

Rexroth IndraMotion MLC 12VRS First Steps

R911333846
Edition 01

Commissioning Manual



Title	Rexroth IndraMotion MLC 12VRS First Steps								
Type of Documentation	Commissioning Manual								
Document Typecode	DOK-MLC***-F*STEP**V12-CO01-EN-P								
Internal File Reference	RS-bec1385da4450f050a6846a0005b156d-1-en-US-2								
Purpose of Documentation	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. It is the basis for the online help in the system.								
Record of Revision	<table><tr><th>Edition</th><th>Release Date</th><th>Notes</th></tr><tr><td>Edition 01</td><td>03.2011</td><td>First edition</td></tr></table>			Edition	Release Date	Notes	Edition 01	03.2011	First edition
Edition	Release Date	Notes							
Edition 01	03.2011	First edition							
Copyright	© Bosch Rexroth AG 2011 Copying this document, giving it to others and the use or communication of the contents thereof without express authority, are forbidden. Offenders are liable for the payment of damages. All rights are reserved in the event of the grant of a patent or the registration of a utility model or design (DIN 34-1).								
Validity	The specified data is for product description purposes only and may not be deemed to be guaranteed unless expressly confirmed 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 (0)93 52/40-0 • Fax +49 (0)93 52/40-48 85 http://www.boschrexroth.com/ System IndraLogic & Basis Motion Logic HB (vha/MePe)								
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 Required and Supplementing Documentations.....	5
2 Important Instructions on Use.....	11
2.1 Appropriate Use.....	11
2.1.1 Introduction.....	11
2.1.2 Areas of Use and Application.....	11
2.2 Inappropriate Use.....	12
3 Safety Instructions for Electric Drives and Controls.....	13
3.1 Definitions of Terms.....	13
3.2 General Information.....	14
3.2.1 Using the Safety Instructions and Passing Them on to Others.....	14
3.2.2 Requirements for Safe Use.....	14
3.2.3 Hazards by Improper Use.....	15
3.3 Instructions with Regard to Specific Dangers.....	17
3.3.1 Protection Against Contact With Electrical Parts and Housings.....	17
3.3.2 Protective Extra-Low Voltage as Protection Against Electric Shock	18
3.3.3 Protection Against Dangerous Movements.....	18
3.3.4 Protection Against Magnetic and Electromagnetic Fields During Operation and Mounting.....	20
3.3.5 Protection Against Contact With Hot Parts.....	20
3.3.6 Protection During Handling and Mounting.....	21
3.3.7 Battery Safety.....	21
3.3.8 Protection Against Pressurized Systems.....	22
3.4 Explanation of Signal Words and the Safety Alert Symbol.....	22
4 First Steps - IndraLogic.....	25
4.1 Project Goal "First steps - IndraLogic".....	25
4.2 Technical Requirements.....	25
4.3 Creating a New IndraWorks Project.....	26
4.3.1 Resetting the Control (CMLx5).....	26
4.3.2 Starting IndraWorks.....	26
4.3.3 Creating a New Project.....	26
4.4 Adding a New Device to a Project.....	27
4.5 Allocating Basic Settings to the Device.....	28
4.5.1 Device Settings.....	28
4.5.2 Communication Settings.....	29
4.5.3 Function Module Settings.....	30
4.6 Creating a PLC Program.....	32
4.6.1 The Control of a Traffic Light System.....	32
4.6.2 Creating Programs and Function Blocks.....	32
Adding "PlcProg" program in "CFC" or in "ST".....	32

Table of Contents

	Page
Adding "SEQUENCE" Program in "AS".....	34
Adding "TRAFFIC SIGNAL" program in "FBD".....	35
Adding Function Block "WAIT" in "ST".....	36
4.6.3 Editing Programs and Function Blocks - Part 1.....	37
Editing "TRAFFIC SIGNAL" Function Block in "FBD".....	37
Editing "WAIT" Function Block in "ST".....	43
Editing "SEQUENCE" Program in "AS".....	45
4.6.4 Task Configuration.....	55
4.6.5 Declaring Global Variables.....	56
4.6.6 Testing "SEQUENCE" in Simulation Mode.....	57
4.6.7 Editing Programs and Function Blocks – Second Part.....	60
Editing "PlcProg" Program in "ST".....	60
Testing the Application in Simulation Mode, PlcProg in ST.....	61
Editing "PlcProg" Program in "CFC".....	62
Testing the Application in Simulation Mode, PlcProg in CFC.....	66
4.6.8 Testing the Application on the Control.....	67
4.7 Creating Visualization for the PLC Program.....	69
4.7.1 Visualization in IndraLogic.....	69
4.7.2 Creating a New Visualization.....	69
4.7.3 Editing Visualization.....	70
Adding and Editing the First Element "Signal Lamp".....	70
Copying, Adding and Editing the "Signal Lamp" Element.....	73
Adding and Editing the "Traffic Light Housing" Element.....	74
Adding and Editing "Traffic Light Mast" Element.....	75
Adding and Editing "Traffic Light Base" Element.....	75
Adding and Editing Two Traffic Lights.....	76
Adding and Editing an ON Switch.....	77
Adding and Editing Labeling.....	78
Displaying Variable Values.....	79
4.7.4 Testing Visualization.....	80
4.8 Restoring a Project.....	81
5 First Steps - Motion.....	83
5.1 Project Goal "First Steps - Motion".....	83
5.2 Technical Requirements.....	83
5.3 Create the Motion Subproject.....	84
5.3.1 Resetting the Control (CMLx5).....	84
5.3.2 Starting IndraWorks.....	84
5.3.3 Creating a New Project.....	84
5.3.4 Adding a New Device to a Project.....	85
5.3.5 Specifying Basic Settings for the Device.....	86
Device Settings.....	86
Communication Settings.....	87
Function Module Settings.....	88
5.3.6 Adding SERCOS III Drives (Real Axes) to the Project.....	89
5.3.7 Adding a Virtual Axis to the Project.....	90

Table of Contents

	Page
5.3.8 Downloading the Motion Subproject.....	91
5.4 Programming the Motion Sequence in Function Block Diagram (FBD).....	93
5.4.1 Programming a Motion Sequence in FBD, General Information.....	93
5.4.2 Programming the Virtual Axis VA1, Programming Language FBD.....	95
5.4.3 Download of the PLC Sub-project Virtual Axis VA1 (FBD).....	102
5.4.4 Recording the Motion of Axis VA1 (Trace), FBD.....	106
5.4.5 Programming the Real Axis RA1 (FBD).....	111
5.4.6 Downloading the Complete PLC Project (FBD).....	115
5.4.7 Workflow "Virtual Axis → Electronic Gear → Real Axis ".....	115
5.4.8 Recording the Motion of Both Axes (Trace), FBD.....	116
5.5 Programming the Motion Sequence in Structured Text (ST).....	117
5.5.1 Programming a Motion Sequence in ST, General Information.....	117
5.5.2 Programming the Virtual Axis VA1, Programming Language ST.....	119
5.5.3 Downloading the PLC Sub-project Virtual Axis VA1 (ST).....	125
5.5.4 Recording the Motion of Axis VA1 (Trace), ST.....	129
5.5.5 Programming the Real Axis RA1 (ST).....	134
5.5.6 Downloading the Complete PLC Project (ST).....	138
5.5.7 Workflow "Virtual Axis → Electronic Gear → Real Axis ".....	138
5.5.8 Recording the Motion of Both Axes (Trace), ST.....	140
5.6 Restoring the Project.....	141
6 First Steps - FlexProfile.....	143
6.1 Project Goal "First steps - FlexProfile".....	143
6.2 Technical Requirements.....	144
6.3 Creating the Motion Subproject.....	144
6.4 Creating a FlexProfile.....	145
6.4.1 Creating a FlexProfile, General Information.....	145
6.4.2 Creating the FlexProfile via IndraWorks Dialogs.....	145
6.4.3 Creating the FlexProfile via PLC.....	147
6.5 Completing the PLC Subproject, Executing a FlexProfile Motion Step.....	151
6.6 Downloading and Testing the PLC Subproject.....	154
6.7 Workflow "Virtual Axis → FlexProfile 0 → Real Axis ".....	155
6.8 Recording the Motion of Both Axes (Trace).....	156
6.9 Restoring the Project.....	157
7 First Steps - RobotControl.....	159
7.1 Project Goal "First Steps - RobotControl".....	159
7.2 Required Hardware and Software.....	159
7.3 Procedure when Creating an RoCo Project	160
7.4 Details on the Project "RoCoTestV12".....	171
7.5 RobotControl - Application examples.....	173
7.5.1 RoCo Basics for Pick & Place Applications.....	173
General.....	173
Blending.....	173
Belt Synchronization.....	175

Table of Contents

	Page
Belt Synchronization with Blending Motions.....	177
Instructions for Blending during Belt Synchronization.....	178
Example Program.....	179
 8 Service and Support.....	 183
 Index.....	 185

1 About this Documentation

1.1 Validity of the Documentation

- Target group** This documentation is intended for users new to the topic using an IndraMotion MLC control to get started with the first steps of the IndraWorks Engineering system.
- The documentation includes the chapters:
- [First Steps - IndraLogic, page 25](#),
basics of programming with IndraLogic and visualization,
The chapter can also be understood without control in the simulation mode (soft PLC).
 - [First Steps - Motion, page 83](#),
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).
 - [First Steps - FlexProfile, page 143](#),
creation and application of MotionProfiles.
This chapter is based on [First Steps - Motion](#).
 - [First Steps - RobotControl, page 159](#),
creation of a RobotControl project until the execution of the RoCo source code
- General notes** The following chapters provide information on
- [Required and Supplementing Documentations, page 5](#),
 - [Important Instructions on Use, page 11](#), and
 - [Safety Instructions for Electric Drives and Controls, page 13](#).

1.2 Required and Supplementing Documentations

Documentation titles with type designation codes and parts numbers

IndraWorks		MLC	XLC
/36/	Rexroth IndraWorks 12VRS Software Installation DOK-IWORKS-SOFTINS*V12-COxx-EN-P, R911334396 This documentation describes the IndraWorks installation.	X	X
/5/	Rexroth IndraWorks 12VRS Engineering DOK-IWORKS-ENGINEE*V12-APxx-EN-P, R911334388 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 12VRS Functional Description DOK-MLC***-FUNC****V12-APxx-EN-P, R911333848 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraMotion MLC.	X	

About this Documentation

/20/	Rexroth IndraLogic XLC 12VRS Functional Description DOK-XLC***-FUNC***V12-APxx-EN-P, R911333878 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraLogic XLC.		X
/7/	Rexroth IndraWorks 12VRS CamBuilder DOK-IWORKS-CAMBUIL*V12-APxx-EN-P, R911333842 This documentation describes the basic principles and operation of the CamBuilder, the cam editing tool.	X	X
/37/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Automation Interface DOK-XLCMLC-AUT*INT*V12-APxx-EN-P, R911334178 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-APxx-EN-P, R911334398 This documentation describes the IndraWorks FDT Container functionality. It includes the activation of the functionality in the project and working with DTMs.	X	X
/29/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Project Conversion DOK-XLCMLC-PROCONV*V12-APxx-EN-P, R911334187 This documentation described the project conversion of IndraLogic 04VRS and IndraMotion MLC04VRS on IndraWorks Version 12 with IndraLogic 2G. It especially focuses on changes in the field of Motion and PLC.	X	X
/28/	Rexroth IndraMotion MLC 12VRS Commissioning DOK-MLC***-STARTUP*V12-COxx-EN-P, R911333858 This documentation describes the steps required for commissioning and performing service on the IndraMotion MLC system. It includes checklists for tasks to be frequently performed and a detailed description of the steps.	X	

Motion		MLC	XLC
/23/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS PLCopen Libraries DOK-XLCMLC-FUNLIB**V12-LIxx-EN-P, R911334182 This documentation describes the function blocks, functions and data types of the RIL_CommonTypes, ML_Base and ML_PLCOpen libraries for the IndraLogic XLC/IndraMotion MLC. It also includes the error reactions of function blocks.	X	X
/27/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Generic Application Template DOK-XLCMLC-TF*GAT**V12-APxx-EN-P, R911334191 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 12VRS RCL Programming Instruction DOK-MLC***-RCL*PRO*V12-APxx-EN-P, R911333852 This documentation provides information on the RobotControl. It is given most importance to the programming language RCL (RobotControl Language). The program structure, variables, functions, motion statements and the required system parameters are described.	X	
/21/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Parameters DOK-XLCMLC-PARAM***V12-RExx-EN-P, R911334176 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

/10/	Rexroth IndraDrive; Firmware for Drive Controllers MPH, MPB, MPD, MPC-07 DOK-INDRV*-MP*-07VRS**-FKxx-EN-P, R911328670		
/11/	Rexroth IndraDrive MPx-16 Functions DOK-INDRV*-MP*-16VRS**-APxx-EN-P, R911326767		

Field buses		MLC	XLC
/39/	Rexroth IndraMotion MLC 11VRS PLCopen Field Bus DOK-IM*ML*-PLCFBUS*V11-APxx-EN-P, R911333896 This documentation describes the creation of field bus drives in an IndraWorks project, function blocks, functions and data types of the libraries RIL_CommonTypes.library (excerpt for field bus drives), RMB_PLCPopenFieldBus.library, RIL_Utility.library (excerpt for field bus drives). It also includes the error reactions of function blocks.	X	X
/4/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Field Buses DOK-XLCMLC-FB*****V12-APxx-EN-P, R911334394 This documentation describes the supported field buses and their diagnostic function blocks.	X	X

HMI		MLC	XLC
/8/	Rexroth IndraWorks 12VRS HMI DOK-IWORKS-HMI*****V12-APxx-EN-P, R911334392 This documentation describes the functions, configuration and operation of the user interfaces IndraWorks HMI Engineering and IndraWorks HMI Operation.	X	X
/6/	Rexroth IndraWorks 12VRS WinStudio DOK-IWORKS-WINSTUD*V12-APxx-EN-P, R911333844 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 12VRS HMI Connection DOK-XLCMLC-HMI*****V12-APxx-EN-P, R911334184 This documentation describes the visualization systems supported by the IndraLogic XLC and IndraMotion MLC and their connection.	X	X

PLC		MLC	XLC
/3/	Rexroth IndraWorks 12VRS IndraLogic 2G Programming Instruction DOK-IWORKS-IL2GPRO*V12-APxx-EN-P, R911334390 This documentation describes the PLC programming tool IndraLogic 2G and its usage. It includes the basic usage, first steps, visualization, menu items and editors.	X	X
/33/	Rexroth IndraWorks 12VRS, Basic Libraries, IndraLogic 2G DOK-IL*2G*-BASLIB**V12-LIxx-EN-P, R911333835 This documentation describes the system-comprehensive PLC libraries.	X	X

Technology			
/30/	Rexroth IndraMotion MLC 12VRS Technology Libraries DOK-MLC***-TF*LIB**V12-LIxx-EN-P, R911333868 This documentation describes the function blocks, functions and data types of the libraries "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	

About this Documentation

/60/	Rexroth IndraMotion MLC 12VRS RegisterControl (Library) DOK-MLC***-REGI*CO*V12-LIxx-EN-P, R911333856 This documentation describes the inputs and outputs of the individual function blocks and provides notes on their usage.	X	
/62/	Rexroth IndraMotion MLC 12VRS RegisterControl (Application Manual) DOK-MLC***-REGI*CO*V12-APxx-EN-P, R911333854 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 on errors and query diagnostics. This documentation is written for machine setters and machine operators.	X	
/49/	Rexroth IndraMotion MLC 12VRS Application of Winder Functions DOK-MLC***-TF*WIND*V12-APxx-EN-P, R911333870 This application-related system documentation describes the usage of the winder technology functions.	X	

Hardware		MLC	XLC
/1/	Rexroth IndraControl L45/L65 DOK-CONTRL-IC*L45*L65*-PRxx-EN-P, R911324661	X	X
/2/	Rexroth IndraControl L25 DOK-CONTRL-IC*L25*****-PRxx-EN-P, R911328474	X	X
/24/	Rexroth IndraControl Lxx 12VRS Function Modules DOK-CONTRL-FM*LXX**V12-APxx-EN-P, R911333830 This documentation describes all function modules of the Lxx controls including engineering and diagnostics.	X	X
/12/	Rexroth IndraDrive Drive Controllers MPx-02 to MPx-07 DOK-INDRV*-GEN-**VRS**-PAxx-EN-P, R911297317		
/13/	Rexroth IndraDrive MPx-02 to MPx-07 and HMV DOK-INDRV*-GEN-**VRS**-WAxx-EN-P, R911297319		
/35/	Rexroth IndraDrive Drive Controller Control Sections CSB01, CSH01, CDB01 DOK-INDRV*-CSH*****-PR08-EN-P, R911295012		

Diagnostics and Service		MLC	XLC
/26/	Rexroth IndraMotion MLC/XLC 11VRS Service Tool DOK-IM*ML*-IMST***V11-RExx-EN-P, R911331940	X	X
/22/	Rexroth IndraLogic XLC IndraMotion MLC 12VRS Diagnostics DOK-XLCMLC-DIAG***V12-RExx-EN-P, R911334180 This documentation includes all control parameters implemented in the control systems IndraLogic XLC and IndraMotion MLC.	X	X

System Overview		MLC	XLC
/48/	Rexroth IndraMotion for Printing 12VRS System Overview DOK-IM*PR*-SYSTEM**V11-PRxx-EN-P, R911333840 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

/48/	Rexroth IndraMotion for Packaging 12VRS System Overview DOK-IM*PA*-SYSTEM**V12-PRxx-EN-P, R911333838 This documentation describes the product IndraMotion for Packaging. It introduces the control systems, drive systems and I/O systems as well as the commissioning and programming.	X	
/9/	Rexroth IndraMotion MLC 12VRS System Overview DOK-MLC***-SYSTEM**V12-PRxx-EN-P, R911333860 This documentation provides an overview on the possible hardware/software components of the automation system IndraMotion MLC of the named version. It helps assembling a system.	X	
/9/	Rexroth IndraLogic XLC 12VRS System Overview DOK-XLC***-SYSTEM**V12-PRxx-EN-P, R911333880 This documentation provides an overview on the possible hardware/software components of the automation system IndraLogic XLC of the named version. It helps assembling a system.		X

First Steps		MLC	XLC
/25/	Rexroth IndraMotion MLC 12VRS First Steps DOK-MLC***-F*STEP**V12-COxx-EN-P, R911333846 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 12VRS First Steps DOK-XLC***-F*STEP**V12-COxx-EN-P, R911333876 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

xx Respective edition
Fig. 1-1: XCL/MLC documentation overview

2 Important Instructions on Use

2.1 Appropriate Use

2.1.1 Introduction

Rexroth products represent state-of-the-art developments and manufacturing. They are tested prior to delivery to ensure operating safety and reliability.

The products may only be used in the appropriate way. If they are not used in the appropriate way, situations resulting in property damage and personal injury can occur.



Bosch Rexroth as manufacturer is not liable for any damages resulting from inappropriate use. In such cases, the guarantee and the right to payment of damages resulting from inappropriate use are forfeited. The user alone carries all responsibility of the risks.

Before using Rexroth products, make sure that all the pre-requisites for an appropriate use of the products are satisfied:

- Personnel that in any way, shape or form uses our products must first read and understand the relevant safety instructions and be familiar with appropriate use.
- If the products take the form of hardware, then they must remain in their original state, in other words, no structural changes are permitted. It is not permitted to decompile software products or alter source codes.
- Do not mount damaged or faulty products or use them in operation.
- Ensure that the products have been installed in the manner described in the relevant documentation.

2.1.2 Areas of Use and Application

Rexroth IndraMotion MLC

IndraMotion MLC and its function modules are intended for Motion/Logic applications.



The IndraMotion MLC and its function modules may only be used with the accessories and mounting parts listed in this documentation. Components that are not expressly mentioned must neither be attached nor connected. The same is valid for cables and lines.

The operation must only be carried out with the hardware component configurations and combinations that were expressly mentioned and with the software and firmware indicated and specified in the respective documentations and functional descriptions.

IndraMotion MLC and its function modules have been developed for use in single and multi-axis drive and control tasks.

For the application-specific use of the machine operating and visualization terminals, device models featuring different equipments and different interfaces are available.

Typical scopes of use where IndraMotion MLC and its function modules can be applied are:

- [Handling and mounting systems]
- [Packaging and food machines]
- [Printing and paper-processing machines]

Important Instructions on Use

Rexroth IndraMotion MLC and the function modules must only be operated under the mounting and installation conditions, the position, and the ambient conditions (temperature, type of protection, moisture, EMC, etc.) specified in the related documentations.

2.2 Inappropriate Use

The use of Rexroth IndraMotion MLC and its function modules in applications other than those specified or described in the documentation and the technical data is considered as "improper".

The Rexroth IndraMotion MLC and its function modules may not be used if it is

- exposed to operating conditions which do not correspond to the specified ambient conditions. Operation under water, extreme temperature fluctuations or extreme maximum temperatures etc. is prohibited.
- Furthermore, the IndraMotion MLC and its function modules must not be used in any applications not expressly approved by Bosch Rexroth. Therefore, read the statements given in the general safety instructions!

3 Safety Instructions for Electric Drives and Controls

3.1 Definitions of Terms

Application Documentation	Application documentation comprises the entire documentation used to inform the user of the product about the use and safety-relevant features for configuring, integrating, installing, mounting, commissioning, operating, maintaining, repairing and decommissioning the product. The following terms are also used for this kind of documentation: User Guide, Operation Manual, Commissioning Manual, Instruction Manual, Project Planning Manual, Application Manual, etc.
Component	A component is a combination of elements with a specified function, which are part of a piece of equipment, device or system. Components of the electric drive and control system are, for example, supply units, drive controllers, mains choke, mains filter, motors, cables, etc.
Control System	A control system comprises several interconnected control components placed on the market as a single functional unit.
Device	A device is a finished product with a defined function, intended for users and placed on the market as an individual piece of merchandise.
Electrical Equipment	Electrical equipment encompasses all devices used to generate, convert, transmit, distribute or apply electrical energy, such as electric motors, transformers, switching devices, cables, lines, power-consuming devices, circuit board assemblies, plug-in units, control cabinets, etc.
Electric Drive System	An electric drive system comprises all components from mains supply to motor shaft; this includes, for example, electric motor(s), motor encoder(s), supply units and drive controllers, as well as auxiliary and additional components, such as mains filter, mains choke and the corresponding lines and cables.
Installation	An installation consists of several devices or systems interconnected for a defined purpose and on a defined site which, however, are not intended to be placed on the market as a single functional unit.
Machine	A machine is the entirety of interconnected parts or units at least one of which is movable. Thus, a machine consists of the appropriate machine drive elements, as well as control and power circuits, which have been assembled for a specific application. A machine is, for example, intended for processing, treatment, movement or packaging of a material. The term "machine" also covers a combination of machines which are arranged and controlled in such a way that they function as a unified whole.
Manufacturer	The manufacturer is an individual or legal entity bearing responsibility for the design and manufacture of a product which is placed on the market in the individual's or legal entity's name. The manufacturer can use finished products, finished parts or finished elements, or contract out work to subcontractors. However, the manufacturer must always have overall control and possess the required authority to take responsibility for the product.
Product	Examples of a product: Device, component, part, system, software, firmware, among other things.
Project Planning Manual	A project planning manual is part of the application documentation used to support the sizing and planning of systems, machines or installations.
Qualified Persons	In terms of this application documentation, qualified persons are those persons who are familiar with the installation, mounting, commissioning and operation of the components of the electric drive and control system, as well as with the hazards this implies, and who possess the qualifications their work requires. To comply with these qualifications, it is necessary, among other things,

Safety Instructions for Electric Drives and Controls

- 1) to be trained, instructed or authorized to switch electric circuits and devices safely on and off, to ground them and to mark them
- 2) to be trained or instructed to maintain and use adequate safety equipment
- 3) to attend a course of instruction in first aid

User A user is a person installing, commissioning or using a product which has been placed on the market.

3.2 General Information

3.2.1 Using the Safety Instructions and Passing Them on to Others

Do not attempt to install and operate the components of the electric drive and control system without first reading all documentation provided with the product. Read and understand these safety instructions and all user documentation prior to working with these components. If you do not have the user documentation for the components, contact your responsible Bosch Rexroth sales partner. Ask for these documents to be sent immediately to the person or persons responsible for the safe operation of the components.

If the component is resold, rented and/or passed on to others in any other form, these safety instructions must be delivered with the component in the official language of the user's country.

Improper use of these components, failure to follow the safety instructions in this document or tampering with the product, including disabling of safety devices, could result in property damage, injury, electric shock or even death.

3.2.2 Requirements for Safe Use

Read the following instructions before initial commissioning of the components of the electric drive and control system in order to eliminate the risk of injury and/or property damage. You must follow these safety instructions.

- Bosch Rexroth is not liable for damages resulting from failure to observe the safety instructions.
- Read the operating, maintenance and safety instructions in your language before commissioning. If you find that you cannot completely understand the application documentation in the available language, please ask your supplier to clarify.
- Proper and correct transport, storage, mounting and installation, as well as care in operation and maintenance, are prerequisites for optimal and safe operation of the component.
- Only qualified persons may work with components of the electric drive and control system or within its proximity.
- Only use accessories and spare parts approved by Bosch Rexroth.
- Follow the safety regulations and requirements of the country in which the components of the electric drive and control system are operated.
- Only use the components of the electric drive and control system in the manner that is defined as appropriate. See chapter "Appropriate Use".
- The ambient and operating conditions given in the available application documentation must be observed.
- Applications for functional safety are only allowed if clearly and explicitly specified in the application documentation "Integrated Safety Technology". If this is not the case, they are excluded. Functional safety is a safety

Safety Instructions for Electric Drives and Controls

concept in which measures of risk reduction for personal safety depend on electrical, electronic or programmable control systems.

- The information given in the application documentation with regard to the use of the delivered components contains only examples of applications and suggestions.

The machine and installation manufacturers must

- make sure that the delivered components are suited for their individual application and check the information given in this application documentation with regard to the use of the components,
- make sure that their individual application complies with the applicable safety regulations and standards and carry out the required measures, modifications and complements.
- Commissioning of the delivered components is only allowed once it is sure that the machine or installation in which the components are installed complies with the national regulations, safety specifications and standards of the application.
- Operation is only allowed if the national EMC regulations for the application are met.
- The instructions for installation in accordance with EMC requirements can be found in the section on EMC in the respective application documentation.

The machine or installation manufacturer is responsible for compliance with the limit values as prescribed in the national regulations.

