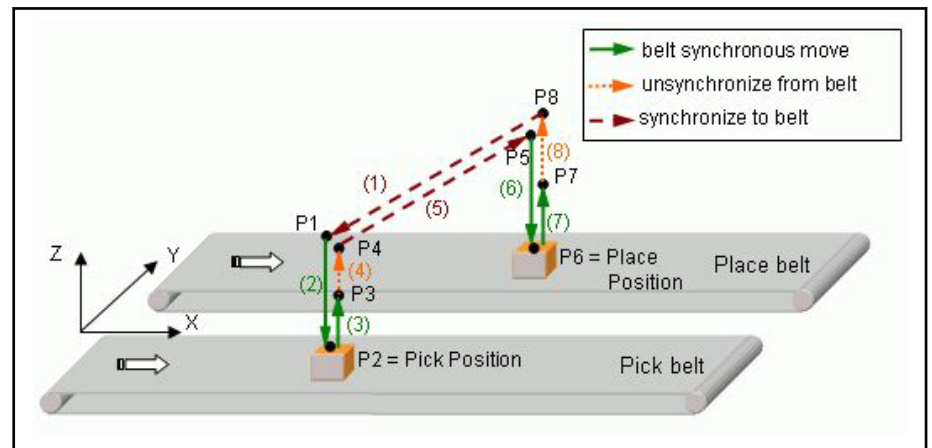


Fig. 5-19: Pick & Place application with belt synchronization

Legend

- P1** Synchronization point for the Pick Belt
- P2** Belt-synchronous position of the object on the Pick Belt
- P3** Belt-synchronous position above P2
- P4** Desynchronization point for the Pick Belt = current position before beginning the motion towards P3. However, z-coordinates are corrected upwards
- P5** Synchronization point for the Place Belt
- P6** Belt-synchronous position of the object on the Place Belt
- P7** Belt-synchronous position above P6
- P8** Desynchronization point for the Place Belt = current position before beginning the motion towards P7. However, z-coordinates are corrected upwards

- Description**
- Synchronization motion (1) on the Pick Belt across the synchronization point P1
 - Belt-synchronous motion (2) after P2
 - Gripping the object
 - Determining the desynchronization position P4 from the current position



Do not desynchronize as otherwise, the motion of the gripper relative to the belt is vertical and the following objects on the belt could be moved by the object at the gripper.

- Belt-synchronous motion (3) after P3
- Desynchronization motion (4) from the Pick Belt across the desynchronization point P4

First Steps - Robot Control V1

- Synchronization motion (5) on the Place Belt across the synchronization point P5
- Belt-synchronous motion (6) after P6
 - Depositing the object
 - Determining the desynchronization position P8 from the current position



Do not desynchronize as otherwise, the motion of the gripper relative to the belt is vertical and if a cardboard box is present it can be moved.

- Belt-synchronous motion (7) after P7
- Desynchronization motion (8) from the Place Belt across the desynchronization point P8
- Waiting for a new object and then proceeding with (1)

Belt Synchronization with Blending Motions

If the motion blocks should be blended, the following motion sequence results:

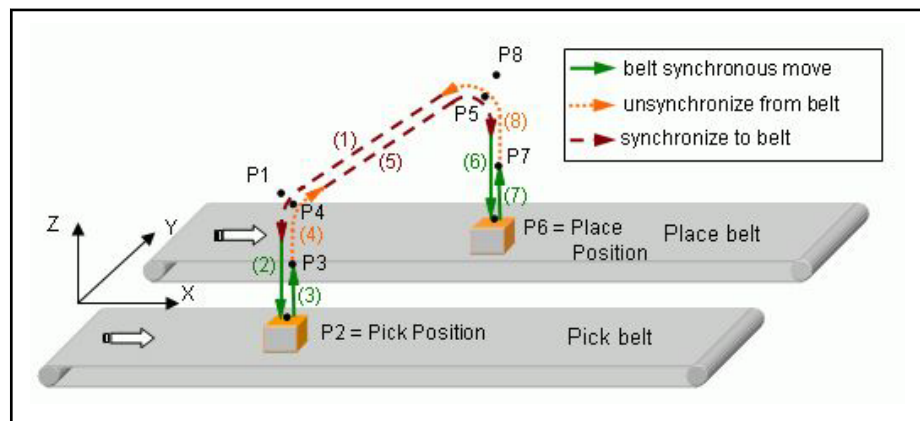


Fig. 5-20: Pick & Place application with belt synchronization and blending motions

Legend

- P1** Synchronization point for the Pick Belt
- P2** Belt-synchronous position of the object on the Pick Belt
- P3** Belt-synchronous position above P2
- P4** Desynchronization point for the Pick Belt = current position before beginning the motion towards P3. However, z-coordinates are corrected upwards
- P5** Synchronization point for the Place Belt
- P6** Belt-synchronous position of the object on the Place Belt
- P7** Belt-synchronous position above P6
- P8** Desynchronization point for the Place Belt = current position before beginning the motion towards P7. However, z-coordinates are corrected upwards

Points P1; P4; P5 and P8 are not traveled through directly due to blending.

- Description**
- Synchronization motion (1) on the Pick Belt across the synchronization point P1

First Steps - Robot Control V1

- When the blending radius is entered, the vertical motion (2) towards P2 starts. Until the blending radius is exited, both the motion blocks (1) and (2) are active
- Belt-synchronous, non-blending motion (2) after P2
 - Gripping the object
 - Determining the complete path from P2 (Pick) to P6 (Place)



Do not desynchronize as otherwise, the motion of the gripper relative to the belt is vertical and the following objects on the belt could be moved by the object at the gripper.

- Belt-synchronous blending motion (3) across P3
 - When the blending radius is entered, the vertical desynchronization motion (4) of the Pick Belt across the desynchronization point P4 starts. Until the blending radius is exited, both the motion blocks (3) and (4) are active
- Desynchronization motion (4) from the Pick Belt across the desynchronization point P4
 - When the blending radius is entered, the horizontal synchronization motion (5) of the Place Belt across the synchronization point P5 starts. Until the blending radius is exited, both the motion blocks (4) and (5) are active
- Synchronization motion (5) on the Place Belt across the synchronization point P5
 - When the blending radius is entered, the vertical belt-synchronous motion (6) towards P6 starts. Until the blending radius is exited, both the motion blocks (5) and (6) are active
- Belt-synchronous, non-blending motion (6) after point P6
 - Depositing the object
 - Check if a new object exists

If there is a new object: Determine the complete path from P6 (Place) to P2 (Pick)

If there is no new object: Determine the path of P6 (Place) up to the desynchronization position P8 based on the current position



Do not desynchronize as otherwise, the motion of the gripper relative to the belt is vertical and if a cardboard box is present it can be moved.

- Belt-synchronous blending motion (7) across P7
 - When the blending radius is entered, the vertical desynchronization motion (8) of the Place Belt across the desynchronization point P8 starts. Until the blending radius is exited, both the motion blocks (7) and (8) are active
- Desynchronization motion (8) from the Place Belt across the desynchronization point P8

If there is no new object: Wait for a new object and then determine the path from P8 to P2 (Pick)
- and continue with (1)

First Steps - Robot Control V1

Instructions for Blending during Belt Synchronization

To blend motions in ongoing motions, some basic conditions must be taken into consideration:

- Commands that prevent blending:
 - Access on synchronous inputs (Point; Real; Integer; Binary)
 - Access on the belt counter (Belt 501-504)
 - Query of the current position (POS; @POS)
 - Setting or "modulating" the belt counter (SYNC)
 - Waiting times (Wait)
 - Pause
- The motions between Pick (P2) and Place (P6) (and vice versa) can only be blended if the complete path was determined before the blending motion starts

This determination must occur exactly in the Pick (P2) or Place (P6) position

- At the Pick (P2) or Place (P6) position, the belt counter must be checked to see if the modulo value has been exceeded. However, the belt counter (SYNC command) modulo correction may only be carried out after desynchronization P4 or P8 (not during a synchronous phase)

Cause:

Belt counters are synchronous inputs (Belt), so they prevent blending.