- The technical data, connection and installation conditions of the components are specified in the respective application documentations and must be followed at all times.

National regulations which the user must take into account

- European countries: In accordance with European EN standards
- United States of America (USA):
 - National Electrical Code (NEC)
 - National Electrical Manufacturers Association (NEMA), as well as local engineering regulations
 - Regulations of the National Fire Protection Association (NFPA)
- Canada: Canadian Standards Association (CSA)
- Other countries:
 - International Organization for Standardization (ISO)
 - International Electrotechnical Commission (IEC)

3.2.3 Hazards by Improper Use

- High electrical voltage and high working current! Danger to life or serious injury by electric shock!
- High electrical voltage by incorrect connection! Danger to life or injury by electric shock!
- Dangerous movements! Danger to life, serious injury or property damage by unintended motor movements!
- Health hazard for persons with heart pacemakers, metal implants and hearing aids in proximity to electric drive systems!
- Risk of burns by hot housing surfaces!

Safety Instructions for Electric Drives and Controls

- Risk of injury by improper handling! Injury by crushing, shearing, cutting, hitting!
- Risk of injury by improper handling of batteries!
- Risk of injury by improper handling of pressurized lines!

3.3 Instructions with Regard to Specific Dangers

3.3.1 Protection Against Contact With Electrical Parts and Housings



This section concerns components of the electric drive and control system with voltages of **more than 50 volts**.

Contact with parts conducting voltages above 50 volts can cause personal danger and electric shock. When operating components of the electric drive and control system, it is unavoidable that some parts of these components conduct dangerous voltage.

High electrical voltage! Danger to life, risk of injury by electric shock or serious injury!

- Only qualified persons are allowed to operate, maintain and/or repair the components of the electric drive and control system.
- Follow the general installation and safety regulations when working on power installations.
- Before switching on, the equipment grounding conductor must have been permanently connected to all electric components in accordance with the connection diagram.
- Even for brief measurements or tests, operation is only allowed if the equipment grounding conductor has been permanently connected to the points of the components provided for this purpose.
- Before accessing electrical parts with voltage potentials higher than 50 V, you must disconnect electric components from the mains or from the power supply unit. Secure the electric component from reconnection.
- With electric components, observe the following aspects:
Always wait **30 minutes** after switching off power to allow live capacitors to discharge before accessing an electric component. Measure the electrical voltage of live parts before beginning to work to make sure that the equipment is safe to touch.
- Install the covers and guards provided for this purpose before switching on.
- Never touch electrical connection points of the components while power is turned on.
- Do not remove or plug in connectors when the component has been powered.
- Under specific conditions, electric drive systems can be operated at mains protected by residual-current-operated circuit-breakers sensitive to universal current (RCDs/RCMs).
- Secure built-in devices from penetrating foreign objects and water, as well as from direct contact, by providing an external housing, for example a control cabinet.

High housing voltage and high leakage current! Danger to life, risk of injury by electric shock!

- Before switching on and before commissioning, ground or connect the components of the electric drive and control system to the equipment grounding conductor at the grounding points.

Safety Instructions for Electric Drives and Controls

- Connect the equipment grounding conductor of the components of the electric drive and control system permanently to the main power supply at all times. The leakage current is greater than 3.5 mA.
- Establish an equipment grounding connection with a minimum cross section according to the table below. With an outer conductor cross section smaller than 10 mm² (8 AWG), the alternative connection of two equipment grounding conductors is allowed, each having the same cross section as the outer conductors.

Cross section outer conductor	Minimum cross section equipment grounding conductor Leakage current ≥ 3.5 mA	
	1 equipment grounding conductor	2 equipment grounding conductors
1.5 mm ² (16 AWG)	10 mm ² (8 AWG)	2 × 1.5 mm ² (16 AWG)
2.5 mm ² (14 AWG)		2 × 2.5 mm ² (14 AWG)
4 mm ² (12 AWG)		2 × 4 mm ² (12 AWG)
6 mm ² (10 AWG)		2 × 6 mm ² (10 AWG)
10 mm ² (8 AWG)		-
16 mm ² (6 AWG)	16 mm ² (6 AWG)	-
25 mm ² (4 AWG)		-
35 mm ² (2 AWG)		-
50 mm ² (1/0 AWG)	25 mm ² (4 AWG)	-
70 mm ² (2/0 AWG)	35 mm ² (2 AWG)	-
...	...	...

Fig. 3-1: Minimum Cross Section of the Equipment Grounding Connection

3.3.2 Protective Extra-Low Voltage as Protection Against Electric Shock

Protective extra-low voltage is used to allow connecting devices with basic insulation to extra-low voltage circuits.

On components of an electric drive and control system provided by Bosch Rexroth, all connections and terminals with voltages between 5 and 50 volts are PELV ("Protective Extra-Low Voltage") systems. It is allowed to connect devices equipped with basic insulation (such as programming devices, PCs, notebooks, display units) to these connections.

Danger to life, risk of injury by electric shock! High electrical voltage by incorrect connection!

If extra-low voltage circuits of devices containing voltages and circuits of more than 50 volts (e.g., the mains connection) are connected to Bosch Rexroth products, the connected extra-low voltage circuits must comply with the requirements for PELV ("Protective Extra-Low Voltage").

3.3.3 Protection Against Dangerous Movements

Dangerous movements can be caused by faulty control of connected motors. Some common examples are:

Safety Instructions for Electric Drives and Controls

- Improper or wrong wiring or cable connection
- Operator errors
- Wrong input of parameters before commissioning
- Malfunction of sensors and encoders
- Defective components
- Software or firmware errors

These errors can occur immediately after equipment is switched on or even after an unspecified time of trouble-free operation.

The monitoring functions in the components of the electric drive and control system will normally be sufficient to avoid malfunction in the connected drives. Regarding personal safety, especially the danger of injury and/or property damage, this alone cannot be relied upon to ensure complete safety. Until the integrated monitoring functions become effective, it must be assumed in any case that faulty drive movements will occur. The extent of faulty drive movements depends upon the type of control and the state of operation.

Dangerous movements! Danger to life, risk of injury, serious injury or property damage!

A **risk assessment** must be prepared for the installation or machine, with its specific conditions, in which the components of the electric drive and control system are installed.

As a result of the risk assessment, the user must provide for monitoring functions and higher-level measures on the installation side for personal safety. The safety regulations applicable to the installation or machine must be taken into consideration. Unintended machine movements or other malfunctions are possible if safety devices are disabled, bypassed or not activated.

To avoid accidents, injury and/or property damage:

- Keep free and clear of the machine's range of motion and moving machine parts. Prevent personnel from accidentally entering the machine's range of motion by using, for example:
 - Safety fences
 - Safety guards
 - Protective coverings
 - Light barriers
- Make sure the safety fences and protective coverings are strong enough to resist maximum possible kinetic energy.
- Mount emergency stopping switches in the immediate reach of the operator. Before commissioning, verify that the emergency stopping equipment works. Do not operate the machine if the emergency stopping switch is not working.
- Prevent unintended start-up. Isolate the drive power connection by means of OFF switches/OFF buttons or use a safe starting lockout.
- Make sure that the drives are brought to safe standstill before accessing or entering the danger zone.
- Additionally secure vertical axes against falling or dropping after switching off the motor power by, for example,
 - mechanically securing the vertical axes,
 - adding an external braking/arrester/clamping mechanism or
 - ensuring sufficient counterbalancing of the vertical axes.

Safety Instructions for Electric Drives and Controls

- The standard equipment **motor holding brake** or an external holding brake controlled by the drive controller is **not sufficient to guarantee personal safety!**
- Disconnect electrical power to the components of the electric drive and control system using the master switch and secure them from reconnection ("lock out") for:
 - Maintenance and repair work
 - Cleaning of equipment
 - Long periods of discontinued equipment use
- Prevent the operation of high-frequency, remote control and radio equipment near components of the electric drive and control system and their supply leads. If the use of these devices cannot be avoided, check the machine or installation, at initial commissioning of the electric drive and control system, for possible malfunctions when operating such high-frequency, remote control and radio equipment in its possible positions of normal use. It might possibly be necessary to perform a special electromagnetic compatibility (EMC) test.

3.3.4 Protection Against Magnetic and Electromagnetic Fields During Operation and Mounting

Magnetic and electromagnetic fields generated by current-carrying conductors or permanent magnets of electric motors represent a serious danger to persons with heart pacemakers, metal implants and hearing aids.

Health hazard for persons with heart pacemakers, metal implants and hearing aids in proximity to electric components!

- Persons with heart pacemakers and metal implants are not allowed to enter the following areas:
 - Areas in which components of the electric drive and control systems are mounted, commissioned and operated.
 - Areas in which parts of motors with permanent magnets are stored, repaired or mounted.
- If it is necessary for somebody with a heart pacemaker to enter such an area, a doctor must be consulted prior to doing so. The noise immunity of implanted heart pacemakers differs so greatly that no general rules can be given.
- Those with metal implants or metal pieces, as well as with hearing aids, must consult a doctor before they enter the areas described above.

3.3.5 Protection Against Contact With Hot Parts

Hot surfaces of components of the electric drive and control system. Risk of burns!

- Do not touch hot surfaces of, for example, braking resistors, heat sinks, supply units and drive controllers, motors, windings and laminated cores!
- According to the operating conditions, temperatures of the surfaces can be **higher than 60 °C (140 °F)** during or after operation.
- Before touching motors after having switched them off, let them cool down for a sufficient period of time. Cooling down can require **up to 140 minutes!** The time required for cooling down is approximately five times the thermal time constant specified in the technical data.

Safety Instructions for Electric Drives and Controls

- After switching chokes, supply units and drive controllers off, wait **15 minutes** to allow them to cool down before touching them.
- Wear safety gloves or do not work at hot surfaces.
- For certain applications, and in accordance with the respective safety regulations, the manufacturer of the machine or installation must take measures to avoid injuries caused by burns in the final application. These measures can be, for example: Warnings at the machine or installation, guards (shieldings or barriers) or safety instructions in the application documentation.

3.3.6 Protection During Handling and Mounting

Risk of injury by improper handling! Injury by crushing, shearing, cutting, hitting!

- Observe the relevant statutory regulations of accident prevention.
- Use suitable equipment for mounting and transport.
- Avoid jamming and crushing by appropriate measures.
- Always use suitable tools. Use special tools if specified.
- Use lifting equipment and tools in the correct manner.
- Use suitable protective equipment (hard hat, safety goggles, safety shoes, safety gloves, for example).
- Do not stand under hanging loads.
- Immediately clean up any spilled liquids from the floor due to the risk of falling!

3.3.7 Battery Safety

Batteries consist of active chemicals in a solid housing. Therefore, improper handling can cause injury or property damage.

Risk of injury by improper handling!

- Do not attempt to reactivate low batteries by heating or other methods (risk of explosion and cauterization).
- Do not attempt to recharge the batteries as this may cause leakage or explosion.
- Do not throw batteries into open flames.
- Do not dismantle batteries.
- When replacing the battery/batteries, do not damage the electrical parts installed in the devices.
- Only use the battery types specified for the product.



Environmental protection and disposal! The batteries contained in the product are considered dangerous goods during land, air, and sea transport (risk of explosion) in the sense of the legal regulations. Dispose of used batteries separately from other waste. Observe the national regulations of your country.

Safety Instructions for Electric Drives and Controls

3.3.8 Protection Against Pressurized Systems

According to the information given in the Project Planning Manuals, motors and components cooled with liquids and compressed air can be partially supplied with externally fed, pressurized media, such as compressed air, hydraulics oil, cooling liquids and cooling lubricants. Improper handling of the connected supply systems, supply lines or connections can cause injuries or property damage.

Risk of injury by improper handling of pressurized lines!

- Do not attempt to disconnect, open or cut pressurized lines (risk of explosion).
- Observe the respective manufacturer's operating instructions.
- Before dismantling lines, relieve pressure and empty medium.
- Use suitable protective equipment (safety goggles, safety shoes, safety gloves, for example).
- Immediately clean up any spilled liquids from the floor due to the risk of falling!



Environmental protection and disposal! The agents (e.g., fluids) used to operate the product might not be environmentally friendly. Dispose of agents harmful to the environment separately from other waste. Observe the national regulations of your country.

3.4 Explanation of Signal Words and the Safety Alert Symbol

The Safety Instructions in the available application documentation contain specific signal words (DANGER, WARNING, CAUTION or NOTICE) and, where required, a safety alert symbol (in accordance with ANSI Z535.6-2006).

The signal word is meant to draw the reader's attention to the safety instruction and identifies the hazard severity.

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 **could** occur.

⚠ CAUTION

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

Safety Instructions for Electric Drives and Controls

NOTICE

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

4 First Steps - IndraLogic

4.1 Project Goal "First steps - IndraLogic"

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

A simple commissioning visualization completes the project.

This section describes a few typical operating sequences in IndraWorks. If you proceed according to these descriptions, you will become familiar with how IndraWorks functions and you will recognize how the individual components interact.

Project restoration:

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

To do so, use the export file "Application_FirstSteps_IndraLogic_Basics.iwx" filed in your installation directory under "Iw:\All Programs\Rexroth\IndraWorks\Projects". More detailed information can be found under [Restoring a Project, on page 81](#).



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

- | | |
|-----------------|---|
| Subtasks | <ul style="list-style-type: none">• Create a project with a control,• Edit the PLC program,• Edit PLC visualization,• Simulate the PLC program,• Load the PLC program onto the control. |
|-----------------|---|

4.2 Technical Requirements

- PC with installed IndraWorks Engineering for IndraMotion MLC, 12VRS, control, type "IndraMotion MLC L25", "IndraMotion MLC L45" or "IndraMotion MLC L65",
current firmware is loaded.

Meeting these prerequisites is described in

- Rexroth IndraMotion MLC 12VRS Functional Description, DOK-IM*ML*-SYSTEM**V12-RExx-EN-P and
- IndraWorks online help, **System IndraLogic XLC/IndraMotion MLC ▶ XLC/MLC Functional Description ▶ Commissioning and Operation**,

Control, e.g. type "IndraLogic XLC L65.1", the current firmware is loaded.

- the IP address is known,
- the control is connected to the programming device (PC) via Ethernet.
- Should a SERCOS III bus be present, this must be bridged.

4.3 Creating a New IndraWorks Project

4.3.1 Resetting the Control (CMLx5)

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

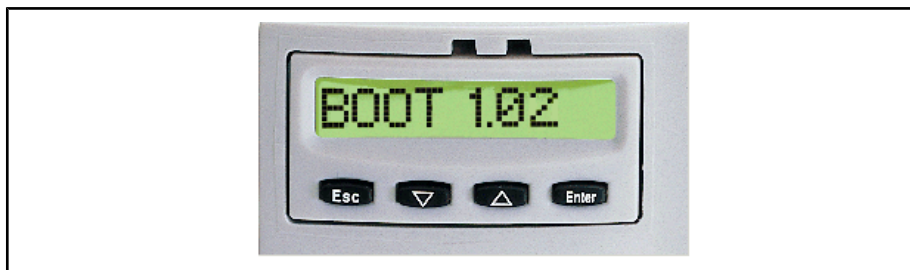


Fig.4-1: Rebooting the control

After the control is switched on, press the outer buttons <ESC> and <Enter> located below the display and when **BOOT 1.02** appears and keep them pressed until **BOOTSTOP** appears. Press the two inner buttons. First **CLEANUP** and then **REBOOT** appears.

The control now runs until **BB RDY / RDY** appears, the memory is empty, any Motion component installed and the PLC component are ready for operation.

4.3.2 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.



Desktop icon:

4.3.3 Creating a New Project

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

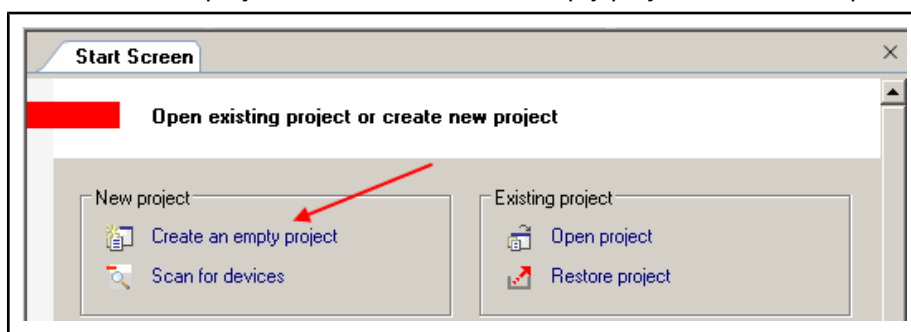


Fig.4-2: Creating an empty project

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

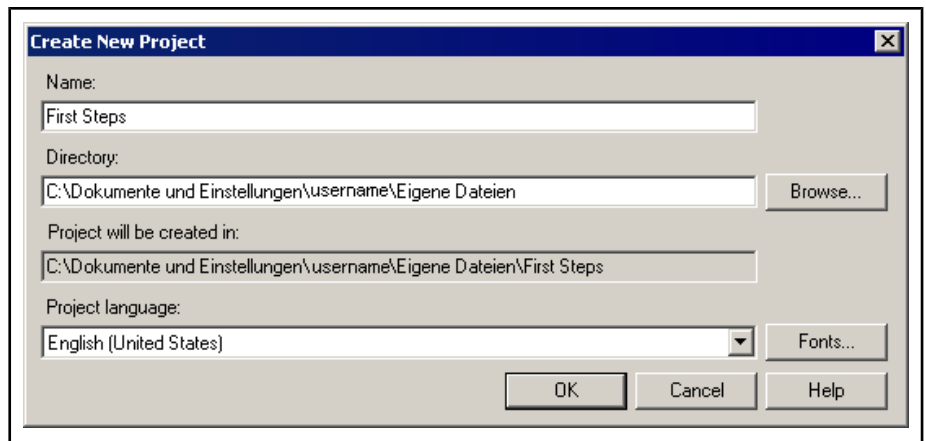


Fig. 4-3: Settings to save a new IndraWorks Project

Name	Specify a name for the project. Suggestion: "First steps".
Directory	The field contains a storage location in the directory C:\Documents and Settings \username\MyDocuments by default. A different directory can also be specified.
Project is created in	Directory and project name are automatically created.
Project language	By selecting the project language, the language of the user texts is determined. The newly created project appears in Project Explorer with the previously defined project name, refer to the following figure.

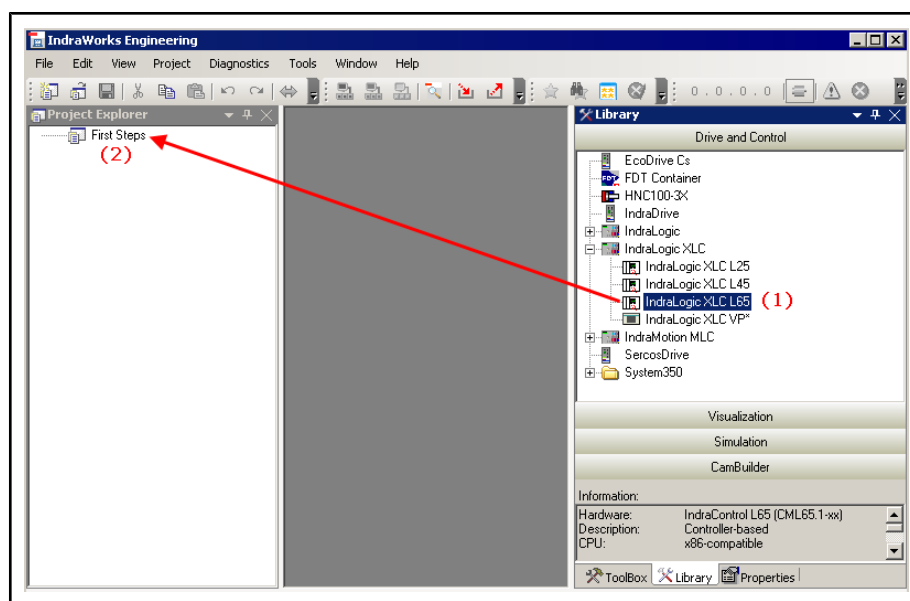


Fig. 4-4: First steps project

4.4 Adding a New Device to a Project

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

First Steps - IndraLogic



(1) Device: IndraLogic XLC L65

(2) Project: First Steps

Fig.4-5: Adding a device

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

4.5 Allocating Basic Settings to the Device

4.5.1 Device Settings

Enter the general information on the device to be added.

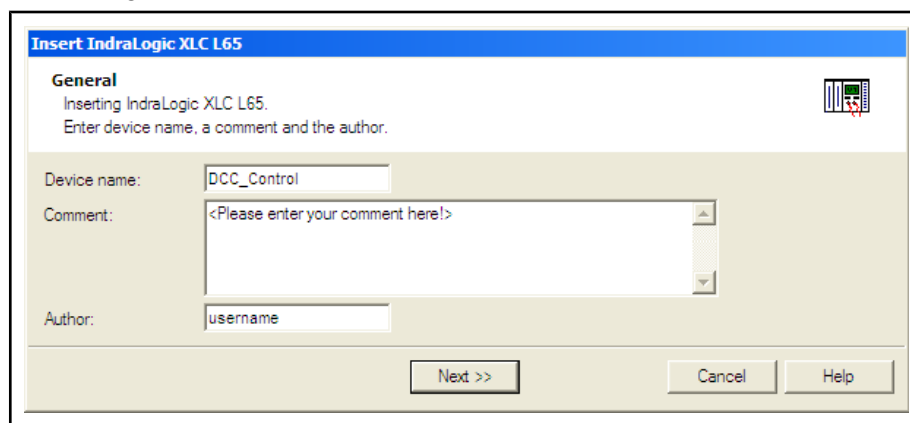


Fig.4-6: Device settings

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

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

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



These entries can also be changed later on.

Confirm the entries with "Next>>".

4.5.2 Communication Settings

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

Fig.4-7: Configuration

Device type

The device type is predetermined by selecting the control. If there are several different configurations for one device type, a manual supplement is necessary (for instance: IndraLogic XLC Lx5 is available with SERCOS III bus (CMLx5.1-3P) and without SERCOS III bus (CMLx5.1-NP)).

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 57. There is no need for a connection test.

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. The default computer name (address) here is "localhost". This means that the locally installed gateway is addressed. This setting is not to be changed.

Connection test

Click on the "Connection text" button to check whether the communication between computer and target control is successful. The test result is recorded in the "Connection test result" 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 "Connection Test Result" window.

First Steps - IndraLogic

The following problems can occur:

- Output: "Communication to the control not successful."
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 change the IP address of the control (see Rexroth IndraMotion MLC Functional Description, Commissioning the Control).
 - The network mask of the PC and/or control is set too small.
Solution: Check and correct network settings (see Rexroth IndraMotion MLC Functional Description, Commissioning the Control).
- Output: "The configured firmware and the firmware in the control do not match."
Causes:
 - The control firmware is not current.
Solution: Update firmware (see Rexroth IndraMotion MLC Functional Description, Commissioning the Control).
 - The selected control does not correspond to the real control type.

Secure online mode If this option is checked, a prompt appears 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.

4.5.3 Function Module Settings



Depending on the device type, the "Device and function module settings" dialog may differ. In this project, the dialog of the "IndraLogic XLC L65" device is described as example.

Enter the desired device and function module settings.

First Steps - IndraLogic

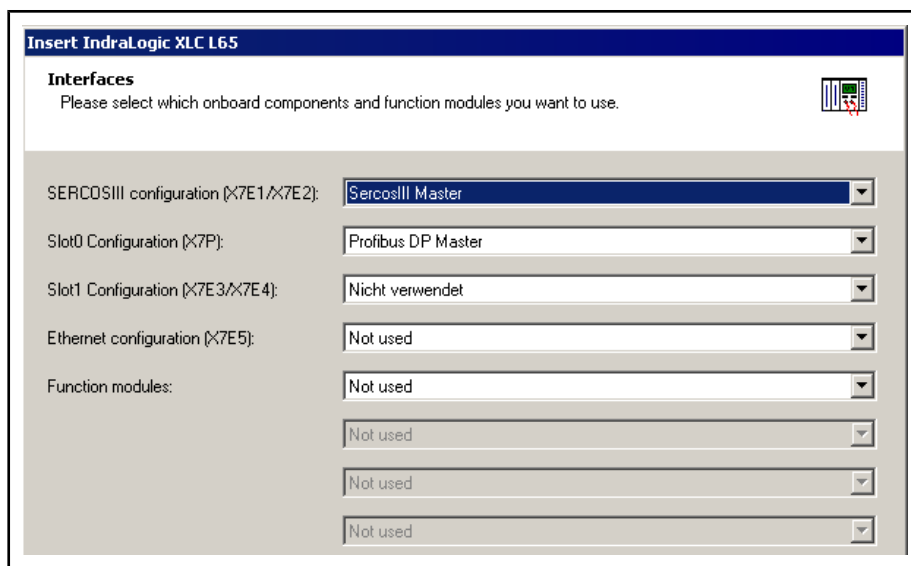


Fig.4-8: Function module settings

- SERCOS III configuration** Current SERCOS III configuration that can be selected when the device is added. The selection includes "Sercos III Master" or "Not used".
- Slot0 configuration** Current PROFIBUS configuration that can be selected when adding a device. "PROFIBUS DP Master", "PROFIBUS DP Slave" or "Not Used" can be selected.
- Slot1 configuration** 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** As this interface is required as Engineering interface (connection directly to your PC or via the operating network), the Ethernet/IP Adapter may not be enabled.
- Function modules 1-4** Up to four function modules can be connected to the "IndraLogic XLC L65". It is possible to select from the respective list if required.
- Confirm the entries with "Finish".
- The newly created device appears in the Project Explorer.

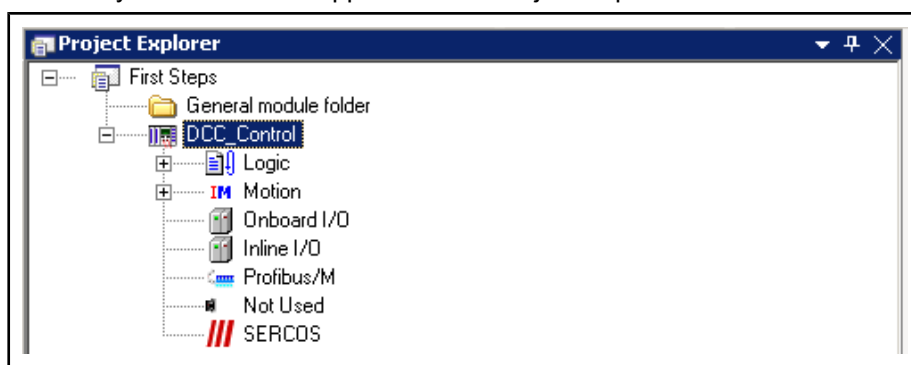


Fig.4-9: Project Explorer: "First Steps" project with an IndraLogic XLC L65 control with SERCOS III bus



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

4.6 Creating a PLC Program

4.6.1 The Control of a Traffic Light System

The following describes the example program "Mini traffic signal". The program is supposed to control two traffic lights at an intersection. Both traffic lights 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 you will see the following:

1. How time-dependent programs are displayed with the IEC 61131-3 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.

4.6.2 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 an L65.

"MlcProg" is not needed in our project but does not interfere with it.



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 34](#).
- When creating in "Continuous Function Chart (CFC)", continue below.

Proceed as follows:

1. In the Project Explorer, highlight the "PlcProg" POU.
2. Click the right mouse button to select the menu item **Delete** from the context menu.



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, you can use the command **Add ► POU** in the application node context menu.

First Steps - IndraLogic

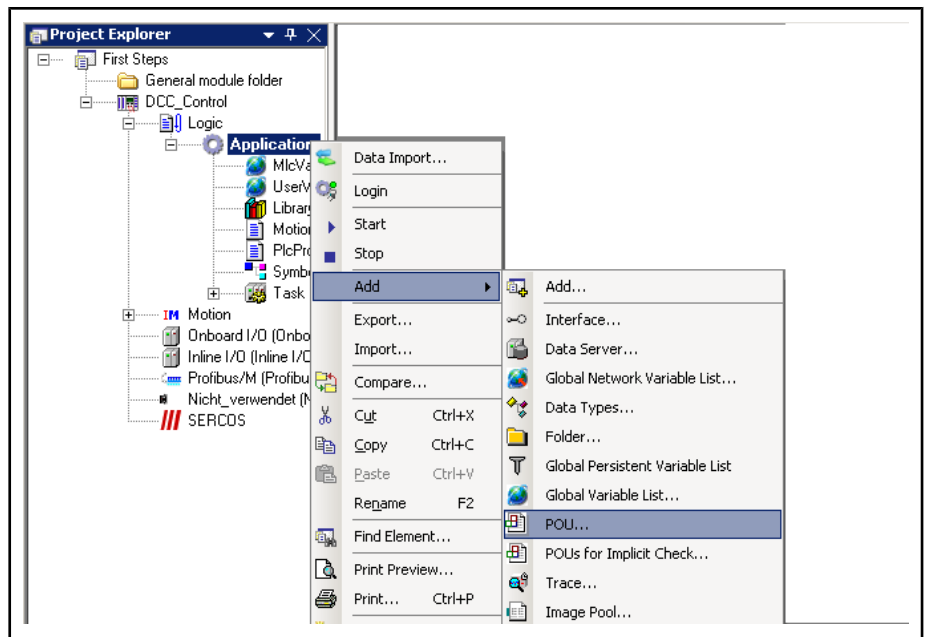


Fig.4-10: Adding a POU from the library

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. In the selection area under "Type:" select "Program" and as "Implementation language:" (programming language), select "Continuous function chart (CFC)".

Confirm using "Finish".

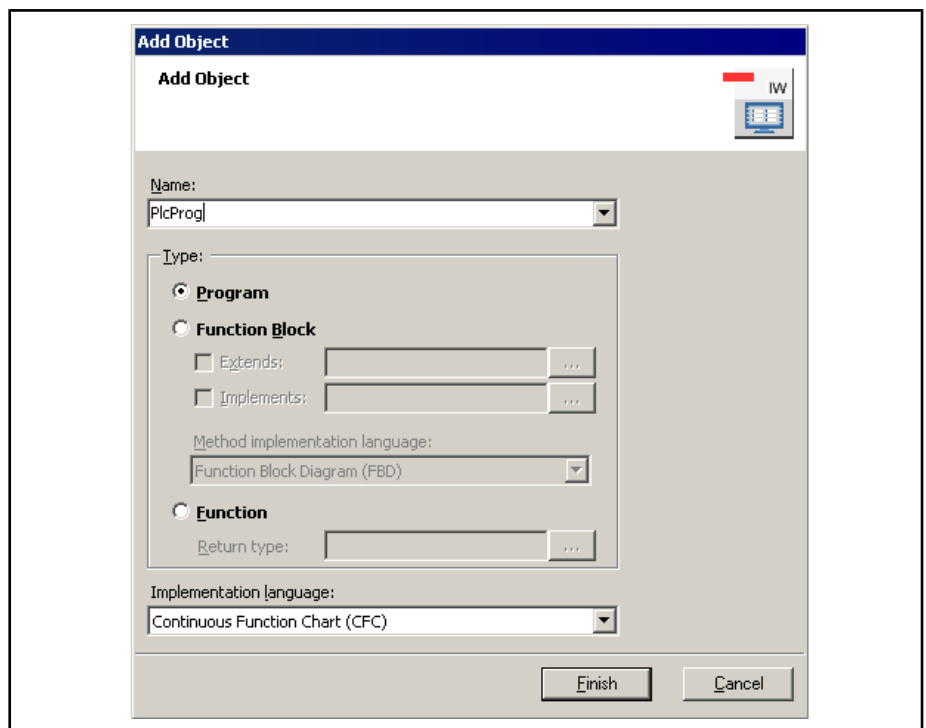


Fig.4-11: Configuring the "PlcProg" POU

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

First Steps - IndraLogic

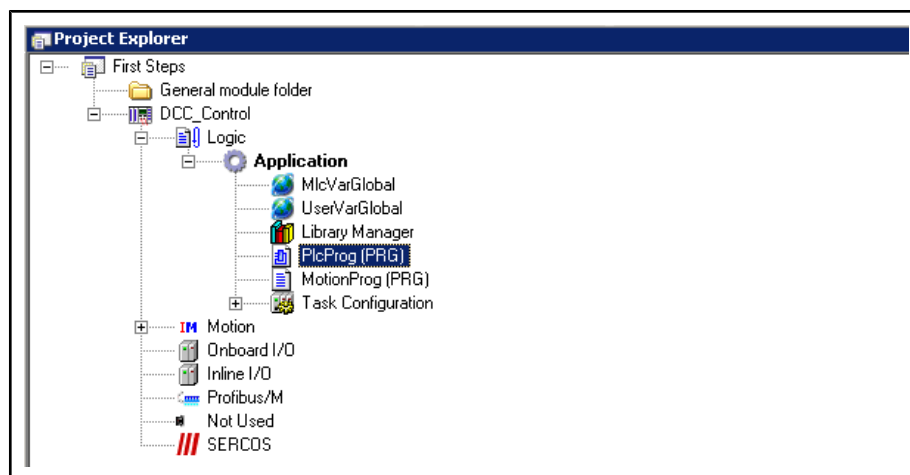


Fig.4-12: "PlcProg" POU has been added to the application

Function of "PlcProg"

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 actuated.

Adding "SEQUENCE" Program in "AS"

Highlight "Application" and click the right mouse button. 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. In the selection area under "Type:" select "Program" and as "Implementation language:" (programming language) "Sequential function chart (SFC)".
3. Confirm using "Finish".

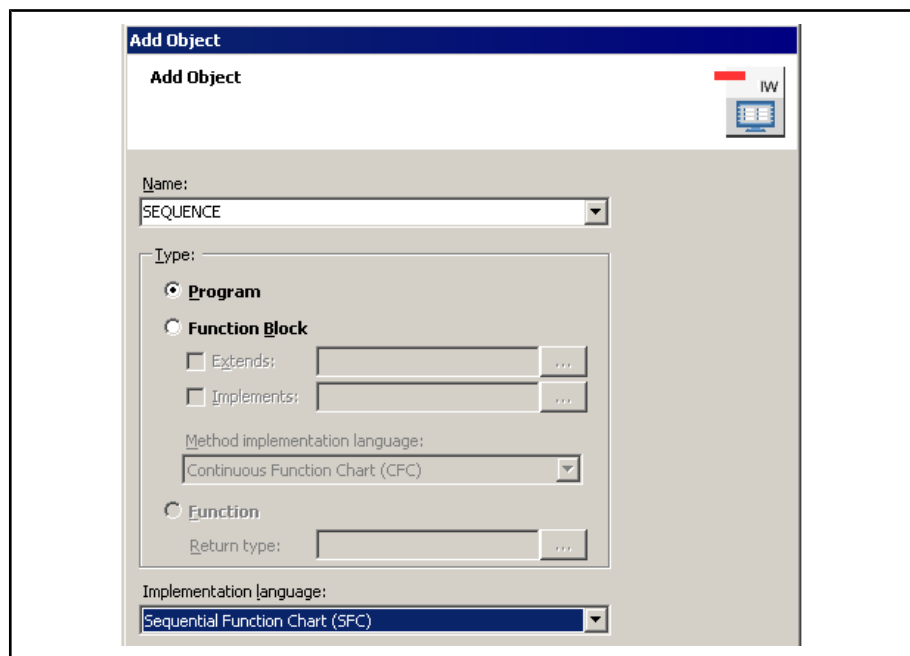


Fig.4-13: Adding "SEQUENCE" program

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

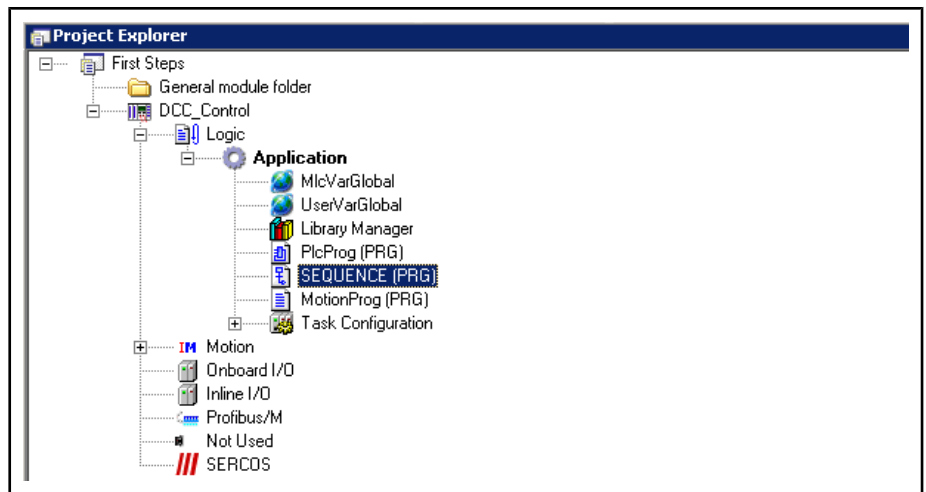


Fig. 4-14: "SEQUENCE" as part of the application

Function of "SEQUENCE"

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 "Application" and click the right mouse button. 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. In the selection area under "Type:" select under "Function Block" and as the "Implementation language:" (programming language) "Function block diagram (FBD)".
3. Confirm using "Finish".

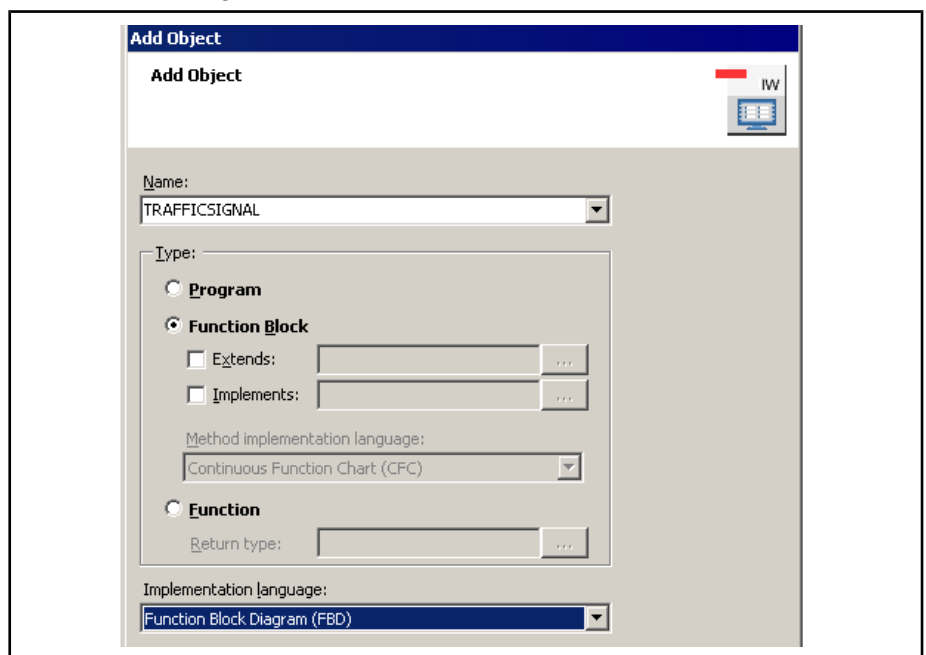


Fig. 4-15: Adding "TRAFFICSIGNAL" function block

First Steps - IndraLogic

The "TRAFFICSIGNAL" POU, of the "function block" type, now appears in the project tree below the application.

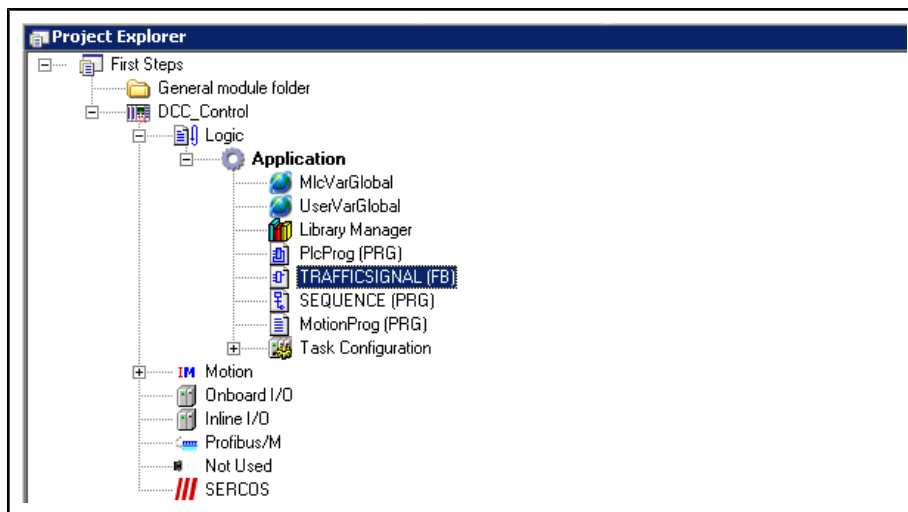


Fig.4-16: "TRAFFICSIGNAL" function block as part of the application

Function of "TRAFFICSIGNAL"

The individual signal phases are assigned to the signal lights 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.

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

Fig.4-17: Signal phase assignment

Adding Function Block "WAIT" in "ST"

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

Highlight "Application" and click the right mouse button. 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. In the selection area under "Type:" select under "Function Block" and as the "Implementation language:" (programming language) "Structured Text (ST)".
3. Confirm using "Finish".

Refer to the following figure.

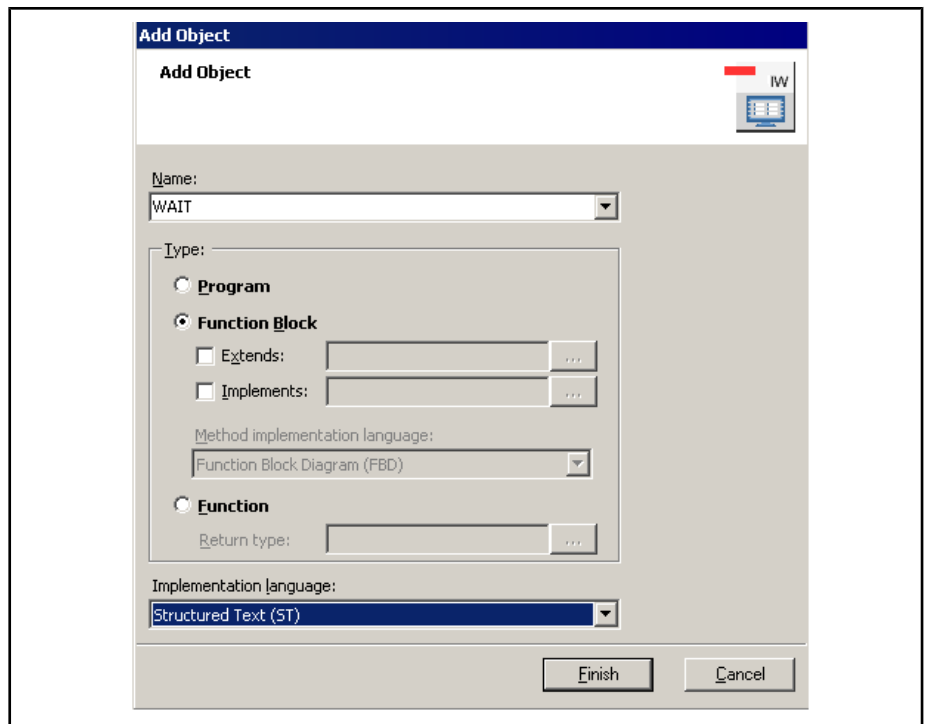


Fig.4-18: Adding "WAIT" function block

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

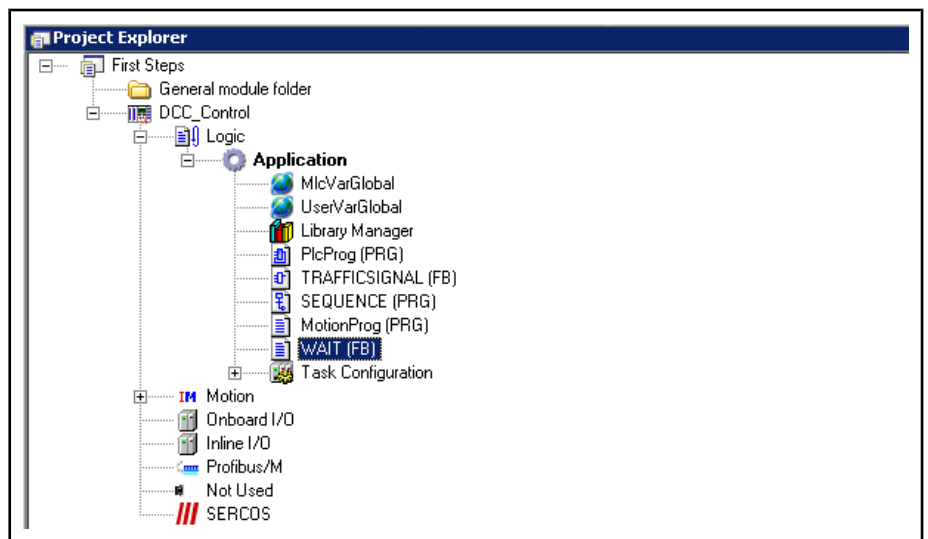


Fig.4-19: "WAIT" function block as part of the application

Function of "WAIT"

In the "WAIT" POU function block, a simple time phase 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.

4.6.3 Editing Programs and Function Blocks - Part 1

Editing "TRAFFIC SIGNAL" Function Block in "FBD"

Opening the POU

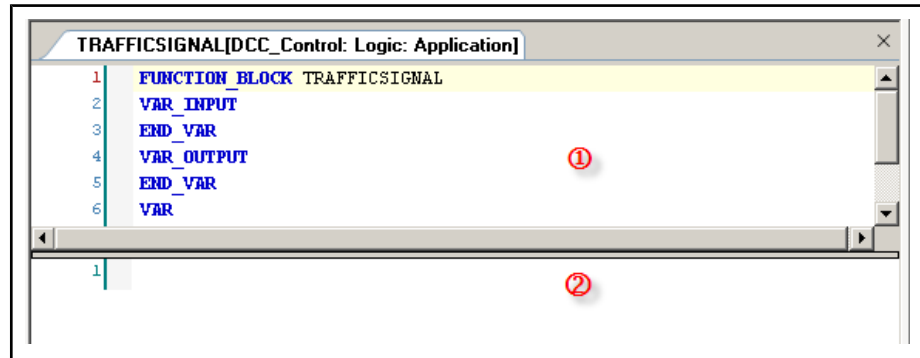
Open the "TRAFFICSIGNAL" function block by positioning the cursor over the "TRAFFICSIGNAL" POU and right-clicking to open the context menu. Select

First Steps - IndraLogic

Open from the context menu. The Editor of the "TRAFFICSIGNAL" function block opens; see the following figure: "Editor of 'TRAFFICSIGNAL' function block opened".



Alternatively you can open the editor for a POU with a double-click.



(1) Declaration

(2) Implementation section or body

Fig. 4-20: Open Editor of the "TRAFFICSIGNAL" function block

The POU editor

The editor is sub-divided into two sections:

1. the declarations and
2. the implementation section

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

The user program is edited in the implementation section.

Declaring variables

There are two 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 (STRUCTs) and of data arrays (ARRAYs) 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 to you again, 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 statement section. Select the first network in the statement section by clicking next to the "1" in the editor. Use the right mouse button to open the context menu and select **Insert FB call**. Alternatively, select **FBD/LD/IL ► Insert FB call** from the menu.

First Steps - IndraLogic

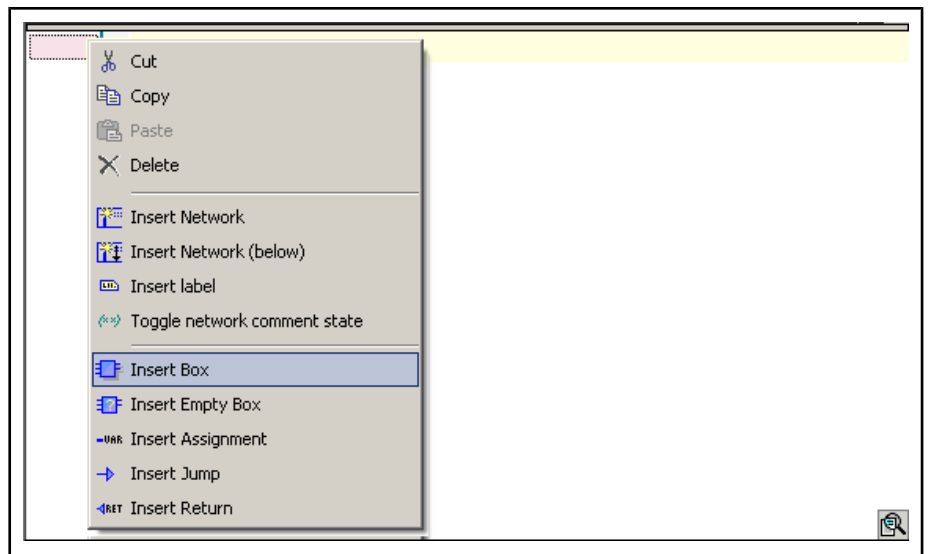


Fig.4-21: Adding an FB call

The "input assistance" window opens:

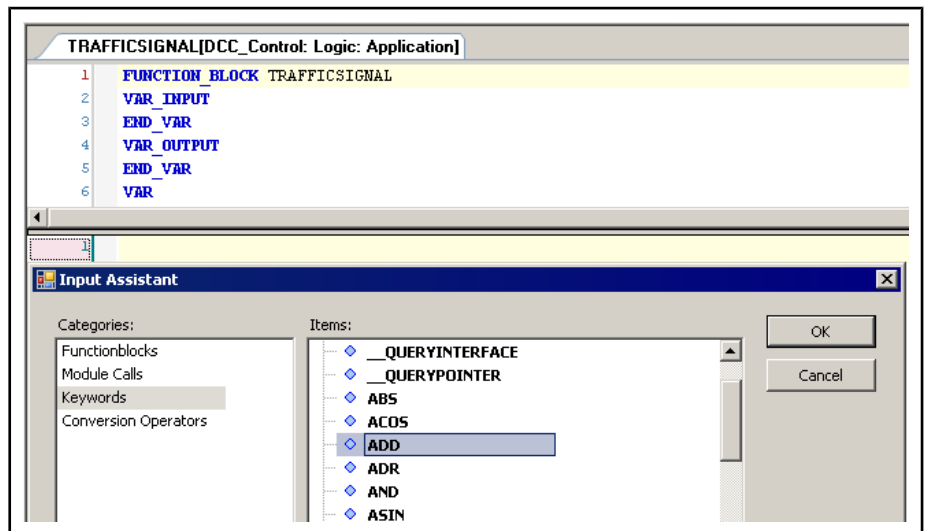


Fig.4-22: Input assistance - adding the function block "ADD"

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

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

Confirm by clicking on "OK".

The statement section looks like this:

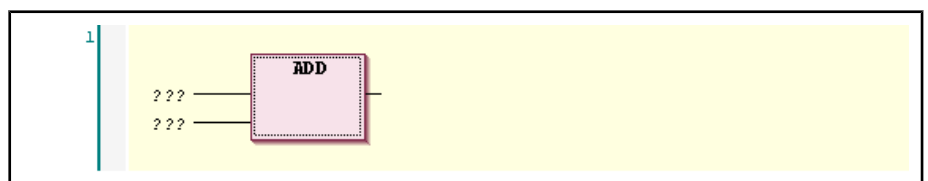


Fig.4-23: "ADD" function block added to the network

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

First Steps - IndraLogic

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

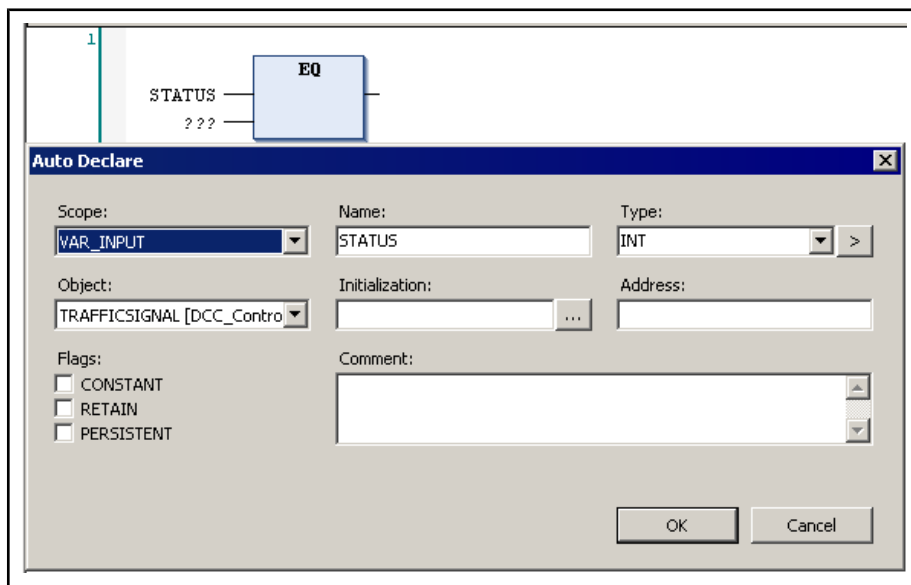


Fig.4-24: Declaration of the "STATUS" input variables

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

4. At the lower "???" input enter "1".
"STATUS" will have four possible states for the different signal phases. These states are "green", "yellow", "yellow-red" and "red".
The four states should be decoded as output variables. The three variables "GREEN", "YELLOW" and "RED" are needed for the four states.
1. Now, in network "1", click in an area following the "EQ" function block to add an assignment to the block. Click the right mouse button and select **Add assignment** in the context menu. Click on the output of the "???" function block and give it the variable name "GREEN".