Realization:

When performing the check, set local variables if the modulo value was exceeded and then use these variables for the modulo correction.

Restriction:

For the modulo correction, the "SYNC" command must be used, which also prevents blending. Therefore, set the modulo value as high as possible (e.g. 900000, but no more than 6 digits before the comma as otherwise the smaller values for belt changes within 2 ms do not have any affect when they are added).

To circumvent this restriction, the modulo correction should always be performed if objects or "objects for deposition" are expected.

Example Program

The following RCL program shows an example of a Pick & Place application with blending motions during belt synchronization.

Pick & Place - Application

```
;;COMPILER = RCL3
;;INT = LINEAR
;;KINEMATICS: ( 1 = Kin)
;;kin.JC_NAMES = Ax1,Ax2,Ax3,Ax4,eAx1,eAx2
;;kin.WC_NAMES = Kx,Ky,Kz,Kc,Kpk,Kpl

; *****
; Program description:
; - sample program for belt synchronous pick from infeed belt and belt synchronous
;   place on outfeed belt
; - pick and place positions are transmitted by the PLC
; - all movements between pick and place position (and vice versa) are blending
;   because: - path calculation of all blending moves is done during belt -
;             synchronous non blending movement (at point 'Pick' / 'Place')
;             - check for modulo range of belt and modulo correction of point is
;               done during belt synchronous non blending movement
;               (at point 'Pick' / 'Place')
;             - modulo correction of the belt is done during the non belt -
```

First Steps - Robot Control V1

```

;               synchronous movement
; *****
PROGRAM Pi_Pl_01
; *****
; *   DECLARATION   *
; *****
VAR:
INPUT POINT :
    701 = xPickPt_i,      ; pick position at pick belt from PLC
    702 = xPlacePt_i      ; place position at place belt from PLC
INPUT REAL :
    201 = rVel_i,         ; commanded velocity from PLC
    202 = rAccel_i        ; commanded acceleration from PLC
INPUT BINARY :
    1001 = bStop_i        ; asynchronous stop signal from PLC
OUTPUT BINARY :
    1 = bFlag_q
CONST :
    ; definitions
    FALSE = 0,
    TRUE = 1,
    C_BELT_NO_1 = 1,
    C_BELT_NO_2 = 2,
    C_BK_PTP = 4,         ; belt kind PTP
    MOD_VAL_B1 = 99000.0, ; modulo value for pick belt (mm)
    MOD_VAL_B2 = 99000.0 ; modulo value for place belt (mm)
BELT:
    500 + C_BELT_NO_1 = B1, ; define channel 501 as B1 (pick belt)
    500 + C_BELT_NO_2 = B2 ; define channel 502 as B2 (place belt)
POINT :
    Home,                ; world coordinate point definition
    PickApp,              ; home position
    Pick,                 ; point above point Pick to synchronize to B1
    PickDep,              ; pick position
    PickDepAsyn,          ; point above point Pick to move synchronous to
    PlaceApp,             ; point above POS to unsynchronize from B1
    Place,                ; point above point Place to synchronize to B2
    PlaceDep,             ; place position
    PlaceDepAsyn,         ; point above point Place to move synchronous to
    ; point above POS to unsynchronize from B2
REAL :
    rTotalLength,         ; variable to setup the belt area
    rBeginLength          ; variable to setup the belt area
BINARY :
    bSyncReq_B1,
    bSyncReq_B2
SPC_FCT :
    ; special function to setup the belt kind
    21 = belt_kind(VALUE INTEGER : belt_no ; number of belt to be configured
                   VALUE INTEGER : kind_belt) ; number of belt kind (only value 4)
SPC_FCT :
    ; special function to setup the belt area
    53 = belt_area(VALUE INTEGER : BeltNo ; number of belt to be configured
                   JC_POINT : @Base ; start pos. of working area on the belt
                   REAL : rTotalLength ; max. reachable length in belt direction
                   REAL : rBeginLength) ; latest pos. within the working area on
    ; the belt, where the synchronization
    ; move has to be started
; -----
; *   IMPLEMENTATION   *
; *****
BEGIN
Init ; call subroutine 'Init'
WAIT 5 ; wait time in seconds before proceeding with program
IF rVel_i > 0 THEN BEGIN
V = rVel_i ; get commanded velocity from PLC
END
IF (rAccel_i > 0) THEN BEGIN
A = rAccel_i ; get commanded acceleration from PLC
END
Pick = xPickPt_i ; get first pick position from PLC
PickApp = Pick ; calculate intermediate point 'PickApp' (with Kpk to synchronize)
PickApp.Kz = Pick.Kz + 100 ; set z coordinate of intermediate point
LoopMain:
; begin of main loop
PickB1 ; call subroutine 'PickB1'
PlaceB2 ; call subroutine 'PlaceB2'
If NOT bStop_i THEN BEGIN ; check for asynchronous stop signal from PLC
JUMP LoopMain ; proceed with main loop
END
MOVE TO Home ; move to home position (point 'Home')
PROGRAM_END

; -----
SUBROUTINE PickB1()
BEGIN
SYNCHRON B1 ; start synchronisation to pick belt with next movement

```

First Steps - Robot Control V1

```

MOVE VIA PickApp ; blending move via point 'PickApp' to synchronize to pick belt
MOVE TO Pick ; stopping move to point 'Pick' synchronous to pick belt
;----- * no blending from here * -----
WAIT 0.05
IF rVel_i > 0 THEN BEGIN
  V = rVel_i ; get commanded velocity from PLC
END
IF (rAccel_i > 0) THEN BEGIN
  A = rAccel_i ; get commanded acceleration from PLC
END
PickDep = Pick ; calculate intermediate point 'PickDep'
PickDep.Kz = Pick.Kz + 50
PickDepAsyn = POS ; calculate intermediate point 'PickDepAsyn' (from actual Pos.)
PickDepAsyn.Kz = Pick.Kz + 100
PickDepAsyn.kpk = 0 ; set belt value to zero to define synchronization position
Place = xPlacePt_i
IF B1 > MOD_VAL_B1 THEN BEGIN
  bSyncReq_B1 = TRUE ; belt modulo has to be done
END
ELSE BEGIN
  bSyncReq_B1 = FALSE
END
IF Place.kpl > B2 THEN BEGIN
  Place.kpl = Place.kpl - MOD_VAL_B2 ; modulo correction for the place position
END
PlaceApp = Place
PlaceApp.Kz = Place.Kz + 100
PlaceDep = Place
PlaceDep.Kz = Place.Kz + 50
;----- * no blending till here * -----
MOVE VIA PickDep ; blending move via point 'PickDep' synchronous to pick belt
MOVE VIA PickDepAsyn ; blending move via point 'PickDepAsyn' to unsynchronize
SYNCHRON_END B1
IF bSyncReq_B1 = TRUE THEN BEGIN
  SYNC B1 >= MOD_VAL_B1 ; belt modulo (subtraction)
END
RETURN
SUB_END

;-----
SUBROUTINE PlaceB2()
BEGIN
  SYNCHRON B2
  MOVE VIA PlaceApp
  MOVE TO Place
  ;----- * no blending from here * -----
  WAIT 0.05
  IF rVel_i > 0 THEN BEGIN
    V = rVel_i ; get commanded velocity from PLC
  END
  IF (rAccel_i > 0) THEN BEGIN
    A = rAccel_i ; get commanded acceleration from PLC
  END
  PlaceDepAsyn = POS ; calculate desynchronization point 'PlaceDepAsyn' depending
  ; on actual position
  PlaceDepAsyn.Kz = Place.Kz + 100
  PlaceDepAsyn.kpl = 0
  Pick = xPickPt_i ; get next pick position from PLC
  IF B2 > MOD_VAL_B2 THEN BEGIN ; check if belt counter has to be modulated
    bSyncReq_B2 = TRUE ; belt modulo has to be done
  END
  ELSE BEGIN
    bSyncReq_B2 = FALSE
  END
  IF Pick.kpk > B1 THEN BEGIN
    Pick.kpk = Pick.kpk - MOD_VAL_B1 ; modulo correction for the pick position
  END
  PickApp = Pick ; calculate intermediate point 'PickApp' (with Kpk to synchronize)
  PickApp.Kz = Pick.Kz + 100
  PickDep = Pick
  PickDep.Kz = Pick.Kz + 50
  ;----- * no blending till here * -----
  MOVE VIA PlaceDep
  MOVE VIA PlaceDepAsyn
  SYNCHRON_END B2
  IF bSyncReq_B2 = TRUE THEN BEGIN ; use 'SYNC' only if B1 > MOD_VAL_B1 otherwise it will
    ; avoid blending
    SYNC B2 >= MOD_VAL_B2 ; belt modulo (subtraction)
  END
RETURN
SUB_END

;-----