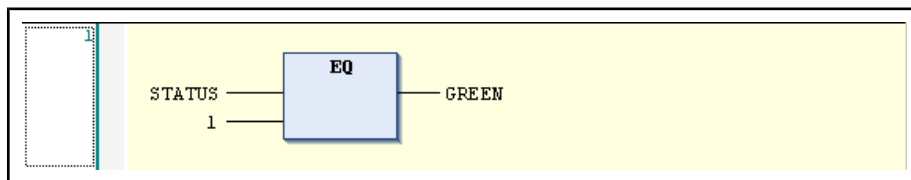


Fig.4-25: "EQ" function block

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, appears the declaration line `GREEN: BOOL;`.

Function of "EQ" 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. Therefore, click on first the 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 **Insert FB call** in the context menu.
3. In "Input assistance" for "Insert FB call" under "Categories:" select "Keywords".

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

Confirm with the "OK" button.

4. Highlight the output for the "EQ" function block and click the right mouse button.
5. Select **Insert FB call** in the context menu.
6. In "Input assistance" for "Insert FB call" under "Categories:" select "Keywords".

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

Confirm with the "OK" button:

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, appears 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 **Insert FB call** from the context menu.
3. In "Input assistance" for "Insert FB call" under "Categories:" select "Keywords".

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

Confirm with the "OK" button.

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>.

First Steps - IndraLogic



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 because it involves copying the second network.

To copy the network, proceed as follows:

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

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

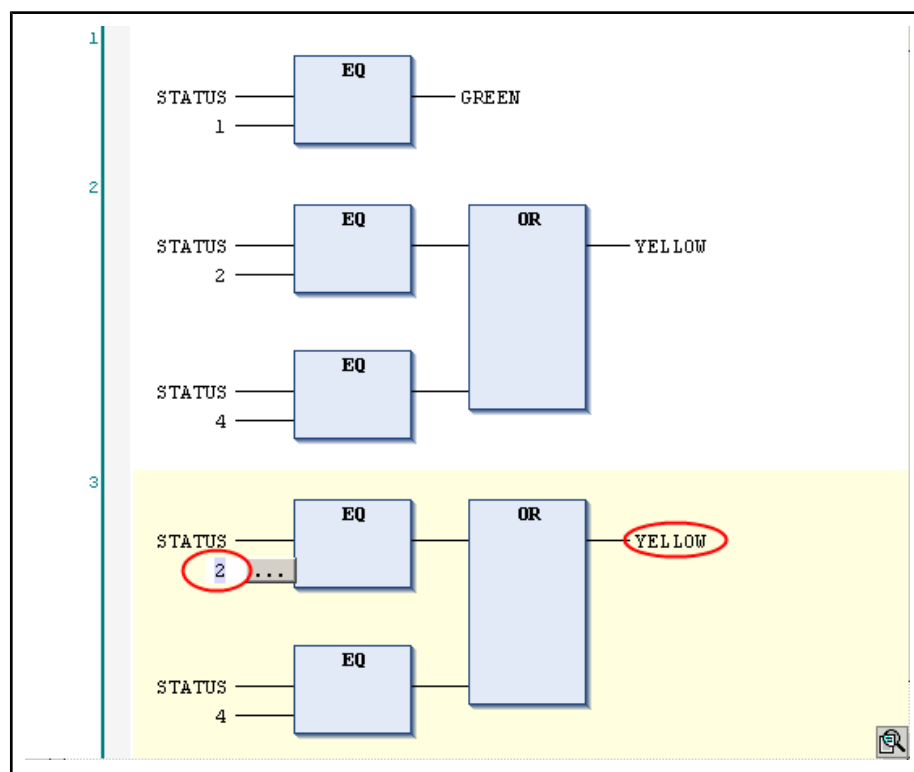


Fig.4-26: Third network added by copying

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".

First Steps - IndraLogic

3. Now, at the output of the "OR" block, left click on the variable assignment "YELLOW" and enter "RED".
4. Complete the entry in the window to declare the "RED" variables.
In the declaration section of the function block, in the "VAR_OUTPUT...END_VAR" area, appears the declaration line RED: BOOL;.

The first POU function block "TRAFFICSIGNAL" is complete.

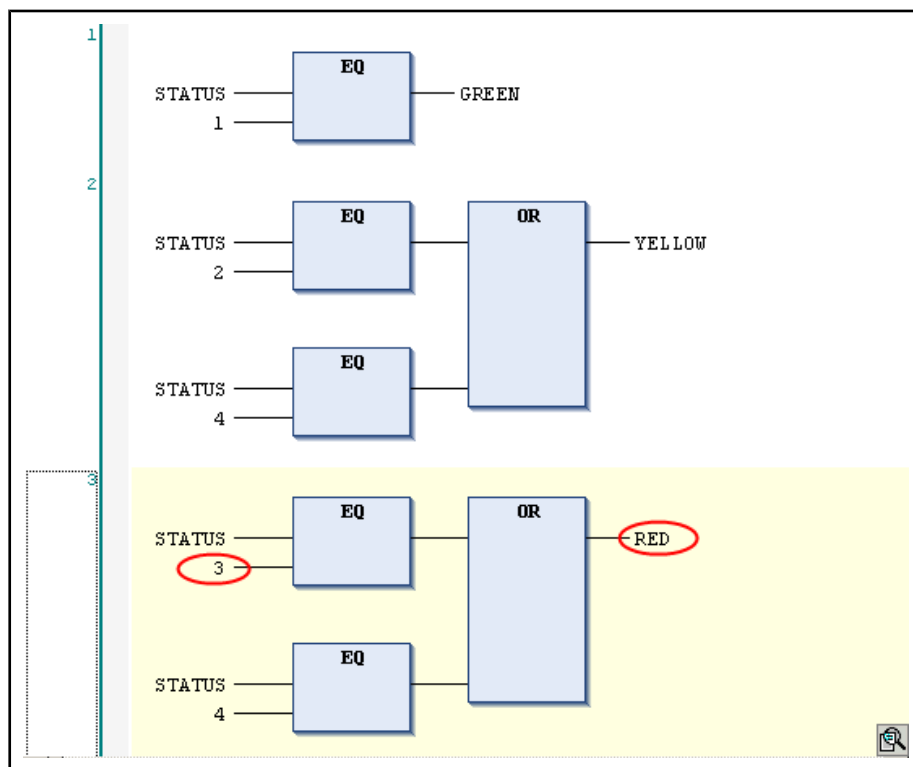


Fig.4-27: Completed "TRAFFICSIGNAL" POU

The declaration section is created incidentally to the implementation:

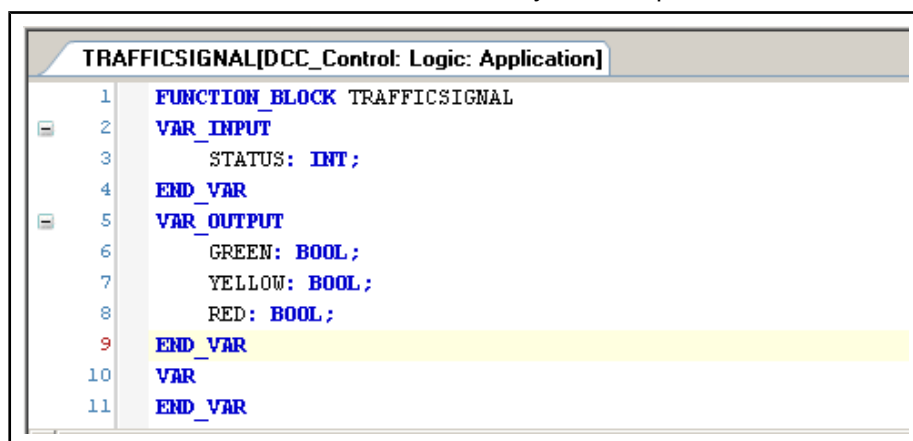


Fig.4-28: Declaration section of the "TRAFFICSIGNAL" function block

Editing "WAIT" Function Block in "ST"

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

First Steps - IndraLogic

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;.

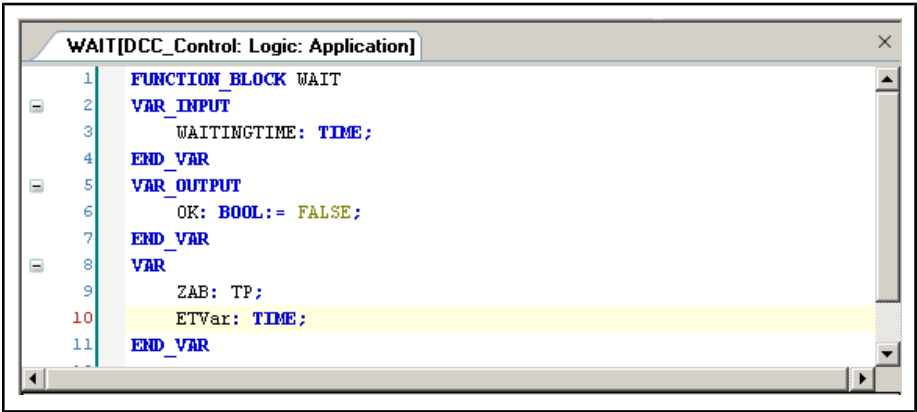


Fig.4-29: "WAIT" POU variable declaration

"TP" function block from 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 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.

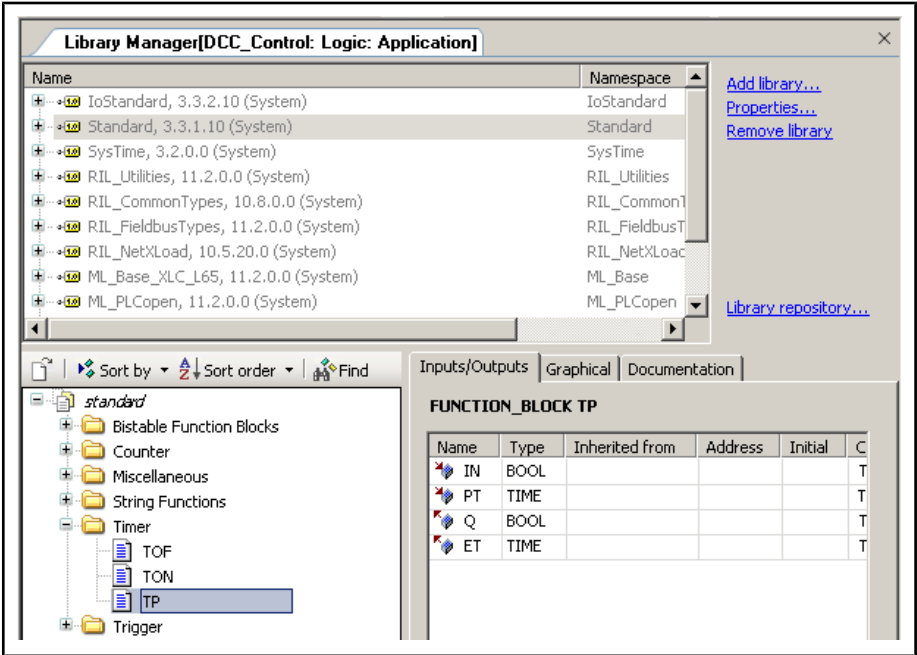


Fig.4-30: Library Manager and standard function block "TP"

First Steps - IndraLogic

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

Function of "TP"

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" achieves a value of "PT", "ET" stops counting.

"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".



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[DCC_Control: Logic: Application]
1  FUNCTION_BLOCK WAIT
2  VAR_INPUT
3    WAITINGTIME: TIME;
4  END_VAR
5  VAR_OUTPUT
6    OK: BOOL := FALSE;

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

Fig.4-31: Completed "WAIT" POU in ST

Function of "WAIT"

In the event that the time step has expired, it 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:

First Steps - IndraLogic

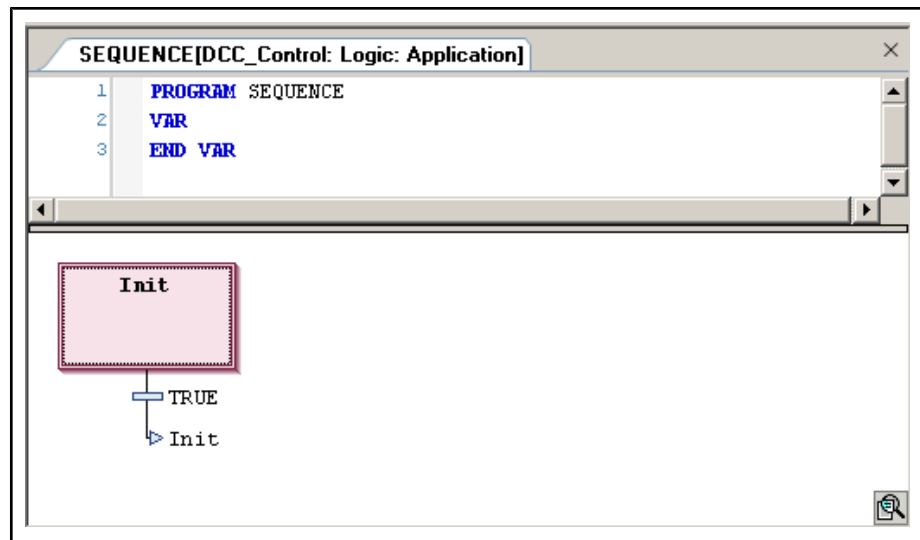


Fig.4-32: Open Editor for "SEQUENCE" program

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.

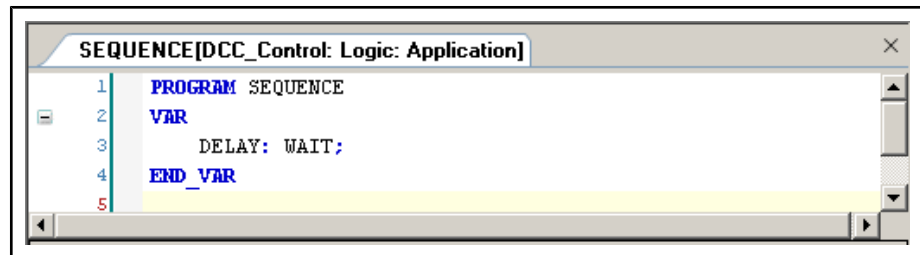


Fig.4-33: Variable declaration for "SEQUENCE" POU - first part

Creating the user program

At the beginning of a sequence cascade, an "Init" action, a subsequent "TRUE" transition and a return to "Init" are located in the body of the "SFC editor". See [Open editor for "SEQUENCE" POU, page 46](#).

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.

The sequence cascade looks like this:

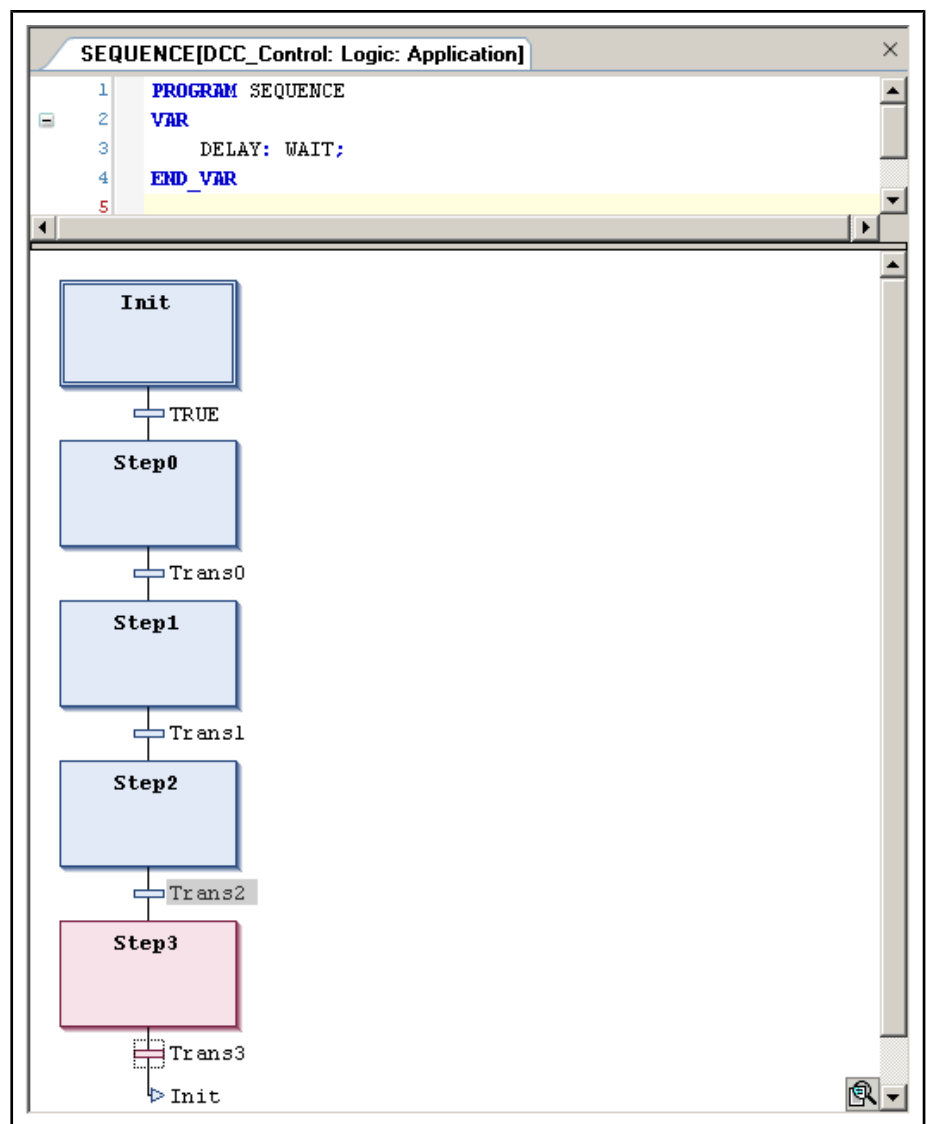


Fig.4-34: "SEQUENCE" program steps added



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

To program a transition, a transition must be added. Here this is done by double clicking the "transition", giving the transition a name and specifying the implementation language (programming language), in which the transition is to be written. Then, 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:

First Steps - IndraLogic

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

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

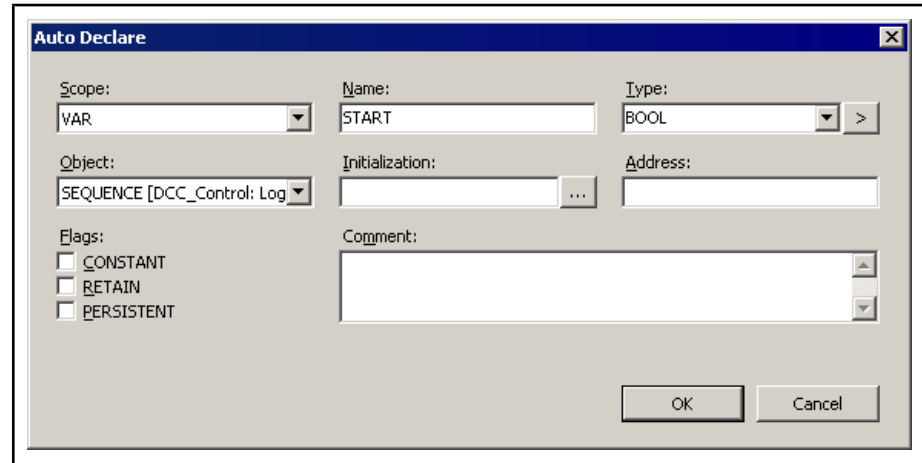


Fig.4-35: Input assistance for variable declaration

3. Under "Type:" select data type "BOOL".
4. For "Visibility:" select variable type "VAR_INPUT".
5. Confirm with "OK".

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".

The sequence cascade looks like this:

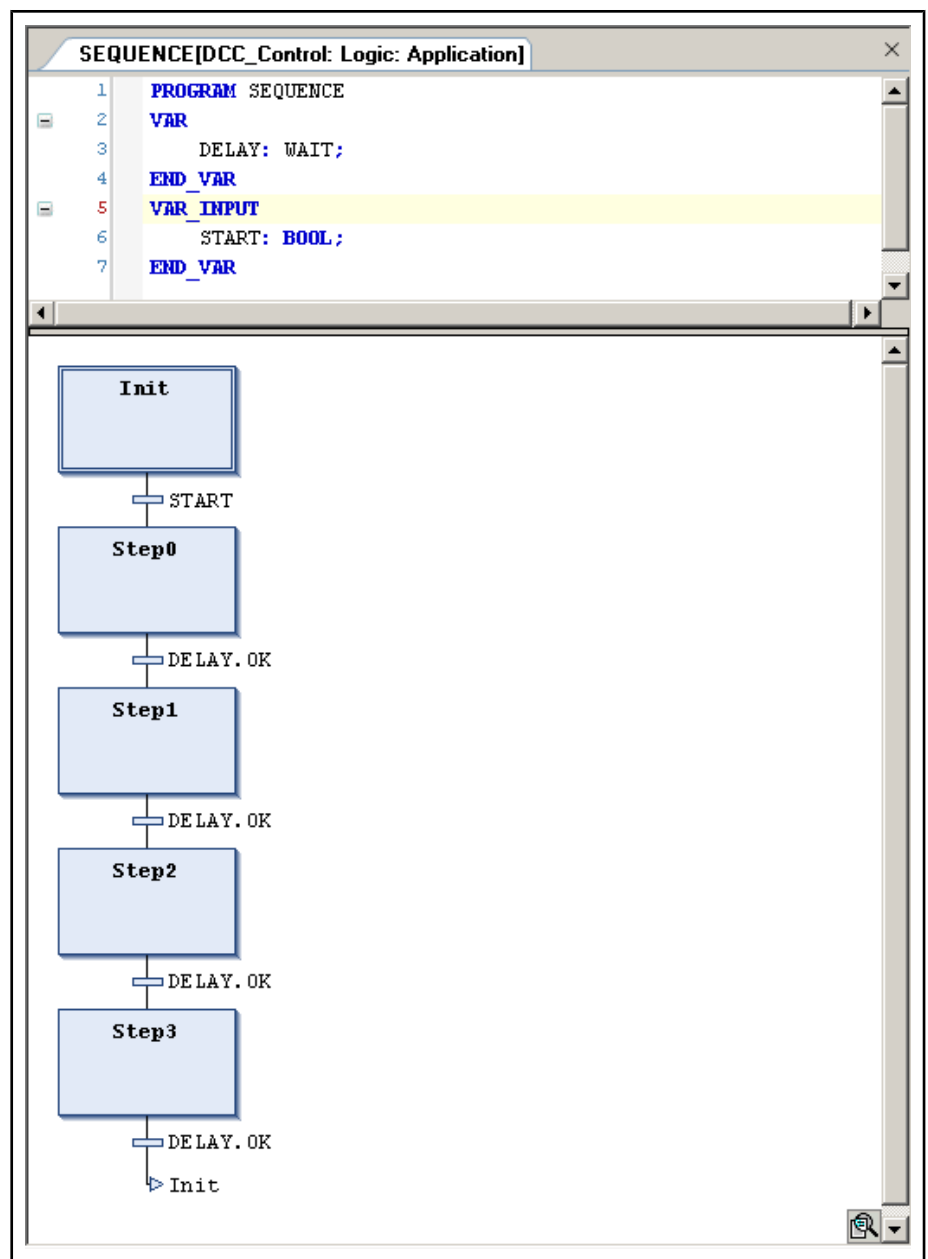


Fig.4-36: Steps of the sequence cascade

Naming the steps

Only the "Init" step keeps its name. The rest of the steps are renamed. They should be named "Switch1", "Green2", "Switch2" and "Green1", consecutively. To rename individual steps, proceed as follows:

1. Click on the name of the step.
2. Enter the desired name.
3. Confirm with the "enter key".

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.

First Steps - IndraLogic

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" there.