```

First Steps - Robot Control V1

```
SUBROUTINE Init()
VAR:
  POINT:
    Base
  JC_POINT: ; joint coordinate point definition
  @Base ; point to setup the startpoint of the belt area
BEGIN
  Home = (0,0,-700,0,0,0) ; define point 'Home'
  V = 1000 ; define path velocity
  A = 1000 ; define path acceleration
  R = 50 ; define maximum blending radius (mm)
  AFACTOR = 1 ; define factor for acceleration (factor for A)
  DFACTOR = 1 ; define factor for deceleration (factor for A)
  BLOCK_SLOPE ; define kind of slope for blending
  belt_kind(C_BELT_NO_1, C_BK_PTP) ; define belt kind for pick belt
  belt_kind(C_BELT_NO_2, C_BK_PTP) ; define belt kind for place belt
  rTotalLength = 600 ; define length of working area over the belt
  rBeginLength = 550 ; define latest position within the working area,
  ; where the synchronization movement has to be started
  Base = (-300,0,-650,0,0,0) ; set point Base in world coordinates to define base
  ; of the pick belt
  @Base = JC(Base) ; convert point Base to joint coordinates and copy to
  ; @Base
  belt_area(C_BELT_NO_1, ; execute special function to setup the working area
    @Base, ; of the pick belt
    rTotalLength,
    rBeginLength)

  Base = (-300, 100, -650, 0, 0, 0) ; set point Base in world coordinates to define base
  ; of the place belt
  @Base = JC(Base) ; convert point Base to joint coordinates and copy to
  ; @Base
  belt_area(C_BELT_NO_2, ; execute special function to setup the working area
    @Base, ; of the place belt
    rTotalLength,
    rBeginLength)

  SYNC B1, bStop_i = FALSE ; set belt counter to 0
  SYNC B2, bStop_i = FALSE ; set belt counter to 0
  MOVE TO Home ; move to home position (point 'Home')
RETURN
SUB_END
```

6 First Steps - Robot Control V2

6.1 Project Goal "First Steps - Robot Control V2"

A simple example should explain the general process of a Robot Control V2 project creation using IndraMotion MLC. The keywords of the minimum configuration required in IndraWorks and the PLC programming are addressed. It covers the procedure from the example project creation up to the minimum required configuration in IndraWorks and the PLC programming.

In the example project "RoCoV2TestV13", it is assumed that only virtual axes are available. Finally, the integration of real axes is described.



The user can either restore the example project from the IndraWorks installation DVD from the directory "\AddOns\MLC\Example-Project\RobotControl" which contains a .zip file with the following content:

- the parameter export file (*.par),
- PLC program
- the visualization data

- or -

the user can create the project step by step executing the following description. The detailed data provided in the appendix is to be used.

6.2 Required Hardware and Software

Required Hardware *Minimum:*

- Control, type "IndraMotion MLC Lx5",
- Current firmware is loaded
- IP address is known,
- Programming device (PC / laptop)
- Control is connected to the programming device via Ethernet

Optional:

- 3 real axes (e.g. IndraDrive, EcoDriveCS,...)
- Current firmware (starting from MPx07) is loaded

Required software IndraWorks installation 13VRS using MLC as target system.

6.3 Procedure when Creating an RoCo Project

Execute the following steps for a step-by-step generation of an example project:

General approach

1. Starting IndraWorks and creating a new project

After starting IndraWorks, create the new project via **File ► New ► Project....**

First Steps - Robot Control V2

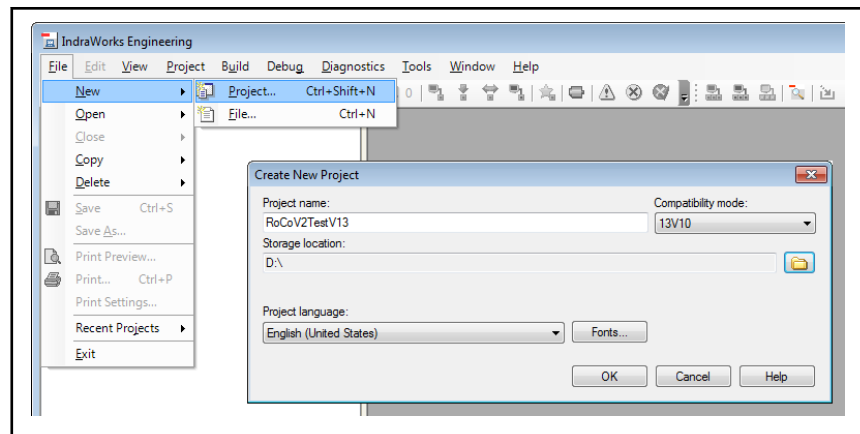


Fig.6-1: Creating a new IndraWorks project

Specify a name for the project (e.g. "RocoV2TestV13").

Select a storage location.

By selecting the project language, the language of the user texts is determined.

Confirm with **OK**.

The created project is displayed in the Project Explorer under the defined project name.

2. Creating and configuring the IndraMotion MLC control

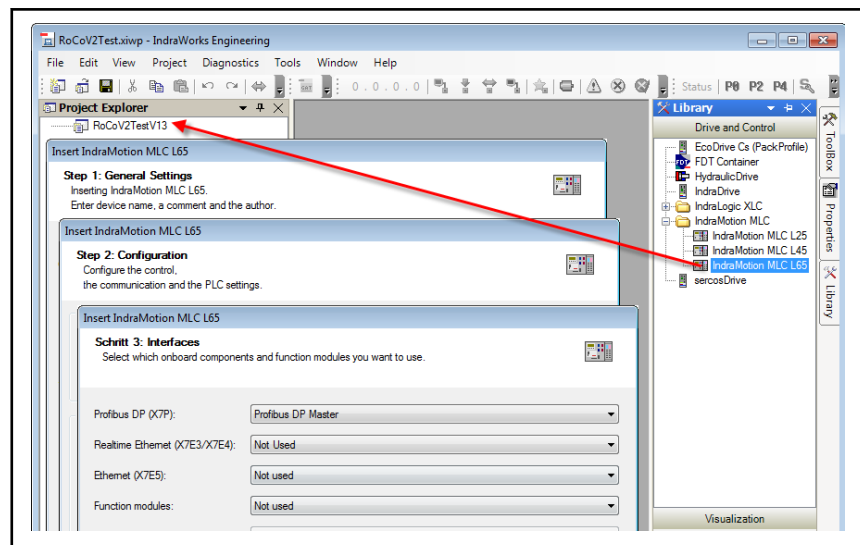


Fig.6-2: Creating and configuring an MLC

Drag the desired device, (e.g. "IndraLogic MLC L65") from the "Drive and Control" library to the newly created project in the Project Explorer.

Subsequently, the wizard automatically starts to set the basic settings.

Enter a user-defined device name (e.g. "mlc_L65") in the "General Settings" dialog and confirm with **Next**.

In the "Configuration" dialog, define the settings for the firmware and communication between IndraWorks and the device.

The default settings for the "device type", the "firmware version" and the "firmware release" do not have to be changed.

First Steps - Robot Control V2

Enter the target address in "IP address", i.e. the IP address of the control where the application should be loaded.

Execute the connection test and verify that the communication between PC and control has been established successfully.

Confirm with the **Next**.

In the "Interfaces" dialog, no other settings have to be selected unless no function modules are connected to the control.

Confirm with the **Finish** button.

3. Creating and configuring three virtual axes

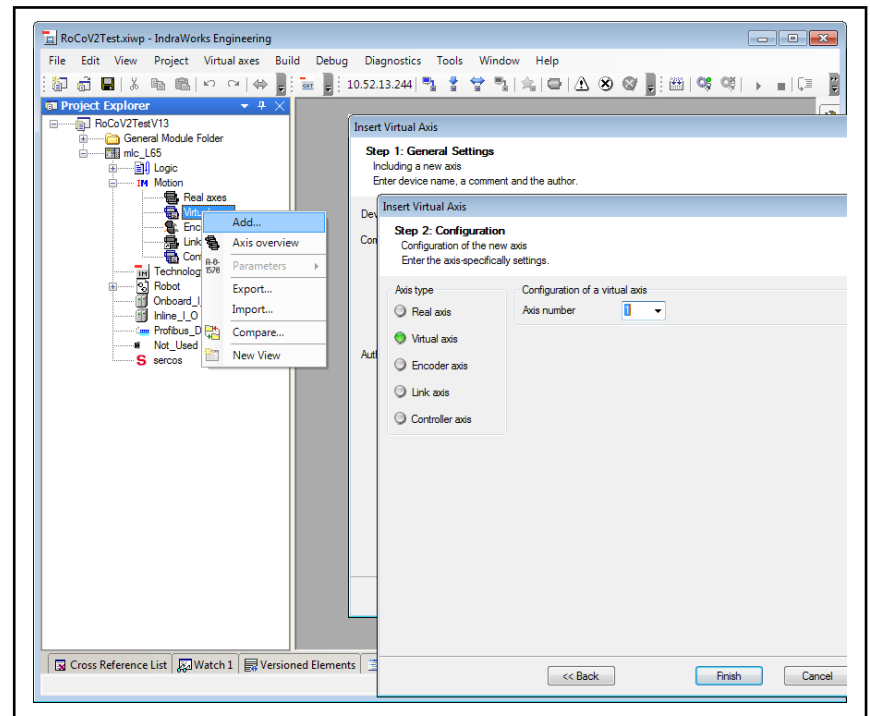


Fig.6-3: Creating virtual axes

Right-click on "Virtual axes" and select "Add" in the context menu.

Enter the name "Axis1" in the "General Settings" dialog under "Device name" and confirm with **Next**.

Enter the axis number "1" and confirm with "Finish" in the "Configuration" dialog.

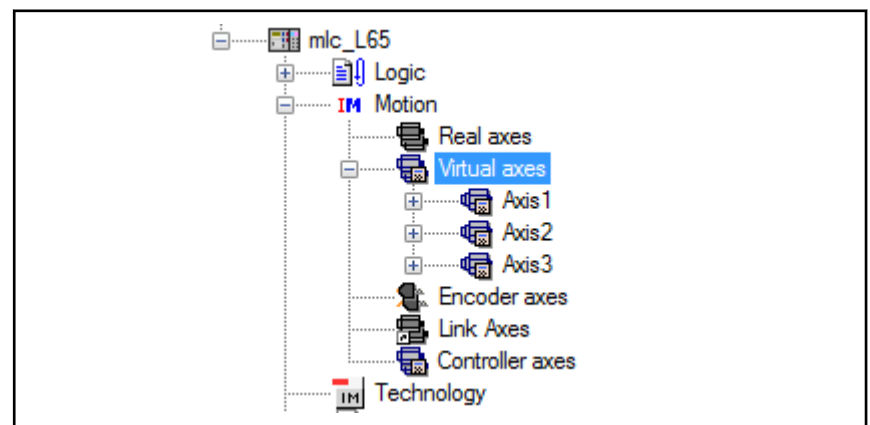


Fig.6-4: Virtual axis created

First Steps - Robot Control V2

Execute these steps for two more virtual axes. Consequently, three virtual axes are displayed under the "Virtual axes" node.

4. Add and configure kinematics

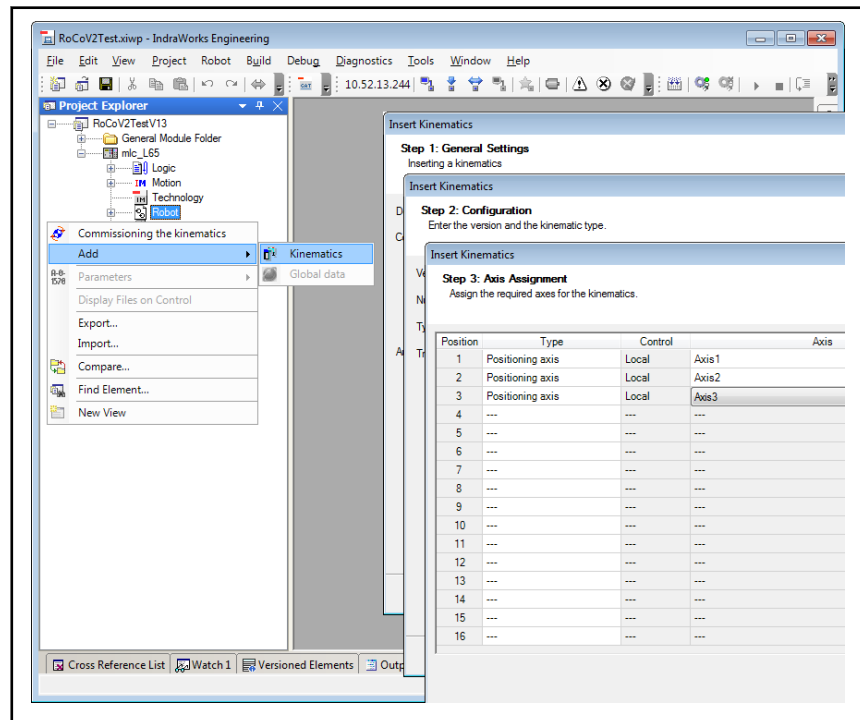


Fig.6-5: Add and configure kinematics

Right-click on the Robot node and select "Add" and "Kinematics" in the context menu.

Enter a user-defined "Device name" (e.g. Kin01) in the "General Settings" dialog and confirm with **Next**.

Select the kinematics type "Cartesian" in the "Configuration" dialog and confirm with **Next**.

Define three positioning axes in the "Axis assignment" dialog and assign the virtual axes.

Subsequently, complete the operation with the **Finish** button.