The sequence cascade looks like this:

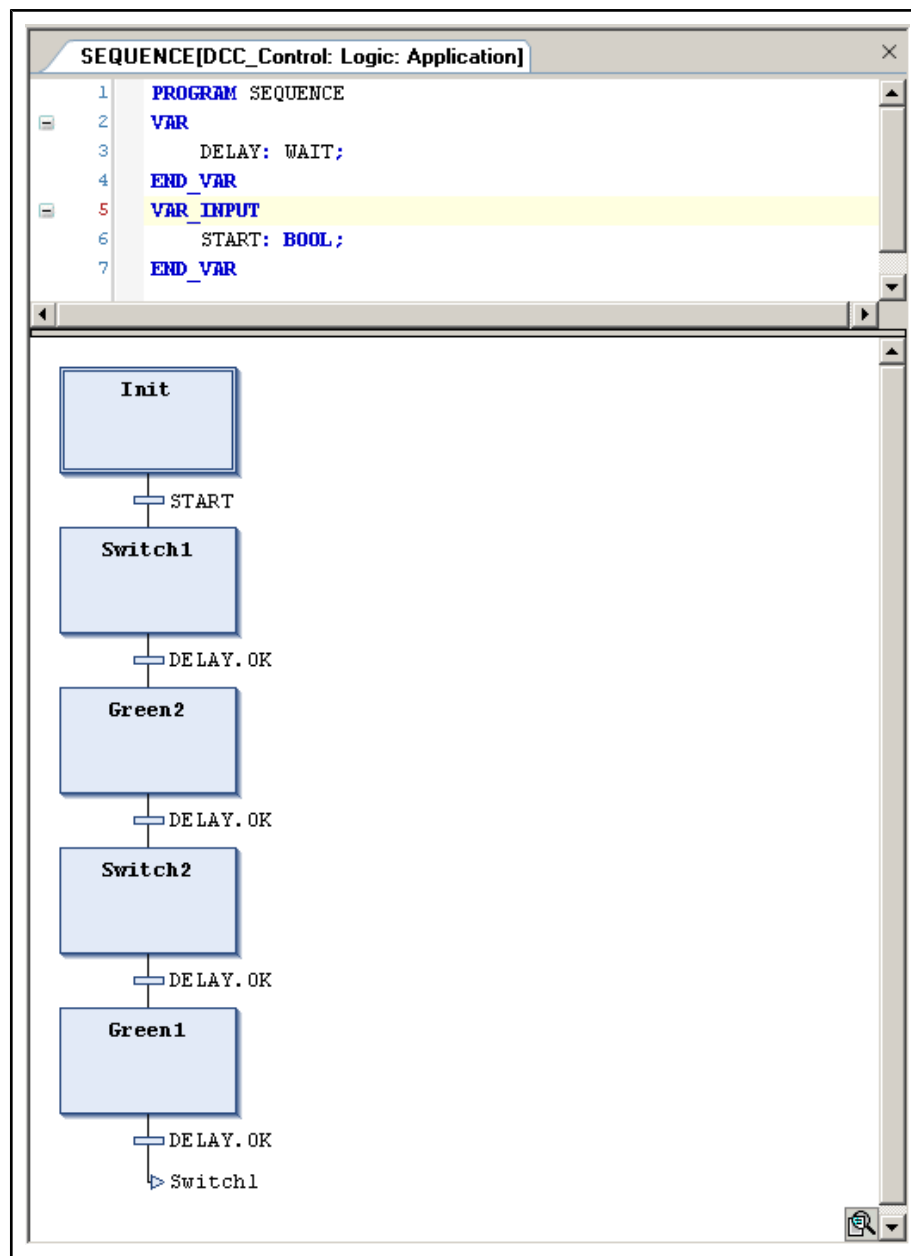


Fig.4-37: Steps of the sequence cascade after naming

Steps and actions

A function block written 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:

First Steps - IndraLogic

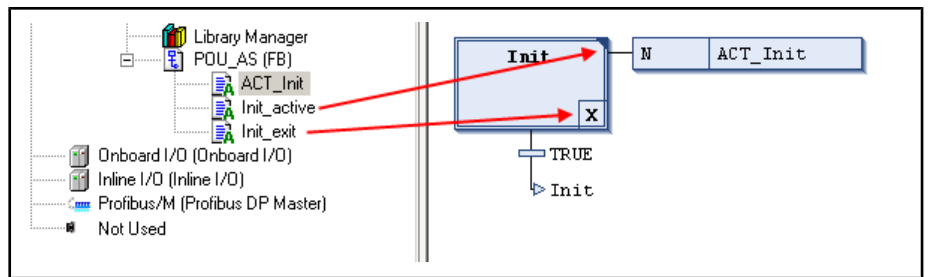


Fig.4-38: Types of step actions

- **Input action:** An input action is processed once precisely when the step becomes active. In this case: "Init_active", a step with an input action is designated by a black triangle in the upper right corner.
- **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". Their 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, it is post-processed precisely once.
- **Output action:** An output action is only executed once, before the step is disabled. In this case: "Init_exit". A step with an output action is 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 Project Explorer, under the "SEQUENCE (FB)" POU, a branch is generated with the action "Init_active".

First Steps - IndraLogic

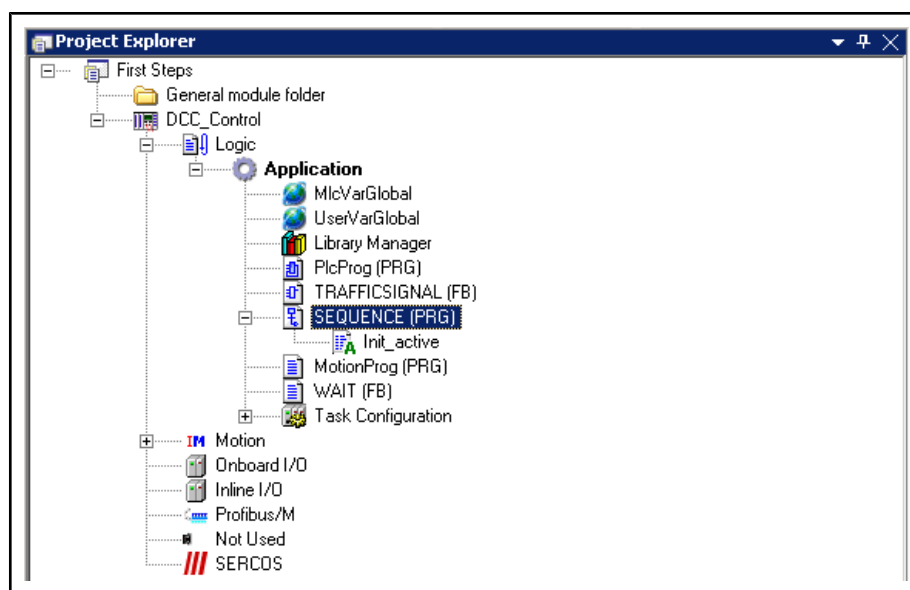


Fig.4-39: "SEQUENCE" POU in Project Explorer with "Init_active" action
Open the editor for the "Init_active" action.

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.

Programming the "Switch1" action with "ST"

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

In the action for the step "Switch1" the STATUS of traffic signal 1 (TRAFFICSIGNAL1) changes to "2" (yellow) and the STATUS for traffic signal 2 (TRAFFICSIGNAL2) 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 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:

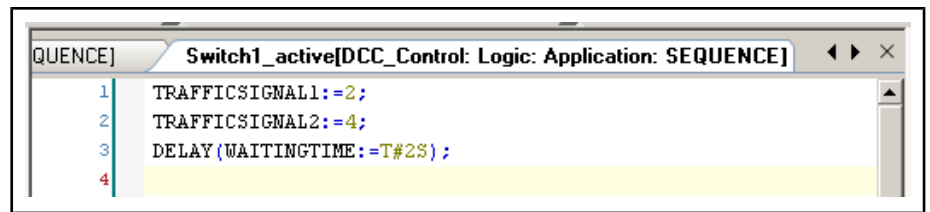


Fig. 4-40: Editor for the "Switch1_active" action

Programming the "Green2" action with "ST"

In the action for the step "Green2" the STATUS of traffic signal 1 (TRAFFICSIGNAL1) changes to "3" (red) and the STATUS for traffic signal 2 (TRAFFICSIGNAL2) 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 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:

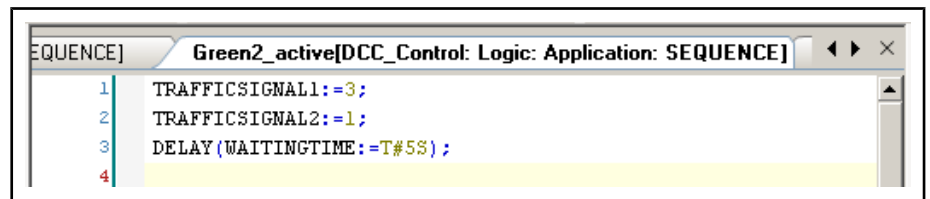


Fig. 4-41: Editor for the "Green2_active" action

Programming the "Switch12" action with "ST"

In the action for the step "Switch2" the STATUS of traffic signal 1 (TRAFFICSIGNAL1) changes to "4" (yellow-red) and the STATUS for traffic signal 2 (TRAFFICSIGNAL2) 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 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:

First Steps - IndraLogic

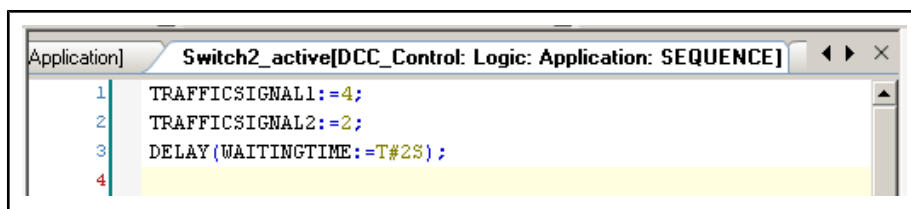


Fig.4-42: Editor for the "Switch2_active" action

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 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:

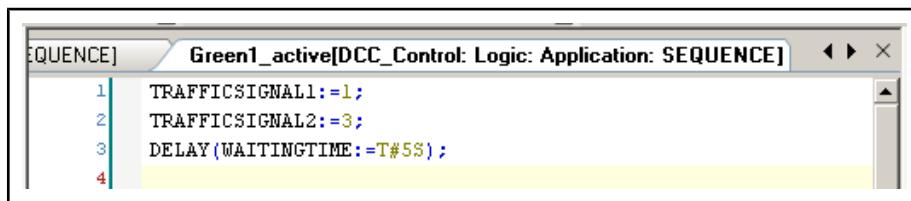


Fig.4-43: Editor for the "Green1_active" action

The completed "SEQUENCE" function block

The "SEQUENCE" function block and the structure in the Project Explorer is now complete; see the following figure.

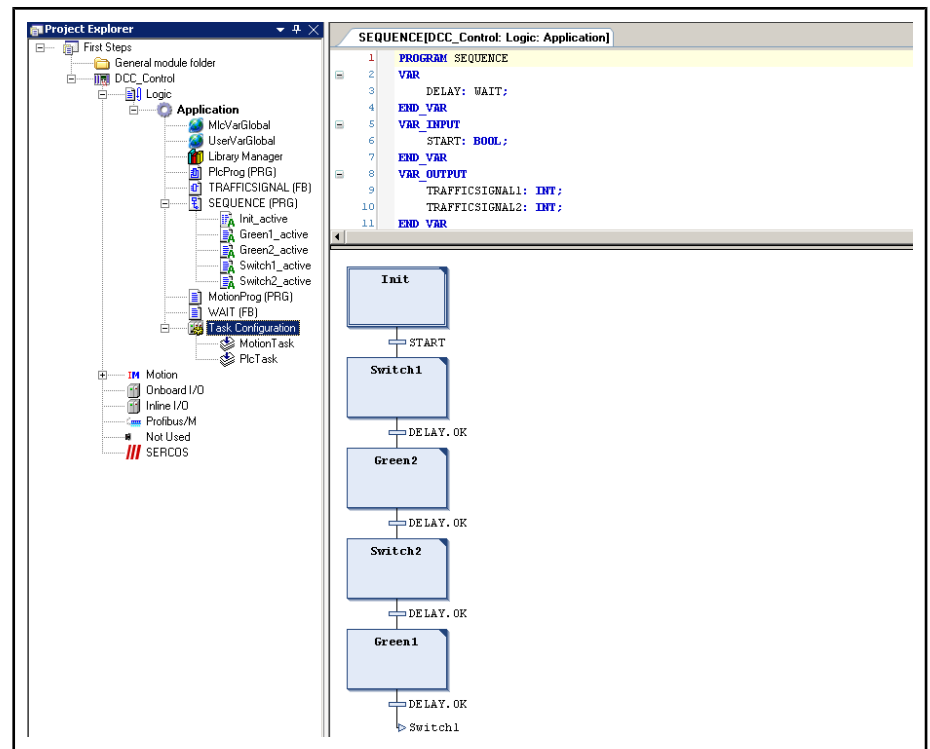


Fig.4-44: Completed "SEQUENCE" POU

4.6.4 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 POUs: "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 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 "DCC_Control" until "SEQUENCE" and "PlcProg" are visible.
4. Select "SEQUENCE" and confirm with the "OK" button.

The "Plc task configuration" now looks like this:

First Steps - IndraLogic

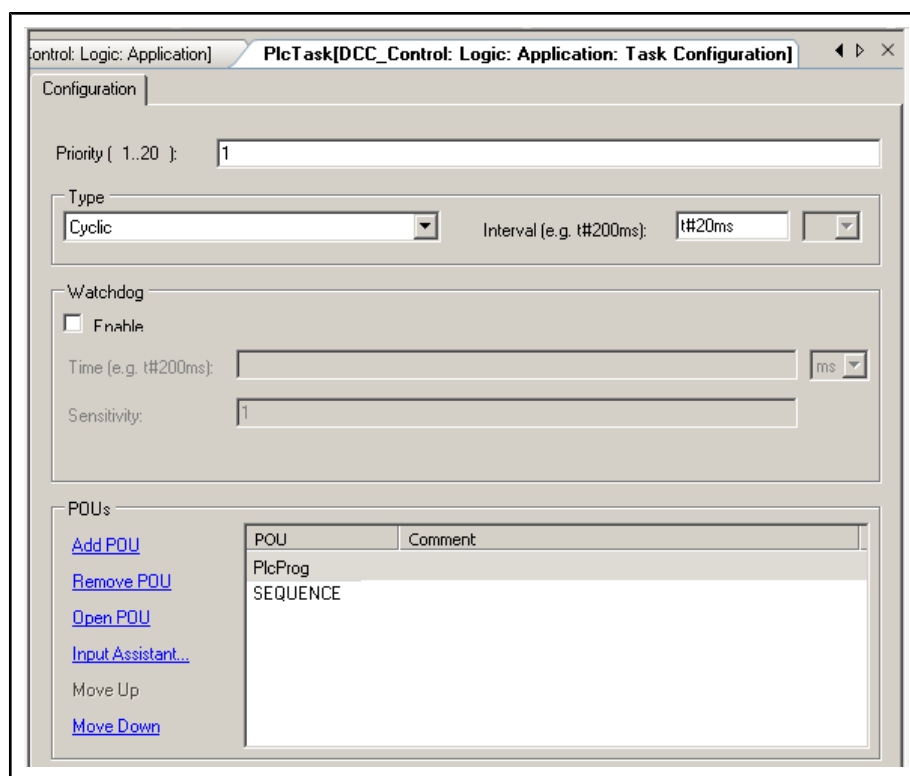


Fig.4-45: task configuration

4.6.5 Declaring Global Variables

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

The respective signal light of the two traffic lights 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 lights.

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

The "Global Variable List" obtains new "BOOL" type variables. Create the following variables:

Global variable list "GVL"

```
VAR_GLOBAL
  ON : BOOL;
  A1_green AT %QX0.0 : BOOL;
```

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

The name of the variable follows the "AT" assignment with a percent sign before the 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 XLC / MLC L45 and L65 have them as onboard outputs (standard mapping).

The controls XLC / MLC L25 must be extended by Inline modules or a field bus. If not, avoid the absolute address or you will obtain a warning when generating the code.

4.6.6 Testing "SEQUENCE" in Simulation Mode

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

Compiling the "Application"

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

Simulating a control

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

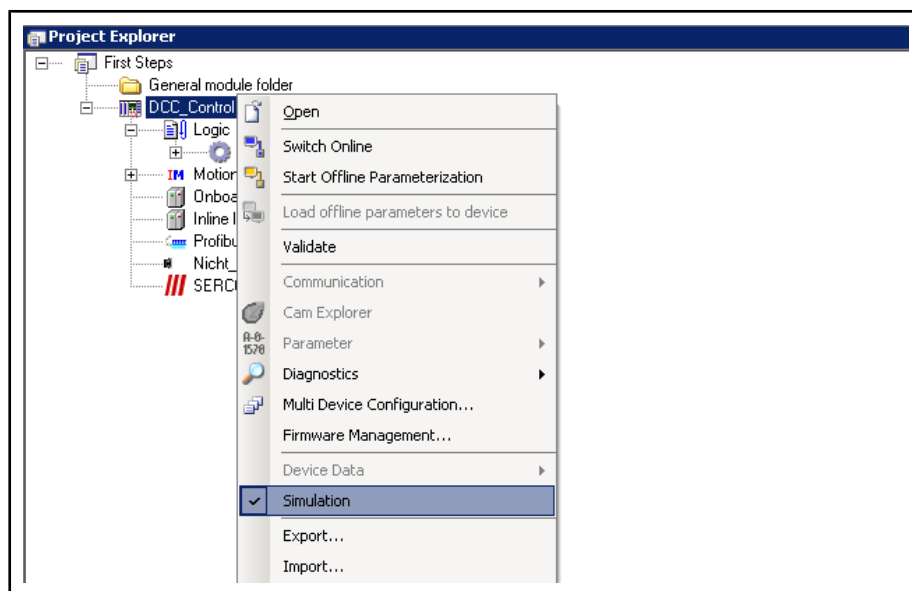


Fig.4-46: Selecting simulation

Logging into the control

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

First Steps - IndraLogic

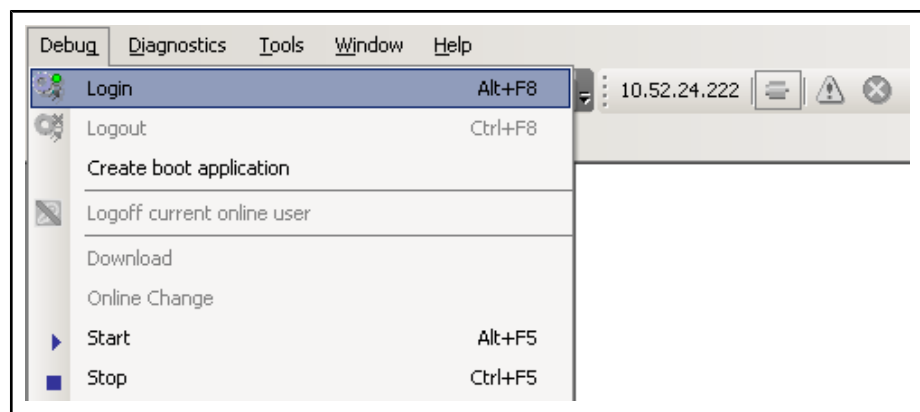


Fig.4-47: Logging into the control

The Project Explorer looks like this:

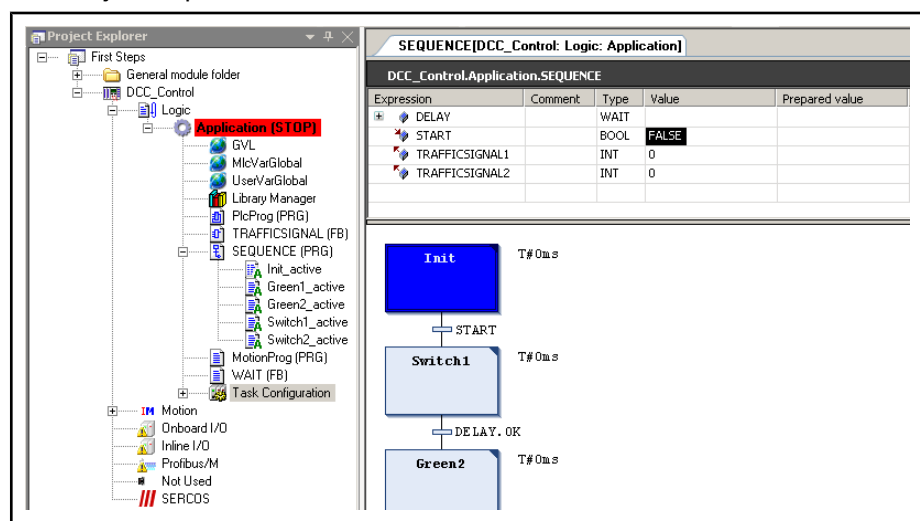


Fig.4-48: Logged into control (Simulation) - STOP application

Starting the "Application"

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

The application is running on the control.

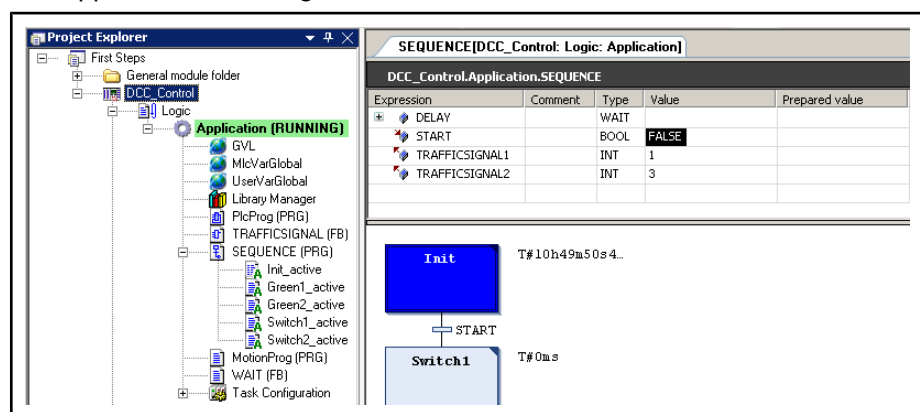


Fig.4-49: Logged into the control - application (RUNNING)

Starting "SEQUENCE"

Open the "SEQUENCE" POU. In the declarations' online mode you can see the variables and their current values. In the declarations, click in the line next to the "Start" variable in the "Prepared value" column until "TRUE" appears.

First Steps - IndraLogic

SEQUENCE[DCC_Control: Logic: Application]				
DCC_Control.Application.SEQUENCE				
Expression	Comment	Type	Value	Prepared value
+	DELAY	WAIT		
+	START	BOOL	FALSE	TRUE
+	TRAFFICSIGNAL1	INT	1	
+	TRAFFICSIGNAL2	INT	3	

Fig.4-50: "SEQUENCE" POU declarations online

Select **Debug** ► **Write values** (<Ctrl>+<F7>)..

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

SEQUENCE[DCC_Control: Logic: Application]				
DCC_Control.Application.SEQUENCE				
Expression	Comment	Type	Value	Prepared value
+	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.4-51: "SEQUENCE" POU declarations 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.

Logging out of the control

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 main menu.

The application is stopped on the control.

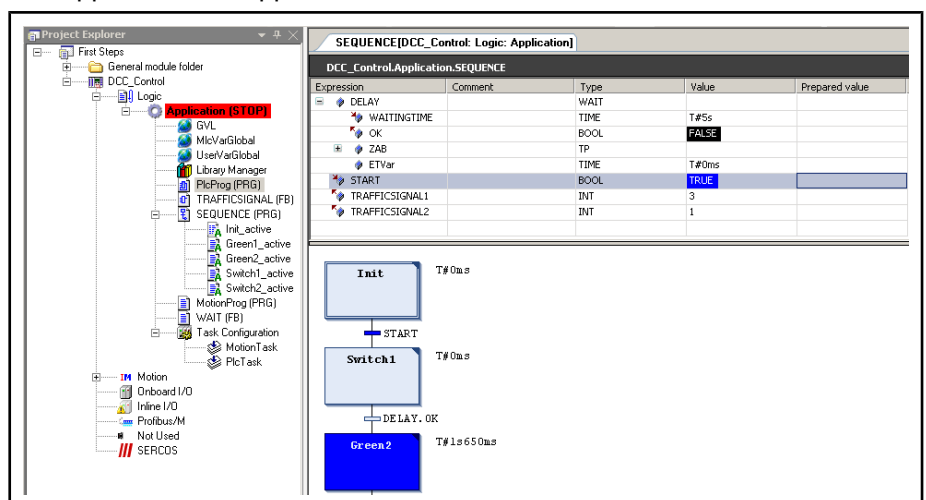


Fig.4-52: Logged into the control - application (STOP)

Log out by selecting **Debug** ► **Logout from**.

First Steps - IndraLogic

IndraWorks is in offline mode.



When there is another login, the values at the time of the last log out are used.

It is possible to start with a reassignment if the old data is deleted prior to the login (menu item **Create ► Clean**).

4.6.7 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 62](#),
- For programming in "Structured Text (ST)", continue below.

Function of "PlcProg"

The time sequence of the signal phases for both traffic lights 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 lights were created in the "Global variable list (GVL)". These variables can now be used in the "PlcProg" program. The application was previously tested only in simulation mode. The application can also be tested in a 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 38](#).

Create variable LIGHT1: TRAFFICSIGNAL;

and LIGHT2: TRAFFICSIGNAL:

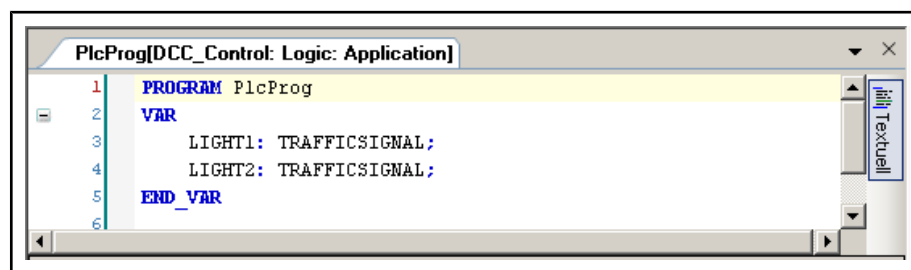


Fig.4-53: Declaration section of the "PlcProg" POU

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, because 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:

```

1  PROGRAM PlcProg
2  VAR
3      LIGHT1: TRAFFICSIGNAL;
4      LIGHT2: TRAFFICSIGNAL;
5  END_VAR
6  SEQUENCE (START:=ON);
7  LIGHT1 (STATUS:=SEQUENCE.TRAFFICSIGNAL1,
8      GREEN=>A1_green,
9      YELLOW=>A1_yellow,
10     RED=>A1_red);
11 LIGHT2 (STATUS:=SEQUENCE.TRAFFICSIGNAL2,
12     GREEN=>A2_green,
13     YELLOW=>A2_yellow,
14     RED=>A2_red);
15

```

Fig.4-54: "PlcProg" program completed

Testing the Application in Simulation Mode, PlcProg in ST

Compiling the "Application"

In order to be able to load the application into the control, the application must first be compiled by the "compiler" with no mistakes. To compile the application, select **Create ► Build**. After compiling, the following appears 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

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

Logging into the control

You should log into the control and load the application into the control. To log in on the control, select **Debug ► Login** in the 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 a simulation.

Switching on the traffic light and observing the signal cycle

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

1. Double click on the "GVL" in the Project Explorer to open the "Global variable list".

First Steps - IndraLogic

GVL[DCC_Control: Logic: Application]				
DCC_Control.Application.GVL				
Expression	Type	Value	Prepared value	
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.4-55: "GVL" online

2.
- Click next to the "ON" variable in the "Prepared value" column such that "TRUE" appears there.
3.
- To write the value to the control, click on **Debug ► Write values**.
The signal cycle will be processed. The signal cycle can be seen in the "PlcProg" function block.