5. Downloading the Motion configuration to the device

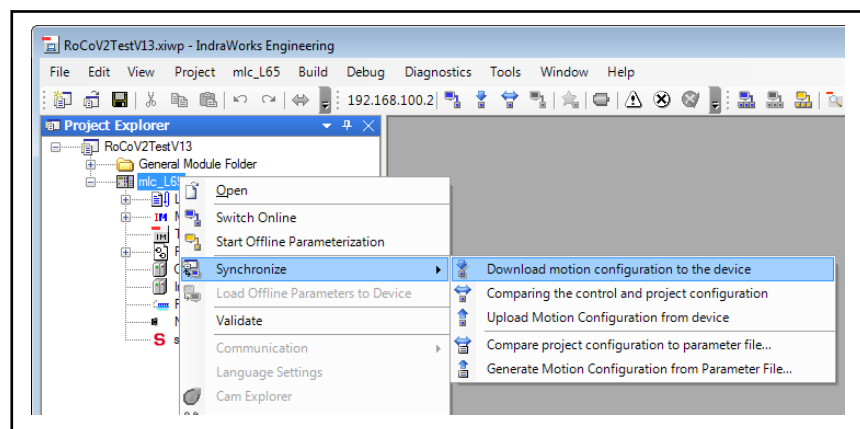


Fig.6-6: Downloading the Motion configuration to the device

First Steps - Robot Control V2

Right-click on the control node and select "Synchronize" and select "Download Motion configuration to the device" from the context menu. The selected settings are applied to the control.

6. Import parameters

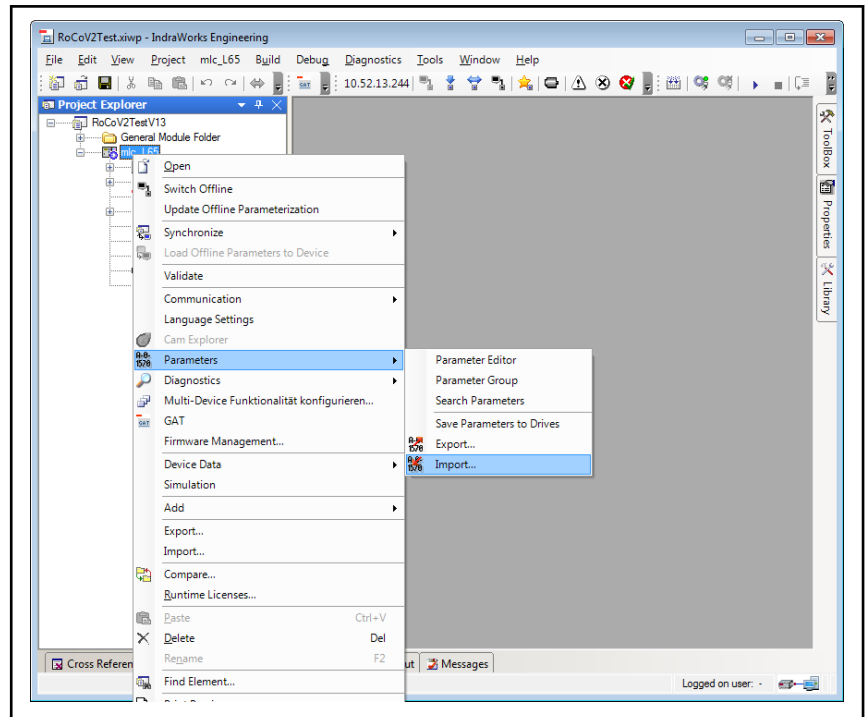


Fig.6-7: Import parameters

Right-click on the control node and select "Parameters" and "Import" in the context menu.



The parameter file can be found in the project directory on the installation DVD: "\\AddOns\\MLC\\Example-Project\\RobotControl\\RoCoV2TestV13\\RoCoV2TestV13virtuell.par".

First Steps - Robot Control V2

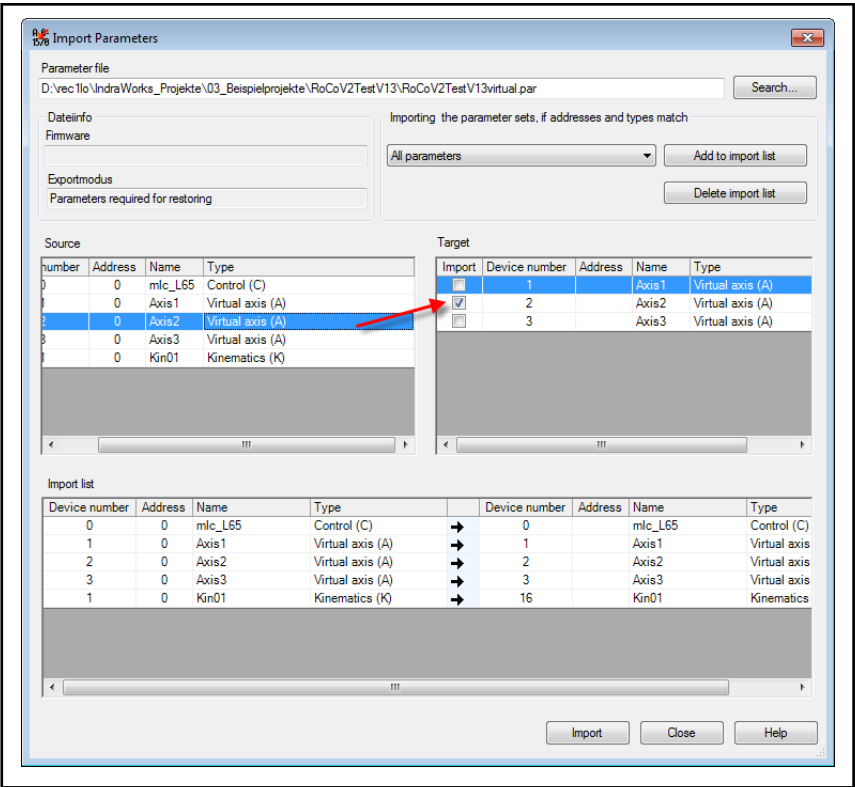


Fig.6-8: *Select the parameters*

Subsequently, the wizard automatically opens for the import. Assign a target to each source to be able to import the control, axis and kinematic parameters. Use the **Import** button to confirm the entry.

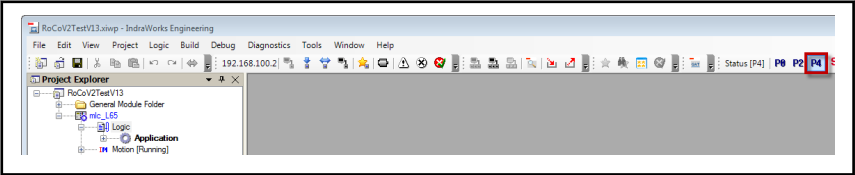


Fig.6-9: *Ready for operation state*

After a successful parameter import, it should be possible to switch the control to BB (ready for operation) (P4).