PlcProg[DCC_Control: Logic: Application]				
DCC_Control.Application.PlcProg				
Expression	Comment	Type	Value	Prepared value
LIGHT1		TRAFFICSIGNAL		
STATUS		INT	4	
GREEN		BOOL	FALSE	
YELLOW		BOOL	TRUE	
RED		BOOL	TRUE	
LIGHT2		TRAFFICSIGNAL		
STATUS		INT	2	
GREEN		BOOL	FALSE	
YELLOW		BOOL	TRUE	
RED		BOOL	FALSE	

```
1  SEQUENCE (START:= GVL.ONTRUE);
2
3  LIGHT1 (STATUS[ 4 ]:= SEQUENCE.TRAFFICSIGNAL1[ 4 ],
4    GREENFALSE=> GVL.A1_greenFALSE,
5    YELLOWTRUE=> GVL.A1_yellowTRUE,
6    REDTRUE=> GVL.A1_redTRUE);
7
8  LIGHT2 (STATUS[ 2 ]:= SEQUENCE.TRAFFICSIGNAL2[ 2 ],
9    GREENFALSE=> GVL.A2_greenFALSE,
10   YELLOWTRUE=> GVL.A2_yellowTRUE,
11   REDFALSE=> GVL.A2_redFALSE);
12  RETURN
```

Fig.4-56: "PLCProg" program online

Stopping an application

Stop the PLC program via the menu bar with **Debug ► Stop**.

Logging out of the control

To log out of the control, select **Debug ► Logout**.

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 60](#),
- When programming in "Continuous Function Chart (CFC)", continue below.

Function of "PlcProg" The time sequence of the signal phases for both traffic lights 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 lights were created in the "Global variable list (GVL)". These variables can now be used in the "PlcProg" POU. The application was previously tested only in simulation mode. The application can also be tested in a control.

Declaring an instance The "PlcProg" program receives a "LIGHT1" instance, type "TRAFFICSIGNAL", and a "LIGHT2" instance, also "TRAFFICSIGNAL" type.

This instances the "TRAFFICSIGNAL" function block.

Create variable LIGHT1: TRAFFICSIGNAL; and LIGHT2: TRAFFICSIGNAL:

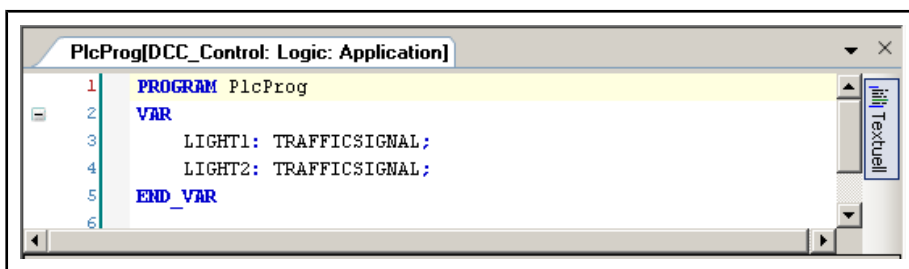


Fig. 4-57: Declaration section of the "PlcProg" POU

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".

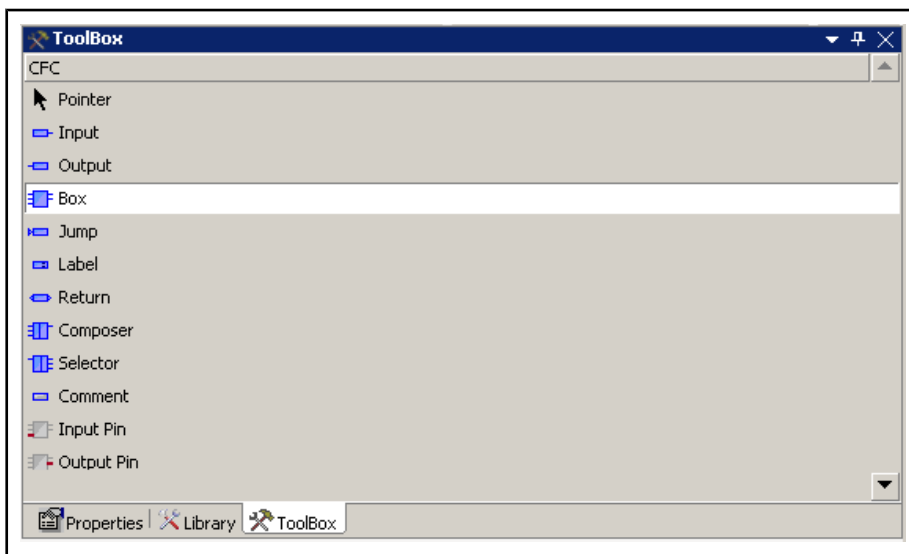


Fig. 4-58: Tools – Function block element

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 enter name the function block "SEQUENCE".

The FB looks like this:

First Steps - IndraLogic

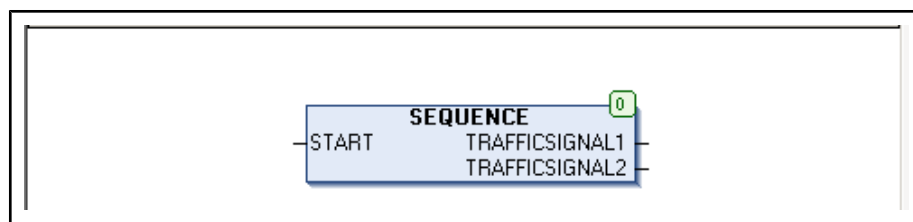


Fig.4-59: SEQUENCE function block

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:

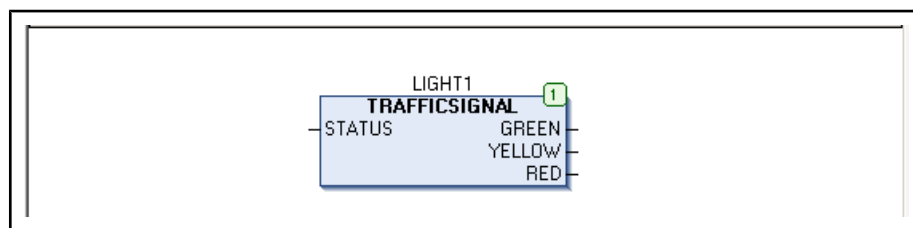


Fig.4-60: TRAFFICSIGNAL - LIGHT1 function block

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:

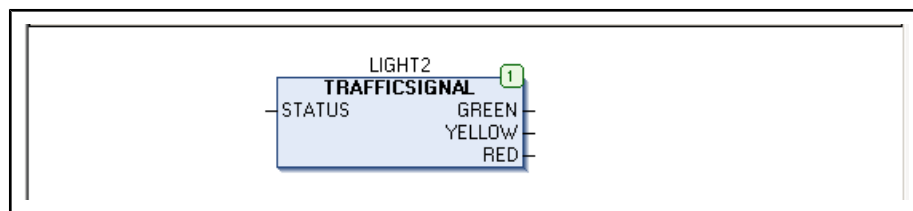


Fig.4-61: TRAFFICSIGNAL - LIGHT2 function block

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

1. In the "ToolBox" window, click to select the "Input" element.

First Steps - IndraLogic

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. Name the outputs "A1_green", "A1_yellow", "A1_red", "A2_green", "A2_yellow", "A2_red"



"A1_green" is correctly called "GVL.A1_green" or ".A1_green" for short, and so on, because 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.
3. 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.
4. 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.
5. Repeat the final step as needed for the other outputs.

The "PlcProg" program looks like this:

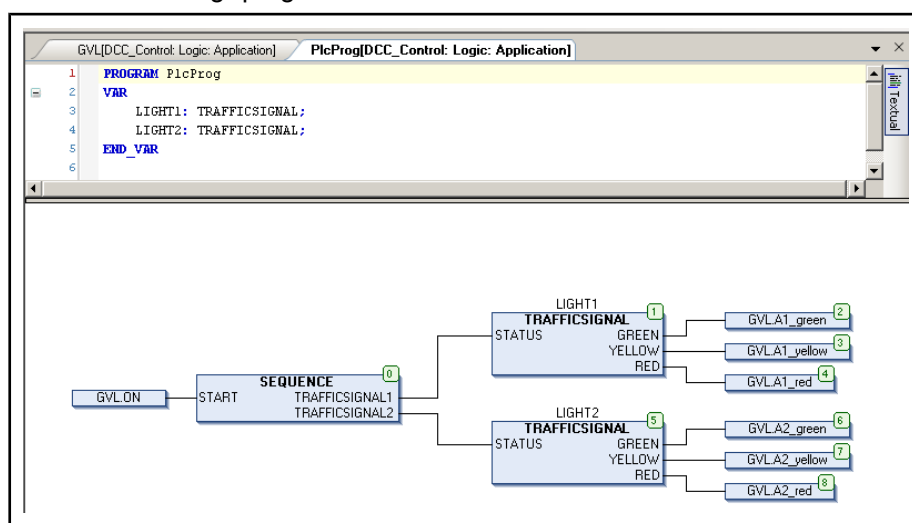


Fig.4-62: "PlcProg" POU completed

First Steps - IndraLogic



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

Compiling the "Application"

In order to be able to load the application into the control, the application must first be compiled by the "compiler" with no mistakes. To compile the application, select **Create ► Build Application**. After compiling, the following appears 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

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

Logging into the control

You should log into the control and load the application into the control. To log in on the control, select **Debug ► Login** in the 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 and select **Start** in the context menu. The "application" is now running as a simulation.

Switching on the traffic light and observing the signal cycle

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

1. Double click on the "GVL" in the Project Explorer to open the "Global variable list".

Expression	Type	Value	Prepared value
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.4-63: "GVL" online

2. Click next to the "ON" variable in the "Prepared value" column such that "TRUE" appears there.
3. To write the value to the control, click on **Debug ► Write values**.
The signal cycle will be processed. The signal cycle can be seen in the "PlcProg" function block.

First Steps - IndraLogic

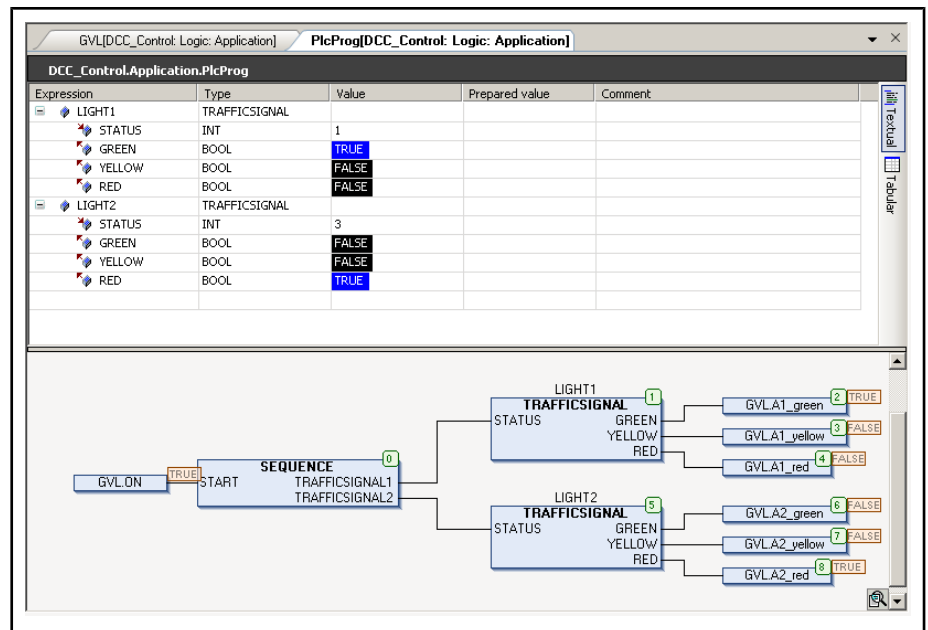


Fig.4-64: "PlcProg" program online

- Stopping an application Stop the PLC program via the menu bar with **Debug ► Stop**.
- Logging out of the control To log out of the control, select **Debug ► Logout**.

4.6.8 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 29](#), must be consistent with the control required. End simulation mode by right-clicking in the "Project Explorer" on the "DCC_Control" control to open the context menu. Click on **Simulation** to deactivate simulation mode. If simulation is no longer active, "Simulation" is deselected.
- 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 Application**. After compiling, the following appears 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**).
- Logging into the control You should log into the control and load the application into the control. To log in on the control, select **Debug ► Login** in the 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 generate a boot project using the menu bar, select **Debug ► Generate boot application**. 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.

First Steps - IndraLogic

Switching on the traffic light and observing 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 lights must be switched on so that the signal cycle is processed. To switch the traffic lights on, proceed as follows:

1. Double click on "GVL" in the Project Explorer to open the global variable list.

GVL[DCC_Control: Logic: Application]			
DCC_Control.Application.GVL			
Expression	Type	Value	Prepared value
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.4-65: "GVL" online

2. Click next to the "ON" variable in the "Prepared value" column such that "TRUE" appears there.
3. To write the value to the control, click on **Debug ► Write values**.

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. A selection window firstly appears offering the choice of "PlcProg.LIGHT1" or "PlcProg.LIGHT2".

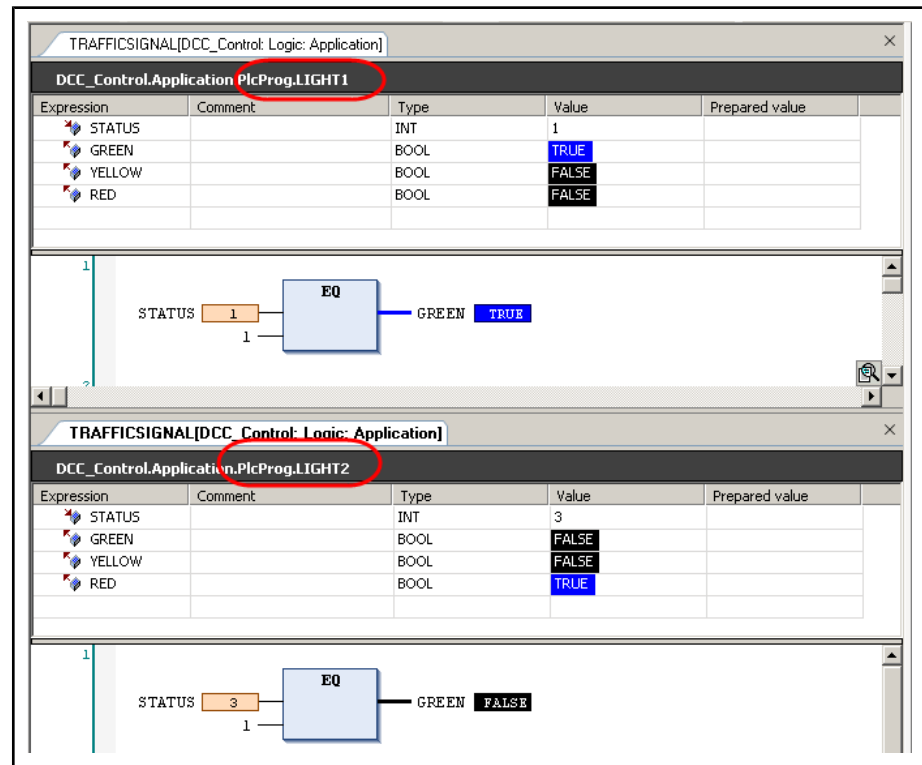


Fig.4-66: Status display in the two function block instances of TRAFFICSIGNAL

- 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** Stop the PLC program via the menu bar with **Debug ► Stop**.
- Logging out of the control** To log out of the control, select **Debug ► Logout**.

4.7 Creating Visualization for the PLC Program

4.7.1 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 32](#), is visualized. The following two signals for the programmed traffic lights and an ON switch are to be visualized and their corresponding status displayed.

4.7.2 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:

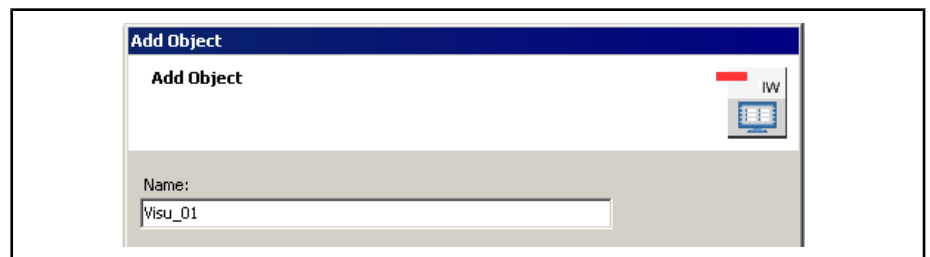


Fig.4-67: Adding a visualization to the application

2. Name the visualization as desired and confirm with "Finish".
Visualization is added together with an instance of the Visualization Manager:

First Steps - IndraLogic

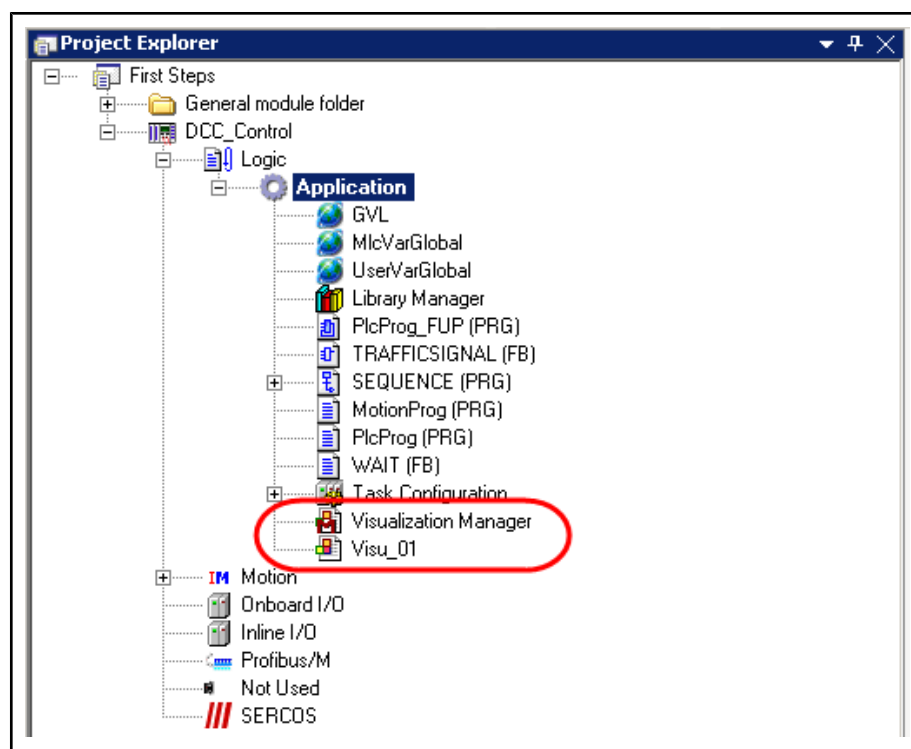


Fig.4-68: Visualization added to the application

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

4.7.3 Editing Visualization

Adding and Editing the First Element "Signal Lamp"

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**.

Three folders in the ToolBox are offered:

- Basic, general visualization objects,
- Windows Control, elements for designing a dialog,
- Complex Control, complex elements, for instance "measuring instruments", Trace objects.

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 "ToolBox" window, see the following figure.

First Steps - IndraLogic

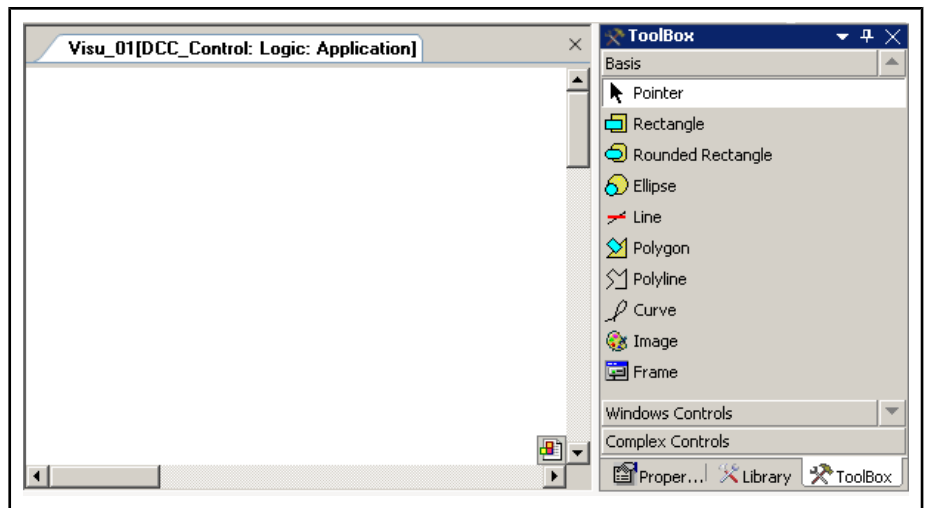


Fig. 4-69: "ToolBox" window

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 to the visualization window and left click again.

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

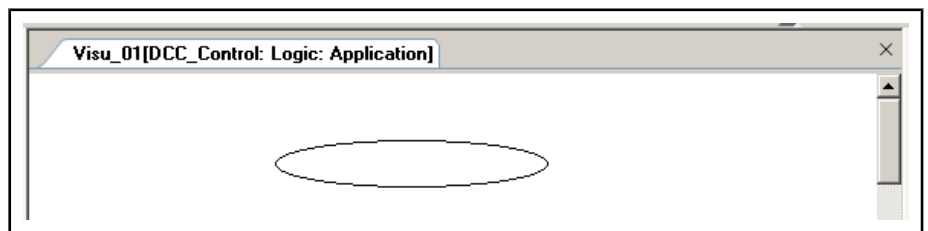


Fig. 4-70: Visualization editor - first ellipse

Changing the form of the ellipse

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

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

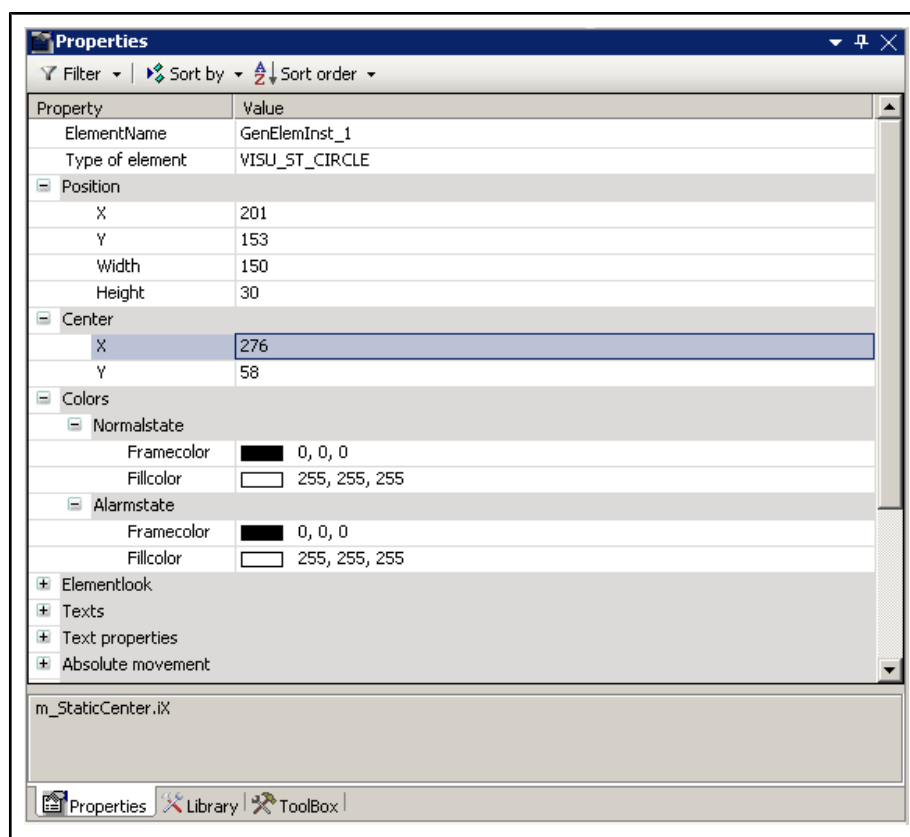


Fig.4-71: "Properties" window for the ellipse

- 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) as follows under "Center":

"X" = "75"

"Y" = "165".

Changing the properties of the signal lights

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

To change the fill color, proceed as follows:

- To change the fill color for the normal condition (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 "FillColors". Select "Silver".
- To change the fill color for the alarm status (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 "FillColors". Select "Red".

First Steps - IndraLogic

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

Proceed as follows:

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".

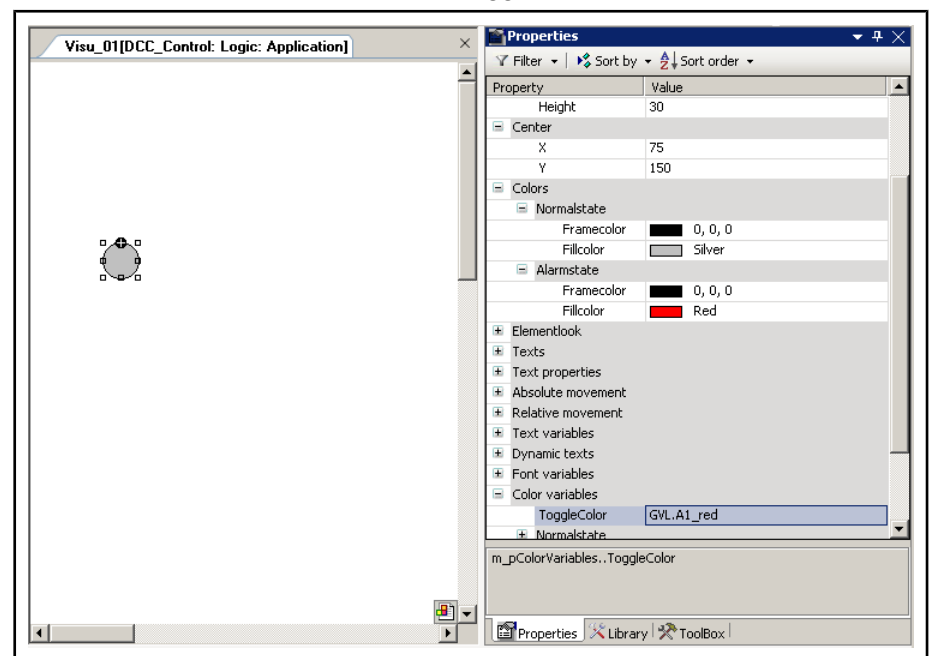


Fig.4-72: Properties – ToggleColor

Copying, Adding and Editing the "Signal Lamp" Element

Copying and inserting the "signal light" element

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

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

1. Click on the signal light to highlight it.
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 signal light, click again in a free area in the visualization editor and select **Insert** in the context menu.
5. Left click on the first signal light you added to highlight it.

First Steps - IndraLogic

Changing the position of the "signal light" element

Both of the signal lights you just added must now be repositioned.

To reposition the signal lights, proceed as follows:

1. Left click on the first signal light you added to highlight it.
2. In the "Properties" window under "Position", select (60, 195) for (X, Y) to change the position.
3. To change the center point, select (75, 210) for (X, Y) under "Center".
4. Left click on the second signal light you added to highlight it.
5. In the "Properties" window under "Position", select (60, 240) for (X, Y) to change the position.
6. To change the center point, select (75, 255) for (X, Y) under "Center".

Changing the properties of the "signal light" element

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

Proceed as follows:

1. Highlight the signal light in the center.
2. To change the fill color for the alarm status (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 signal lights.
6. In the "Color" window, click on "FillColor". Select "Green".

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

Proceed as follows:

1. Highlight the signal light 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. Highlight the signal light at the bottom.
6. Under "GVL variables" select the "A1_green" variable.

Adding and Editing the "Traffic Light Housing" Element**Elements of the traffic light housing**

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

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

Adding a "rectangle" element

To add the traffic light 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 light housing" element

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

First Steps - IndraLogic

1. Highlight the "rectangle" element you added and open the "Properties" window.
2. In the "Properties" window under "Position", select (50, 135) for (X, Y) to change the position.
3. In the "Properties" window under "Position" select (50, 150) for (Width, Height) to change the shape.
4. To change the center point, select (50, 210) for (X, Y) under "Center".
5. Right click next to the highlighted object to open the context menu. Select **Sequence ► Send to back**.

Changing the properties of the "traffic light housing" element

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

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

Adding and Editing "Traffic Light Mast" Element

Adding a "rectangle" element

Next, the traffic light mast will be 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 light mast" element

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

Changing the properties of the "traffic light mast" element

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

Adding and Editing "Traffic Light Base" Element

Adding a "rectangle" element

The first traffic light still needs a base.

To construct the traffic light base, proceed as follows:

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

Changing the properties of the "traffic light base" element

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

First Steps - IndraLogic

The first traffic light looks like this:

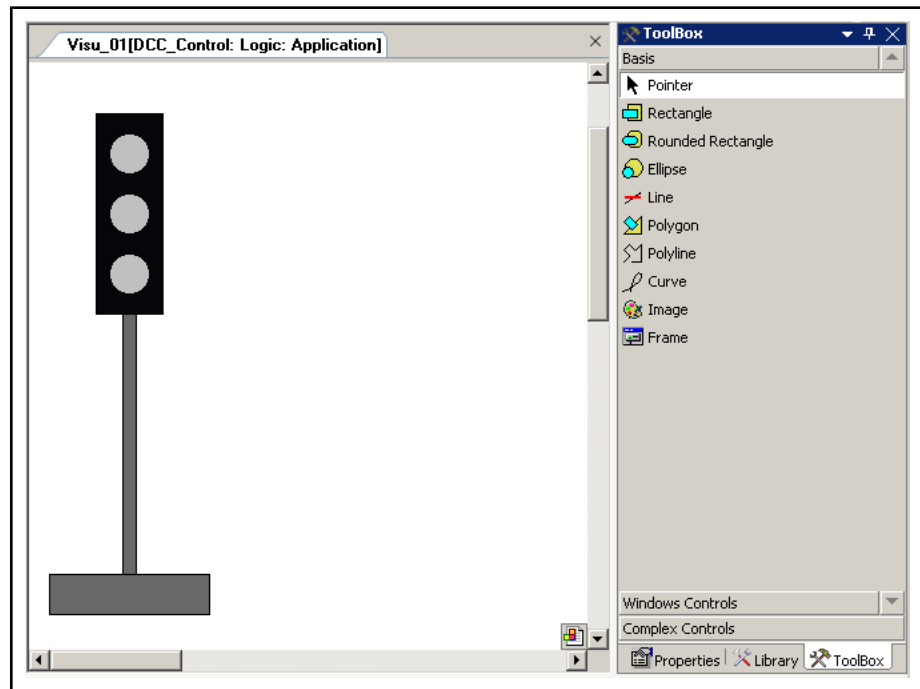


Fig.4-73: First traffic light

Adding and Editing Two Traffic Lights

The second traffic light

To create the second traffic light, highlight all of the traffic light elements. Left click in an area below and to the right of the traffic light and hold the mouse button down. Drag the mouse pointer to the left above the traffic light. Release the mouse button. All of the elements are highlighted.



Alternatively, you can highlight individual elements by pressing and holding the <Ctrl> key when highlighting the desired elements.

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

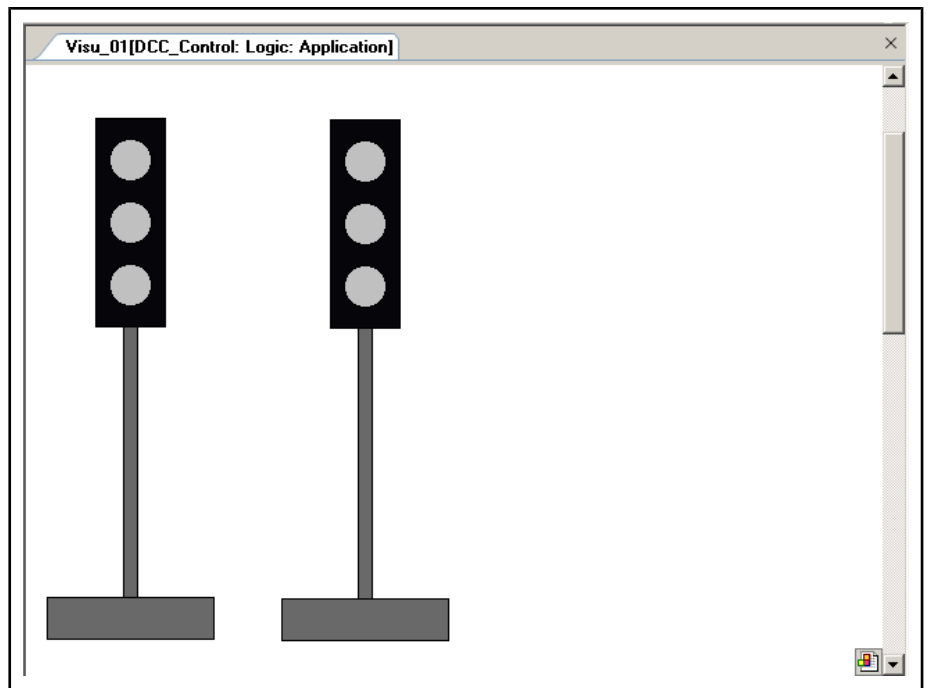


Fig. 4-74: Two traffic lights

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

Here the traffic light is to be given the ability to switch to the start phase. A "button" must be inserted:

1. In **ToolBox ► Windows Control**, 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 (130, 55) for (X, Y) to change the position.
2. In the "Properties" window under "Position" select (70, 40) for (Width, Height) to change the shape.

Changing the properties of the "button" element

3. To change the center point, select (165, 75) for (X, Y) under "Center".
1. In the "Texts" property under "Text" enter "ON".
2. In the "Input" property under "OnClick" 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.

First Steps - IndraLogic

Adding and Editing Labeling

Adding a "rectangle" element

Now both traffic lights need to be labeled as "Traffic light 1" and "Traffic Light 2".

To label the traffic lights, 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 (15, 520) for (X, Y) to change the position.
3. In the "Properties" window under "Position" select (120, 30) for (Width, Height) to change the shape.
4. To change the center point, select (75, 535) for (X, Y) under "Center".

Changing the properties of the "rectangle" element

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

The "Color" window opens.

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)

Copying and adapting the "rectangle" element

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

The visualization is complete:

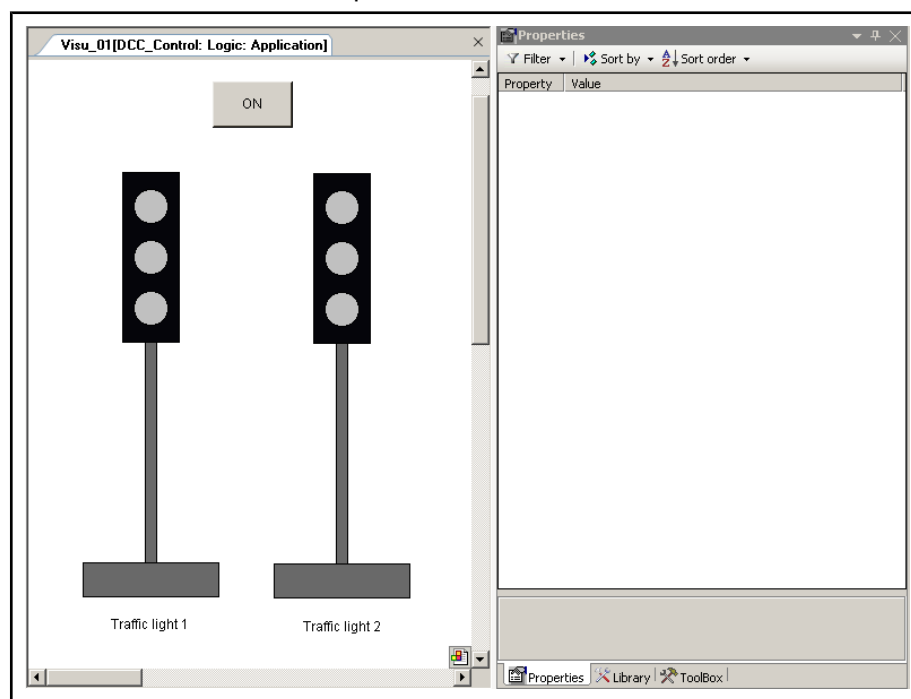


Fig.4-75: Completed visualization in visualization editor

Displaying Variable Values

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

Adding a "rectangle" element

The two traffic lights are to obtain under the labels "Traffic light 1" and "Traffic light 2" a display of their respective "STATUS" variables.

To label the traffic lights, 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 (15, 570) for (X, Y) to change the position.
3. In the "Properties" window under "Position" select (120, 30) for (Width, Height) to change the shape.
4. To change the center point, select (75, 585) for (X, Y) under "Center".

Changing the properties of the "rectangle" element

1. In the "Text" field under "Texts", enter "Traffic light 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 appears instead of the format specification "%s" in online mode.

Copying and adapting the "rectangle" element

1. Copy the display and place it under the second traffic light.
2. For display 2, replace:
 - Text "Traffic light 1: %s" by "Traffic light 2: %s",
 - Text variable "PlcProg.LIGHT1.STATUS" by "PlcProg.LIGHT2.STATUS".

Visualization with completed display:

First Steps - IndraLogic

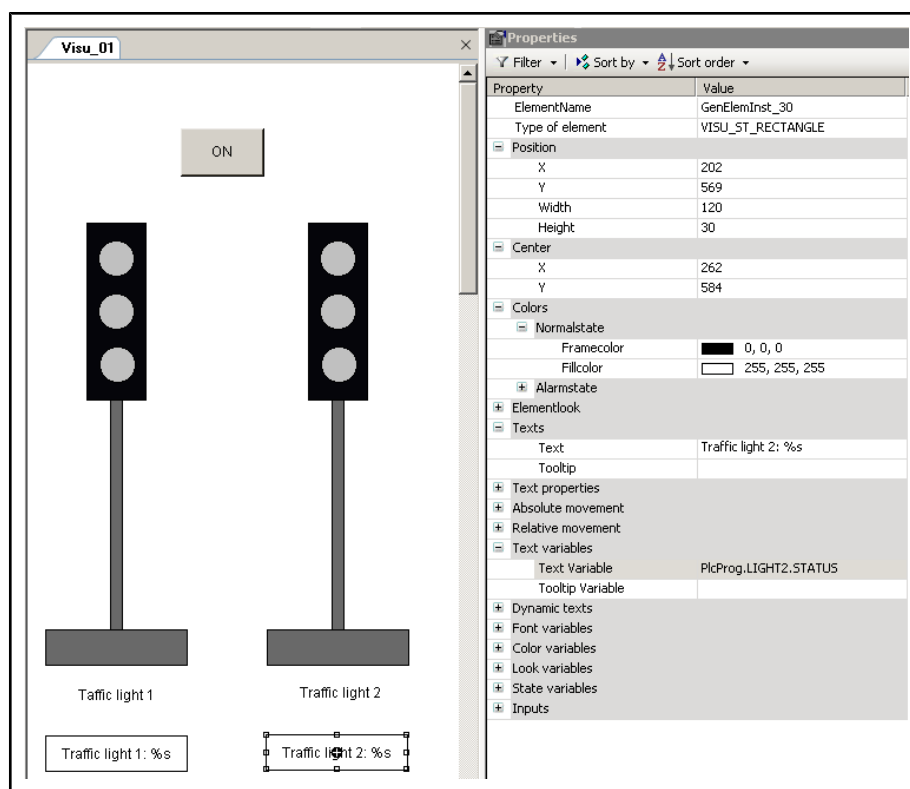


Fig.4-76: Visualization Editor completed by display

4.7.4 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 61 or ["Testing the Application in Simulation Mode \(PlcProg in CFC\)"](#), on page 66.

To test the visualization with a control, refer to ["Testing the application with a control"](#), page 67.

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

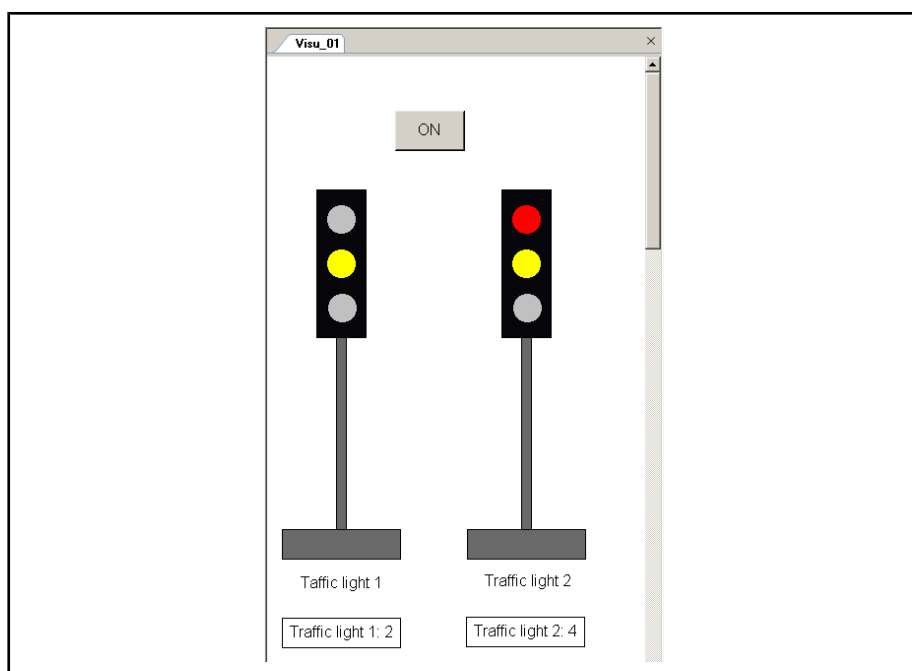


Fig. 4-77: Online visualization

Click on "ON". The traffic light cycles are processed.

4.8 Restoring a Project

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

The export file was created beginning from the "Application" node and contains the following components:

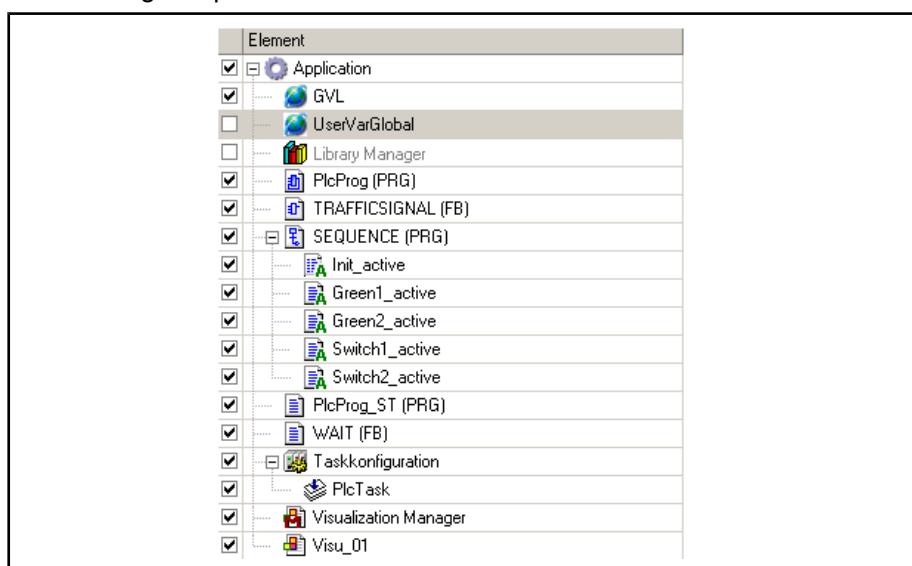


Fig. 4-78: Components of the export file

The given components are overwritten during import.

First Steps - IndraLogic

- Restore** Some involvement by the user is needed to ensure that the project runs on a control of your choice (XLC L25, L45, L65 (with or without SERCOS III bus) or MLC L25, L45, L65):
1. Clear old data in the control
Clear the old memory content of the control.
Details are described under [Resetting the Control \(CMLx5\), page 26](#).
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 MLC L25, L45, L65) from the Device Library 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. Not required if only operating in "Simulation" mode.
End adding as usual.
The desired control is now in the project.
 4. Import the export file "Application_FirstSteps_IndraLogic_Basics.iwx"
In the Project Explorer, expand the control node to **Logic ► Application**.
context menu "Application" **Import**.
Select the export file "Application_FirstSteps_IndraLogic_Basics.iwx" in "Iw:\All Programs\Rexroth\IndraWorks\Projects", "Overwrite" operation, 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 61 or "[Testing the Application in Simulation Mode \(PlcProg in ST\)](#)", on page 61.

To test the project with a control, refer to "[Testing the application with a control](#)", page 67.

5 First Steps - Motion

5.1 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 there is no a drive, the next step "Drive scanning" is omitted.

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

Subtasks

- Create an IndraWorks project, control and drives,
- load the Motion project from the control,
- program the motion sequence,
- load the PLC program on the control and test it.
- Record the motion of the axes using Trace.



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

- [Programming the Motion Sequence in Function Block Diagram \(FBD\), page 93,](#)
- [Programming the Motion Sequence in Structured Text \(ST\), page 117.](#)

5.2 Technical Requirements

- PC with installed IndraWorks Engineering for IndraMotion MLC, 12VRS, control, type "IndraMotion MLC L25", "IndraMotion MLC L45" or "IndraMotion MLC L65",
current firmware is loaded.

Meeting these prerequisites is described in

- Rexroth IndraMotion MLC 12VRS Functional Description, DOK-IM*ML*-SYSTEM**V12-RExx-EN-P and
- IndraWorks online help, **System IndraLogic XLC/IndraMotion MLC ▶ XLC/MLC Functional Description ▶ Commissioning and Operation**,

control, for instance of type "IndraMotion MLC L65.1"; the current firmware is loaded,

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

Drive controller, e.g. of type "IndraDrive M"; 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.

5.3 Create the Motion Subproject

5.3.1 Resetting the Control (CMLx5)

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

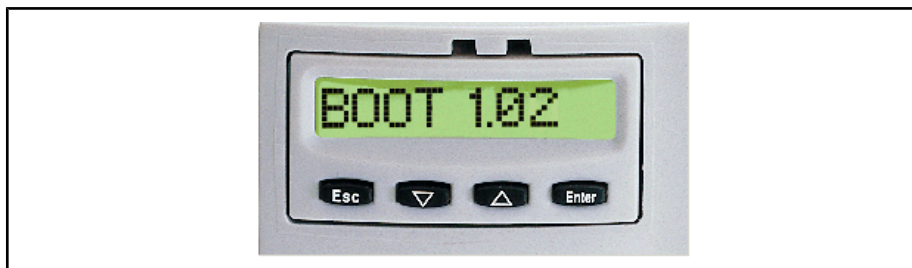


Fig.5-1: Rebooting the control

After the control is switched on, press the outer buttons <ESC> and <Enter> located below the display and when **BOOT 1.02** appears and keep them pressed until **BOOTSTOP** appears. Press the two inner buttons. First **CLEANUP** and then **REBOOT** appears.

Now, the control runs until **BB STOP**, the memory is cleared, the Motion components are ready for operation and the PLC components in process are stopped.

5.3.2 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.



Desktop icon:

5.3.3 Creating a New Project

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

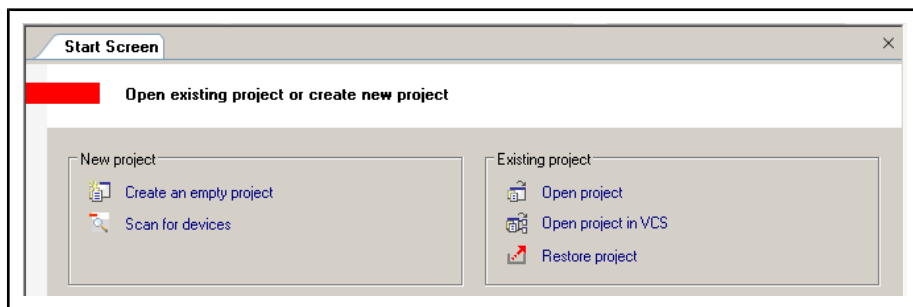


Fig.5-2: Creating an empty project

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.

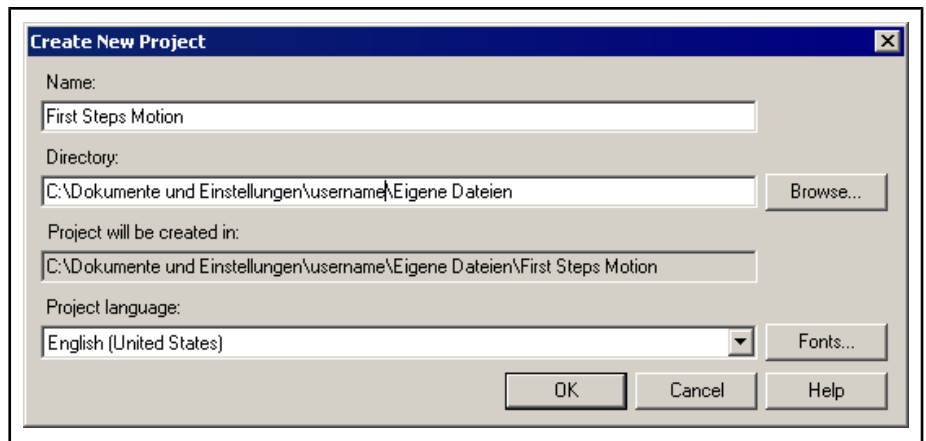


Fig. 5-3: Settings to save a new IndraWorks Project

Name	Specify a name for the project. Suggestion: "First Steps - Motion".
Directory	The field contains a storage location in the directory C:\Documents and Settings \username\MyDocuments by default. A different directory can also be specified.
Project is created in	Directory and project name are automatically created.
Project language	By selecting the project language, the language of the user texts is determined. The newly created project appears in Project Explorer with the previously defined project name, refer to the following figure.

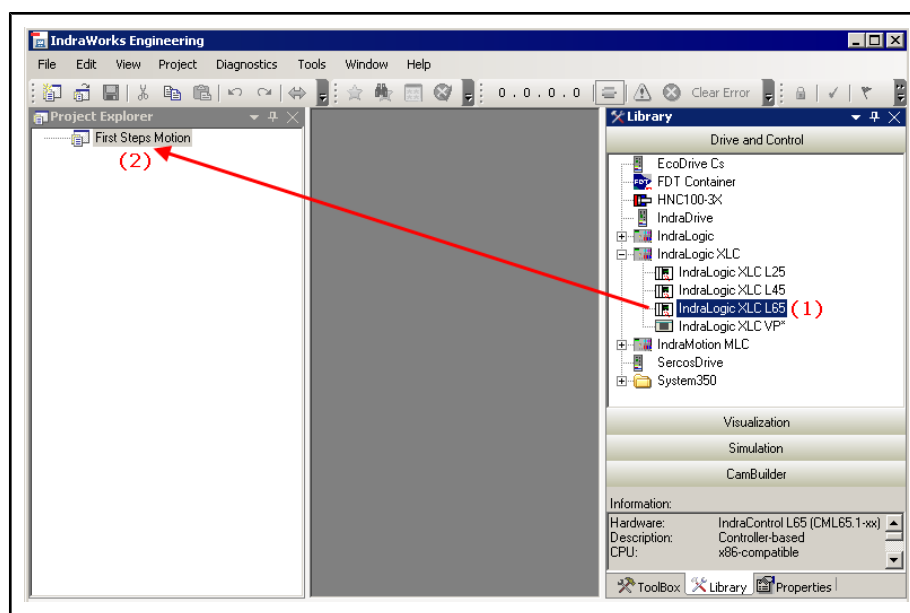


Fig. 5-4: First Steps - Motion project

5.3.4 Adding a New Device to a Project

Drag the desired device e.g. "IndraLogic XLC L65" from the "Drive and Control" library to the newly created project in the Project Explorer.

First Steps - Motion



(1) Device: IndraLogic XLC L65

(2) Project: First Steps - Motion

Fig.5-5: Adding a device

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

5.3.5 Specifying Basic Settings for the Device

Device Settings

Enter the general information on the device to be added.

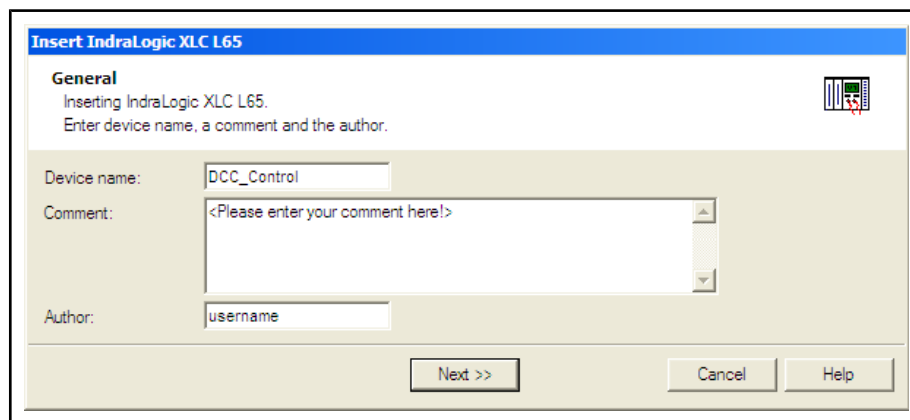


Fig.5-6: Device settings

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

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

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



These entries can also be changed later on.