7. Creating a PLC program

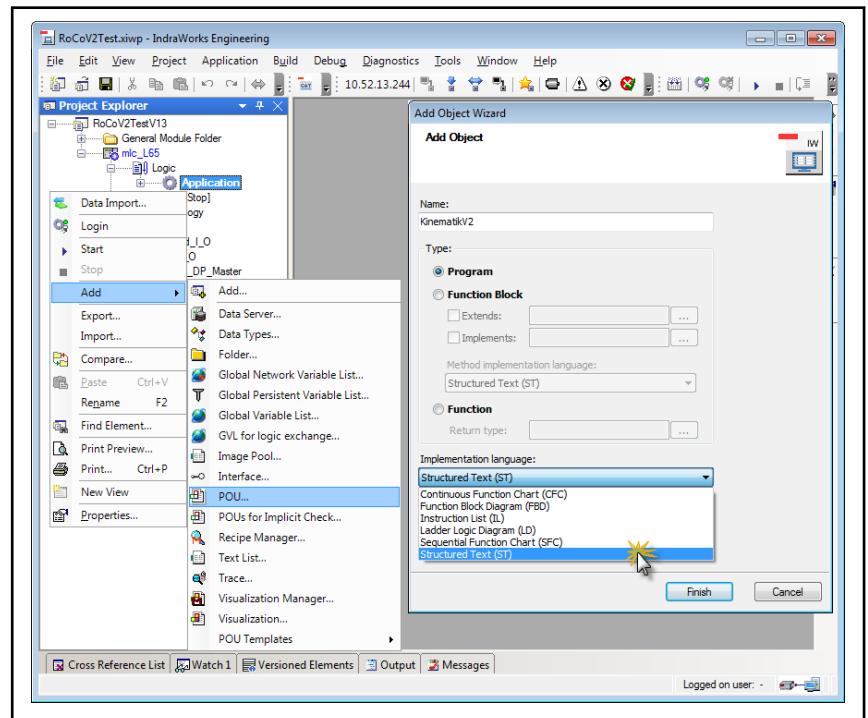


Fig. 6-10: Creating a PLC program

Right-click on the "Application" node and select "Add" and "POU" in the context menu.

Enter a user-defined program name (e.g. KinematikV2) in the "Add object" dialog. Select the object type "Program" and "Structured Text (ST)" as implementation language.

Add the program with declaration part contained in "Details for project RoCoV2TestV13 - PLC program listing (kinematikV2)".

Add the program with declaration part contained in "Details for project RoCoV2TestV13 - PLC program listing (PlcProg)" in the "PlcProg" program.

All programs have to be compiled successfully via .

8. Add program call to MotionTask

Open the task configuration by double-clicking on the MotionTask.

First Steps - Robot Control V2

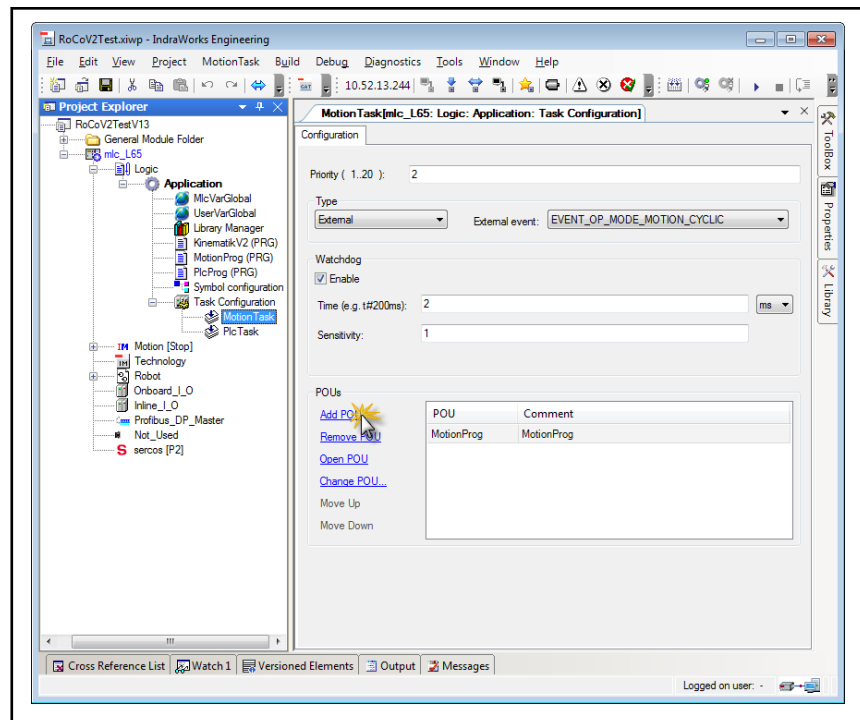


Fig.6-11: Configuring the task

Add the PLC program call. Select the previously created PLC program via "Add call" and confirm with **OK**.

9. Integrating the visualization

A visualization is provided in the project direction on the installation DVD ("**\\AddOns\\MLC\\Example-Project\\RobotControl\\RoCoV2TestV13\\Visualization.iwx**") to operate the PLC program. Using this visualization, all required operations can be executed.

First Steps - Robot Control V2

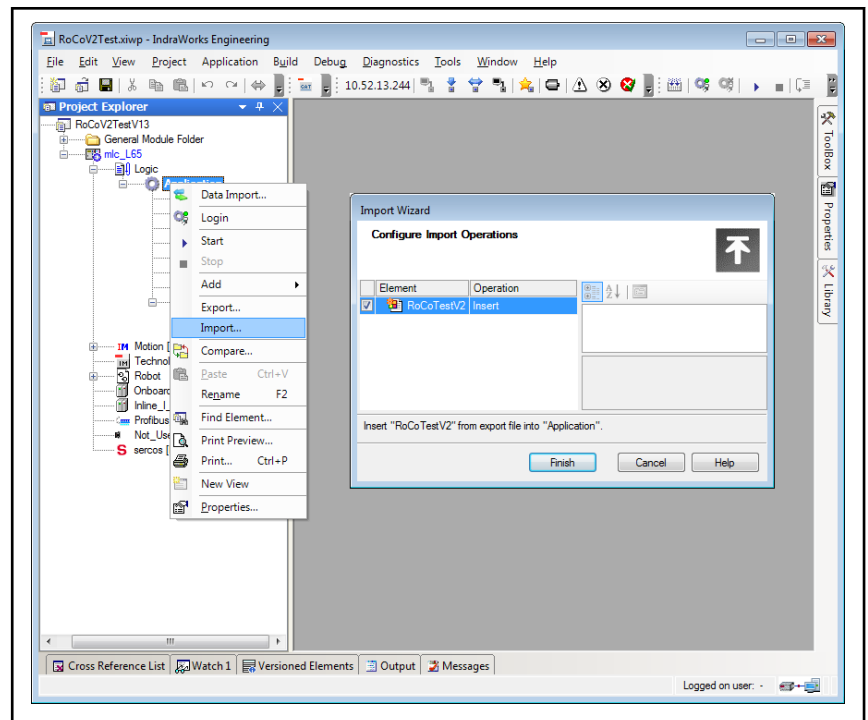


Fig.6-12: Importing the visualization

Right-click on "Application" and select "Import" in the context menu.

Select the "Visualization" file from the project directory and confirm with **Open**.

Subsequently, confirm the dialog with **Finish**.

10. Download the PLC project

By logging in to the control, the PLC program is downloaded to the control.

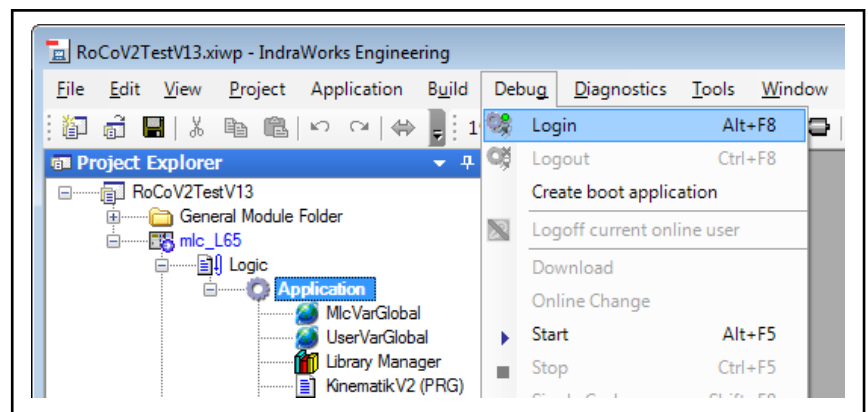


Fig.6-13: Login

Select **Debug** ► **Login**. Subsequently, the control can be started.

First Steps - Robot Control V2

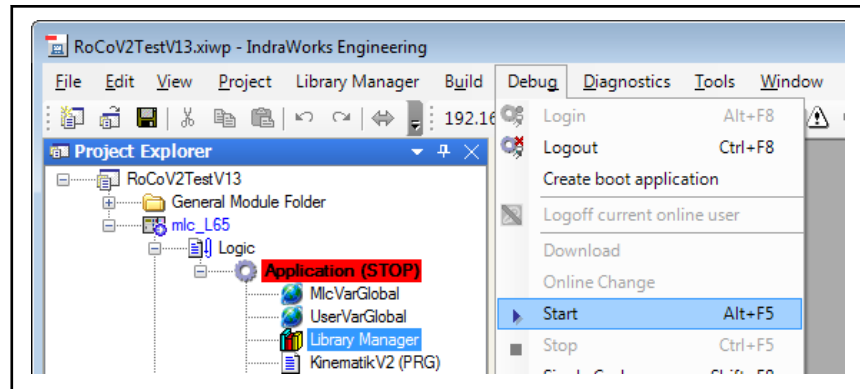


Fig.6-14: Starting the application

Select **Debug ▶ Start**.

11. Start the motion using the visualization

The motion sequence can be prepared, started or stopped using the imported visualization.

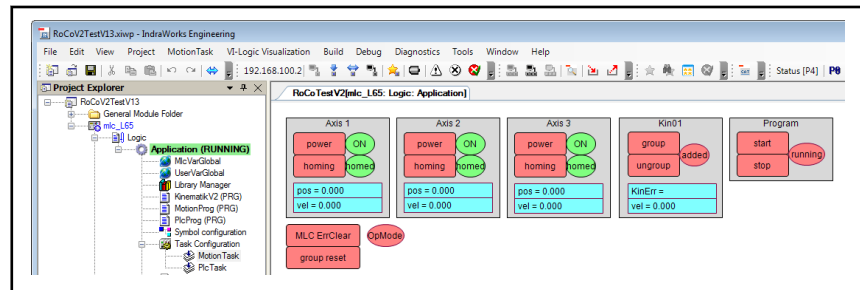


Fig.6-15: Starting the motion using visualization

Group the axes using the **group** button. If the axes have been grouped successfully, the field "added" changes to green. All axes belonging to the defined kinematics are grouped in this step.

Start the program using the **start** button. The program has been started successfully once the "running" field changes to green.

Similarly, the program can be stopped (**stop** button) and the kinematics can be ungrouped (**ungroup** button).

In case of an error, the kinematics can be reset via **group reset**. All control and drive errors can be cleared via **MLC ErrClear** (C-0-1030).

Integrating real axes

Real axes can be used instead of virtual axes for the kinematics. Connect three real drives to the control via the sercos bus and execute the following steps.

First Steps - Robot Control V2

1. Deleting the virtual axes

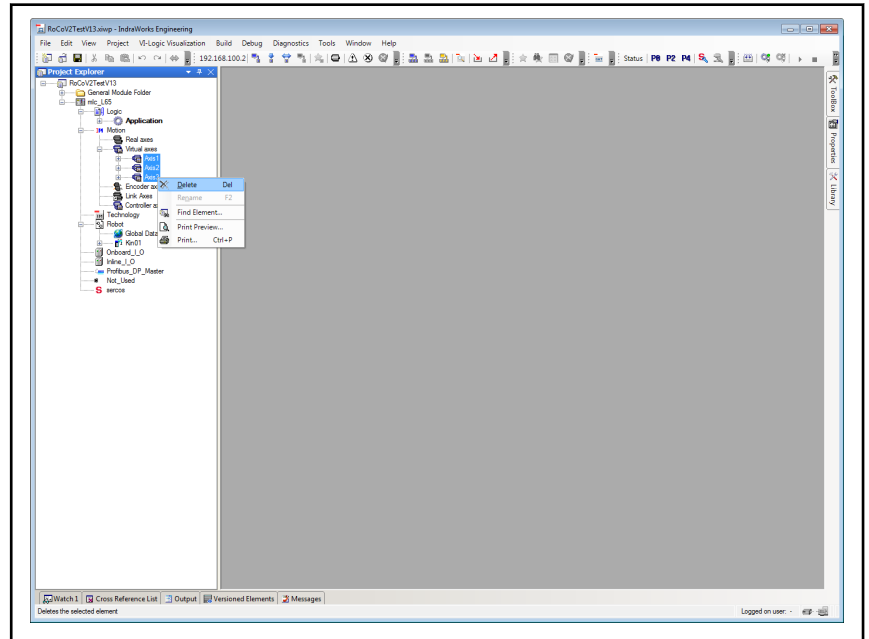


Fig.6-16: Deleting the virtual axes

Select all three virtual axes. Right-click on the selected axes and select "Delete" in the context menu.

2. Create and configure three real drives

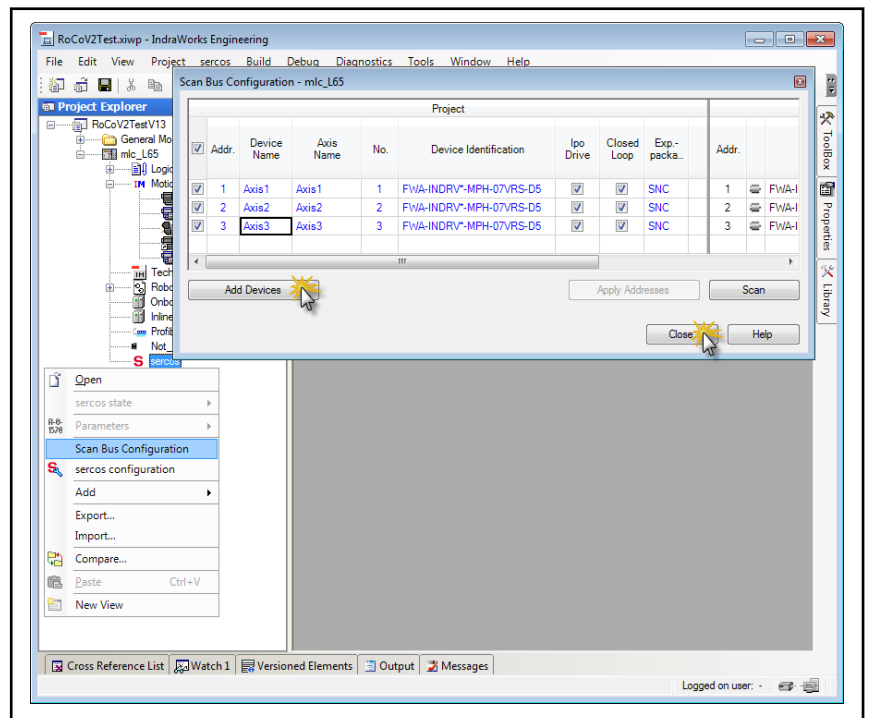


Fig.6-17: Creating real axes

Right-click on the sercos node and select "Scan bus configuration" in the context menu.

First Steps - Robot Control V2

Select the names of the deleted virtual axes as device names (Axis1, Axis2, Axis3). Thus, it is guaranteed that the PLC program continues to run without adjustments.

Add the axes by clicking on **Add Devices** and close the window with **Close**.

3. Adjusting the kinematic configuration

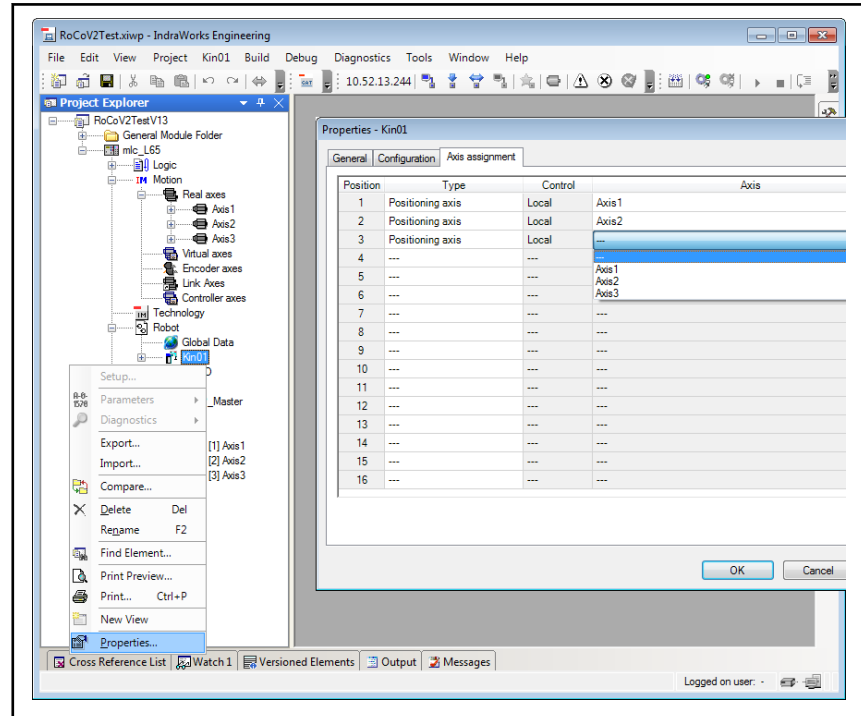


Fig.6-18: Adjusting the kinematic configuration

Open the property dialog of the kinematics and adjust the axis assignments to the new real axes.

Confirm the input with **OK**.

4. Downloading the Motion configuration and importing the parameters

Download the Motion configuration to the control according to the known procedure and import the parameters for the real axes. Control and kinematic parameters do not have to be imported again!



The parameter file for real axes can be found in the project directory on the installation DVD: "AddOns\MLC\Example-Project\RobotControl\RoCoV2TestV13\RoCoV2TestV13real.par".

The kinematics can now be operated with the visualization. Note that real axes have to be connected **power**) - in contrast to virtual axes - and, unless no absolute value encoder is used, homing has to be executed (**homing**). Only then, grouping can take place.

6.4 Details on the Project "RoCoV2TestV13"

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 P0 *)
fbMoveDirectAbsoluteP0(
    Execute      := bMoveDirectAbsoluteP0,
    Position     := arPositionP0,
    Velocity     := rVFac,
    Acceleration := rAFac,
    Deceleration := rDFac,
    Jerk         := rJFac,
    CoordSystem  := CS_MACHINE_COORDINATES,
    SlopeType    := ST_BLOCK_SLOPE,
    SetMode      := SM_T0,

```

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_T0,
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);

```

7 Service and Support

Our worldwide service network provides an optimized and efficient support. Our experts offer you advice and assistance should you have any queries. You can contact us **24/7**.

Service Germany Our technology-oriented Competence Center in Lohr, Germany, is responsible for all your service-related queries for electric drive and controls.

Contact the **Service Helpdesk & Hotline** under:

Phone:	+49 9352 40 5060
Fax:	+49 9352 18 4941
E-mail:	service.svc@boschrexroth.de
Internet:	http://www.boschrexroth.com

Additional information on service, repair (e.g. delivery addresses) and training can be found on our internet sites.

Service worldwide Outside Germany, please contact your local service office first. For hotline numbers, refer to the sales office addresses on the internet.

Preparing information To be able to help you more quickly and efficiently, please have the following information ready:

- Detailed description of malfunction and circumstances resulting in the malfunction
- Type plate name of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your email address)

Index

A

Abbreviations.....	12
About this documentation.....	5
Validity of the documentation.....	5
Application phases.....	5

D

Documentation structure.....	5
------------------------------	---

E

Export file	
First steps - FlexProfile.....	141
First steps - IndraLogic.....	64
First steps - Motion.....	123

F

First steps - FlexProfile.....	127
Create FlexProfile.....	129
Create FlexProfile via IndraWorks dialog.....	129
Create FlexProfile via PLC.....	132
Create the Motion subproject.....	128
Creating a Motion subproject.....	68
Download PLC subproject FlexProfile.....	139
Execute a motion step, ML_FlexProfile.....	135
Restore project.....	141
Technical requirements.....	128
Trace_FlexProfile.....	141
Workflow virtual axis → Virtual axis → FlexProfile 0 → Real axis.....	140
First steps - IndraLogic.....	13
Add new device to a project.....	15
Adding and editing an element.....	60
Basic settings to the device	16
Communication settings.....	16
Copying the visu element.....	57
Create new project.....	14
Create new visualization.....	53
Create visualization for PLC program.....	53
Declare global variables.....	41
Device settings.....	16
Edit AS (SFC).....	32
Edit PlcProg in CFC.....	47
Edit PlcProg in ST.....	44
Edit programs and function blocks – Sec- ond part.....	44
Edit ST.....	30
Edit visualization.....	54
Function module settings.....	18
New IndraWorks project	14
Restore.....	64
Restoring a project.....	64
Start IndraWorks.....	14
Task configuration.....	40
Technical requirements.....	13
Test in simulation mode.....	42

Test visualization.....	63
Visu element button.....	61
Visu element ellipse.....	54
Visu-element display of variable values.....	62
Visu-element labeling.....	61
Visualization in IndraLogic.....	53
First Steps - IndraLogic	
Add function block.....	23
Add POU.....	20, 24
Add program.....	22
Controlling a traffic signal system.....	20
Create PLC program.....	20
Edit FBD.....	24
Programs and function blocks.....	20
Rectangle visu element.....	58
First steps - Motion.....	67
Add a virtual axis.....	74
Add new device to project.....	69
Basic settings for the device	69
Communication settings.....	70
Create new project	68
Device settings.....	69
Download of the PLX sub-project VA1 (FBD).....	85
Download PLC projects RA1 (FBD).....	97
Download PLC projects RA1 (ST).....	120
Download the Motion subproject.....	75
Download the PLC project VA1 (ST).....	107
Function module settings.....	72
Program real axis RA1 (FBD).....	94
Program real axis RA1 (ST).....	117
Program virtual axis VA1 (FBD).....	79
Program virtual axis VA1 (ST).....	101
Restore project.....	123
Scan sercos III drives.....	73
Start IndraWorks.....	68
Technical requirements.....	67
Trace_VA1 (FBD).....	88
Trace_VA1 (ST).....	111
Trace_VA1_RA1 (FBD).....	98
Trace_VA1_RA1 (ST).....	122
Workflow virtual axis → Electronic gear → Real axis.....	97, 121
Write values, input variables (FBD).....	87
Write values, input variables (ST).....	109
First Steps - Motion	
Program motion sequence (ST).....	100
Programming a Motion sequence (FBD).....	78
First steps - Robot Control V1.....	143
Application examples.....	156
Creating a RoCo project.....	144
Pick & Place ,belt synchronization with blending motions.....	160
Pick & Place Applications.....	156
Pick & Place, belt synchronization.....	158
Pick & Place, blending.....	156

Index

Required hardware and software..... 143

First Steps - Robot Control V1

 RoCoTestV** 154

First steps - Robot Control V2..... 167

 Create a RoCo project..... 167

 Required hardware and software..... 167

First Steps - Robot Control V2

 RoCoTestV2..... 178

N

Names..... 12

Names and abbreviations..... 12

S

Safety alert symbol..... 6

Safety instructions..... 6

Signal words..... 6

Support

 See service hotline..... 183

Symbols..... 6

T

Target group..... 5

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



R911336293

DOK-MLC***-F*STEP**V13-CO02-EN-P