Confirm the entries with "Next>>".

Communication Settings

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

Fig. 5-7: Configuration

- | | |
|-------------------------|--|
| Device type | The device type is predetermined by selecting the control. If there are several different configurations for one device type, a manual supplement is necessary (for instance: IndraLogic XLC Lx5 is available with SERCOS III bus (CMLx5.1-3P) and without SERCOS III bus (CMLx5.1-NP)). |
| 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. The default computer name (address) here is "localhost". This means that the locally installed gateway is addressed. This setting is not to be changed. |
| Connection test | <p>Click on the "Connection text" button to check whether the communication between computer and target control is successful. The test result is recorded in the "Connection test result" field.</p> <p>If the PLC communication test was successful, the IP address is shown.</p> <p>Any possible problems are signaled by flashing red or yellow icons in the "Connection Test Result" window.</p> <p>The following problems can occur:</p> <ul style="list-style-type: none"> • Output: "Communication to the control not successful." <p>Causes:</p> |

First Steps - Motion

- 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 change the IP address of the control (see Rexroth IndraMotion MLC Functional Description, Commissioning the Control).
- The network mask of the PC and/or control is set too small.
Solution: Check and correct network settings (see Rexroth IndraMotion MLC Functional Description, Commissioning the Control).

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

Causes:

- The control firmware is not current.
Solution: Update firmware (see Rexroth IndraMotion MLC Functional Description, Commissioning the Control).
- The selected control does not correspond to the real control type.

Secure online mode

If this option is checked, a prompt appears 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.

Function Module Settings

Depending on the device type, the "Device and function module settings" dialog may differ. In this project, the dialog of the "IndraLogic XLC L65" device is described as example.

Enter the desired device and function module settings.

Insert IndraLogic XLC L65

Interfaces
Please select which onboard components and function modules you want to use.

SERCOSIII configuration (X7E1/X7E2): SercosIII Master

Slot0 Configuration (X7P): Profibus DP Master

Slot1 Configuration (X7E3/X7E4): Nicht verwendet

Ethernet configuration (X7E5): Not used

Function modules: Not used

Not used

Not used

Not used

Fig. 5-8: Function module settings

SERCOS III configuration

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

First Steps - Motion

Slot0 configuration	Current PROFIBUS configuration that can be selected when adding a device. "PROFIBUS DP Master", "PROFIBUS DP Slave" or "Not Used" can be selected.
Slot1 configuration	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	As this interface is required as Engineering interface (connection directly to your PC or via the operating network), the Ethernet/IP Adapter may not be enabled.
Function modules 1-4	Up to four function modules can be connected to the "IndraLogic XLC L65". It is possible to select from the respective list if required. Confirm the entries with "Finish". The newly created device appears in the Project Explorer.

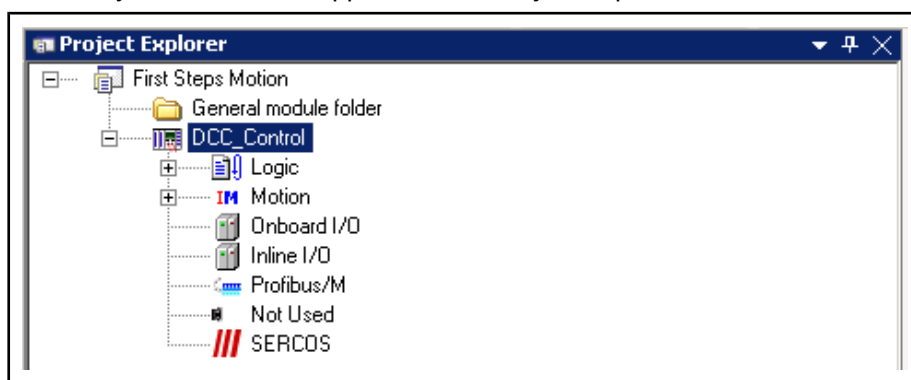


Fig.5-9: Project Explorer: "First Steps - Motion" project with an IndraLogic XLC L65 control with SERCOS III bus



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

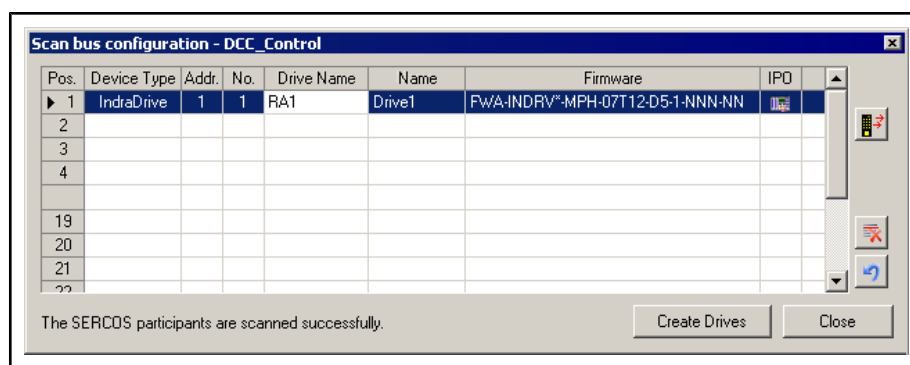
5.3.6 Adding SERCOS III Drives (Real Axes) to the Project

The simplest way to determine the data for all the connected drives is to scan the SERCOS III 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:

First Steps - Motion



Pos. Topology address (order in the ring)
 Device type Drive family
 Addr. SERCOS address
 No. Axis number
 Drive Name Default drive_n, renamed
 Name Name of the axis / I/O device
 Firmware Firmware ID
 IPO Interpolation in the control or drive, in this case in the control
Fig.5-10: List of the SERCOS drives; drive name changed to RA1



Interpolation in the control  or drive  :
 Switching by double-clicking on the icon.

Click on "Create Drives".

The scanned bus configuration is transferred to the Project Explorer. The drive is reflected as a "Real Axis" in the Motion node.

The axis number "1" can be used instead of the axis name "RA1".

The project contains the "DCC_Control" device of type "IndraLogic XLC L65" and the real axis "RA1" (folder **Motion** ► **Real Axes**) or the drive "RA1" (folder **SERCOS**):

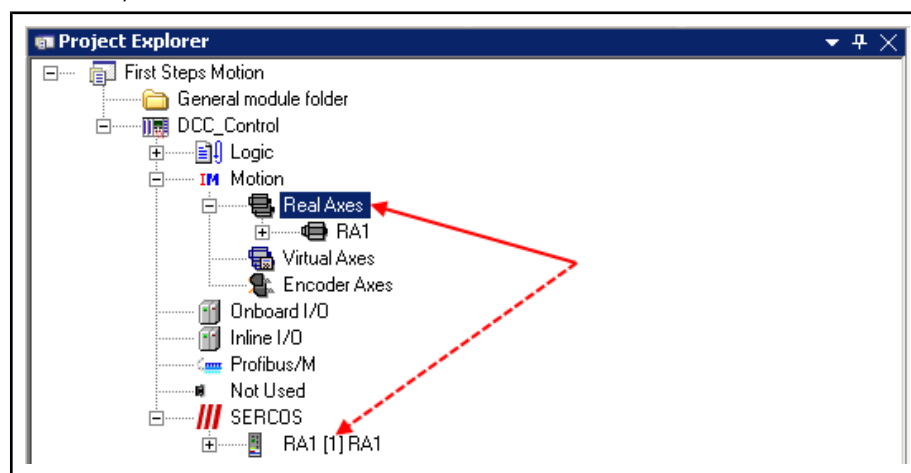


Fig.5-11: Drive/real axis "RA1" is added to the project

5.3.7 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**.

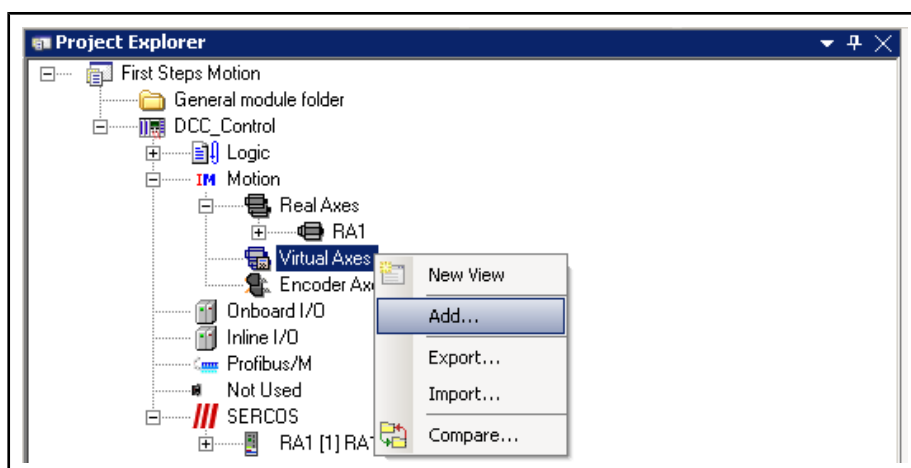


Fig.5-12: Declaring a virtual axis

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

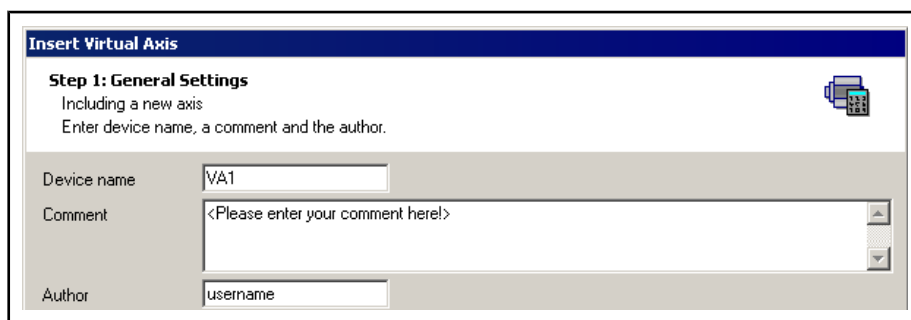


Fig.5-13: "General Settings" dialog for a virtual axis

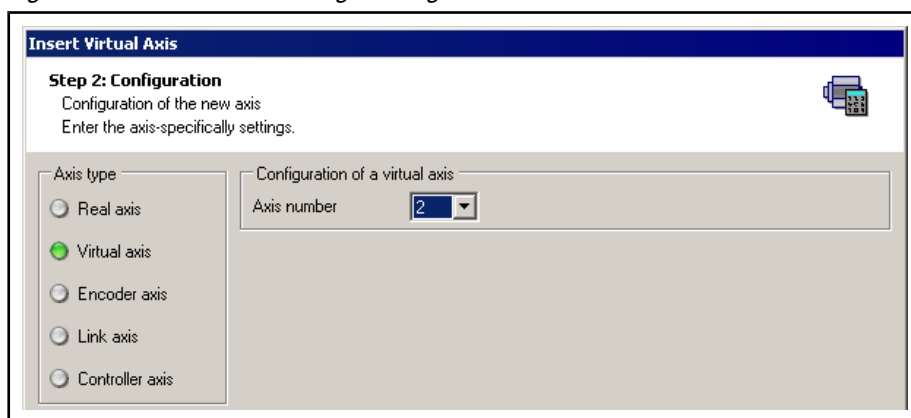


Fig.5-14: "Configuration" dialog for a virtual axis

The axis number "2" can be used instead of the axis name "VA1".
Click on "Finish".

5.3.8 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 (DCC_Control) and click on the icon of the project toolbar or on the menu item **Project ► Switch Device Online**:

First Steps - Motion

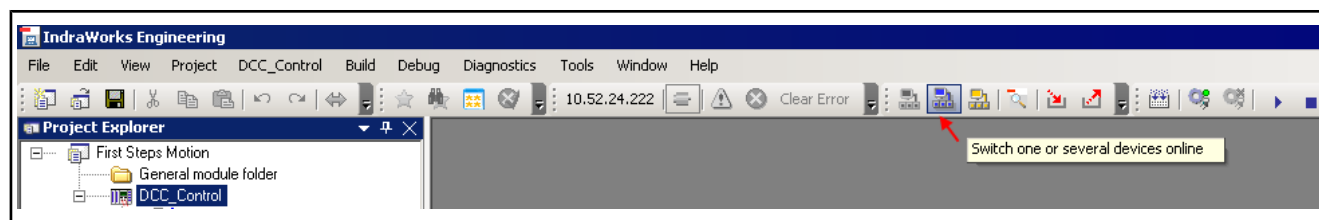
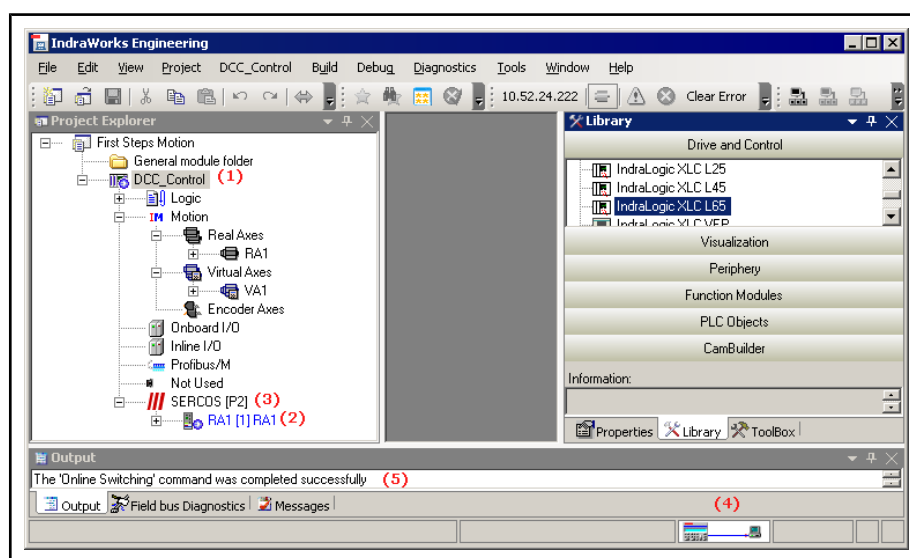


Fig.5-15: Loading Motion data to the control

It is informed about differences between the project on the control and the own project. After accepting the differences, overwriting is executed. The control display changes from **BB STOP** to **P2 STOP** in a series of steps.



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



- (1) Active control (online)
- (2) Active drive (online)
- (3) Communication at the "P2" level, parameterization mode
- (4) Online information
- (5) Status bar

Fig.5-16: Download completed successfully (status bar)



The "Online" information (4) 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. Context menu **RA1Parameters** ► **Load basic parameters...**

The control is then switched from "Parameterization mode (P2)" to "Operating mode (BB)" (context menu of SERCOS node **SERCOS state** ► **Phase P4**).



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 **DCC_Control** ► **Motion** ► **Real Axis** ► **RA1** ► **Establish Dimensional Reference** ► **Dimensional reference - motor encoder** "Set absolute position".

The drive is now in "Ab", Drive ready, status.

The control, Motion components in "BB", are ready for operation, the PLC components are in "STOP"

5.4 Programming the Motion Sequence in Function Block Diagram (FBD)

5.4.1 Programming a Motion Sequence in FBD, General Information

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

When creating a project with the DDC_Control control, the real axis RA1 and the virtual axis VA1, the first actions to program the motion sequence were started below the "Logic" folder:

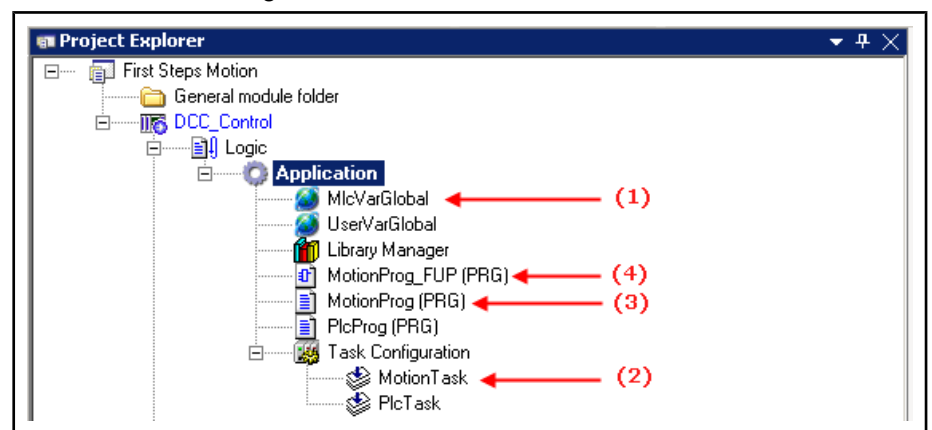


Fig. 5-17: Project Explorer after expanding the "Logic" node

The following programs were created:

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

First Steps - Motion

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 they are not within this part, the control is stopped.

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

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

Global variables/global constants were created 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
```



We use the MotionTask for "First Steps - Motion".

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

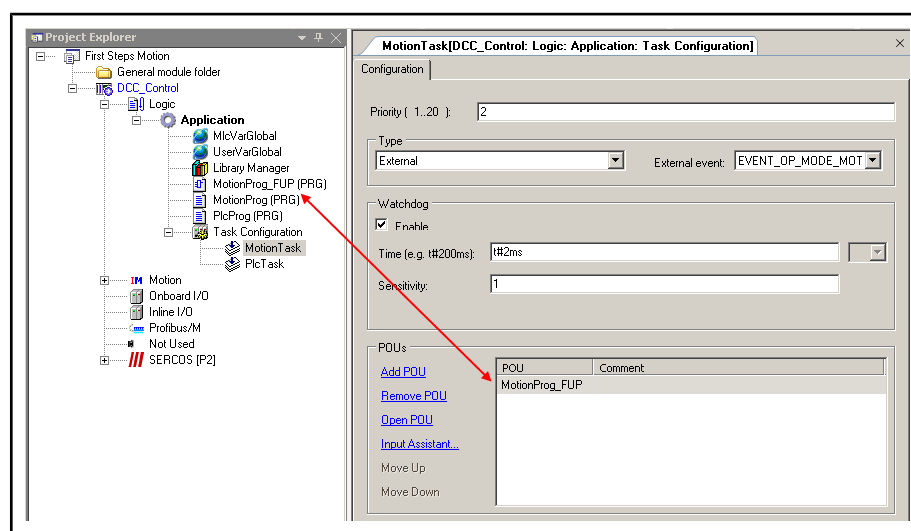


Fig.5-18: Delete call: MotionProg. Add call: MotionProg_FBD

5.4.2 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 119](#).

Similarly to [Project Goal First Steps - Motion, page 83](#), 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 described below:

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

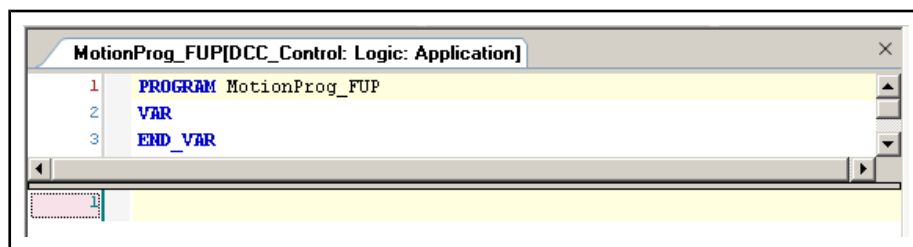


Fig.5-19: New program in FBD (Function Block Diagram)

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

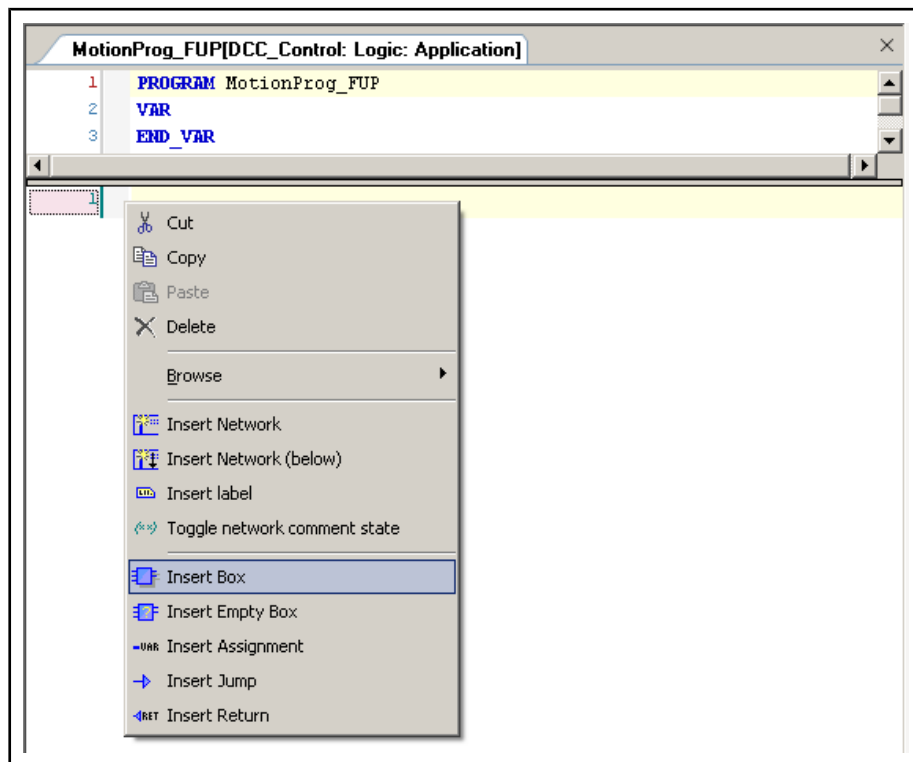


Fig.5-20: Generating a new function block

First Steps - Motion

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

Function Blocks ► ML_PLCOPEN ► POUs ► Motion ► MC_MoveVelocity.

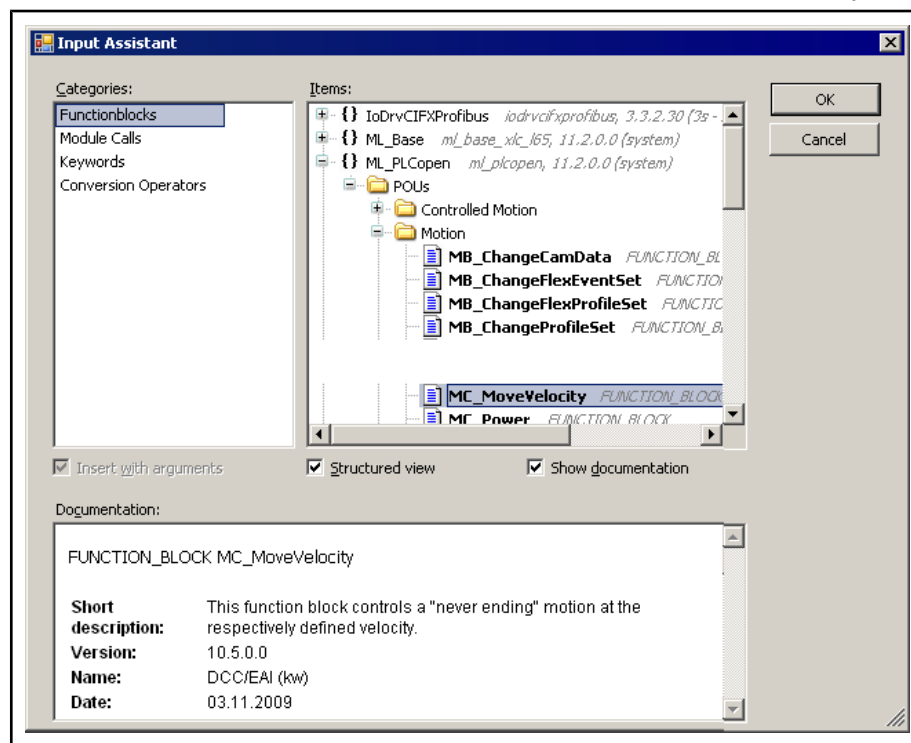


Fig. 5-21: Selection window for function block types

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":

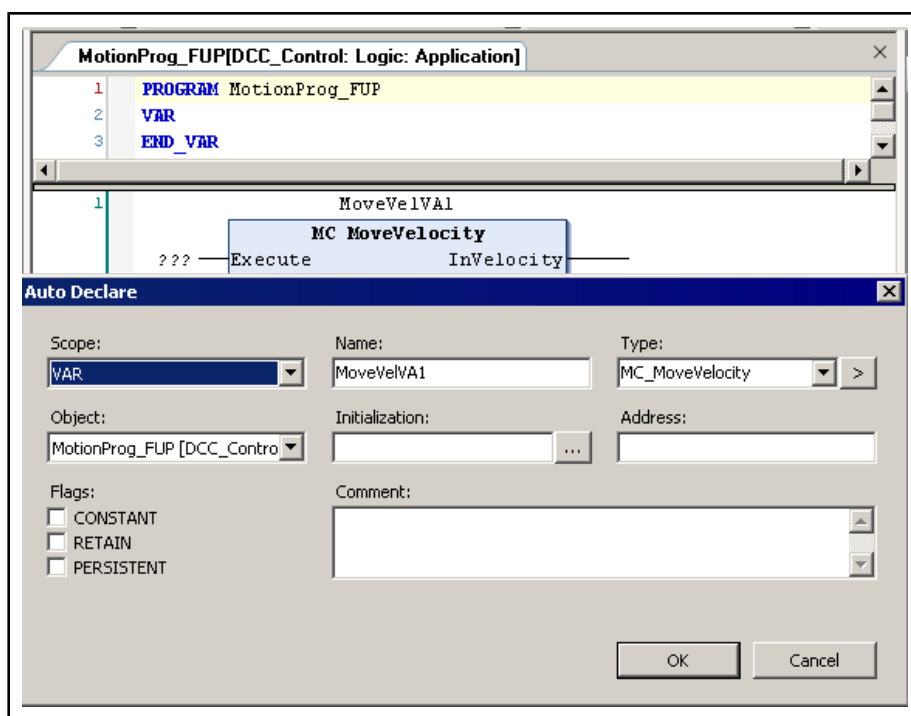


Fig. 5-22: Declaration of the instance name of an FB, type "MC_MoveVelocity" using the "Variable Declaration" window

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

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 press <Enter>
- Enter REAL as type and 200 (rpm) as initial value. Press <Enter>.

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**.

First Steps - Motion

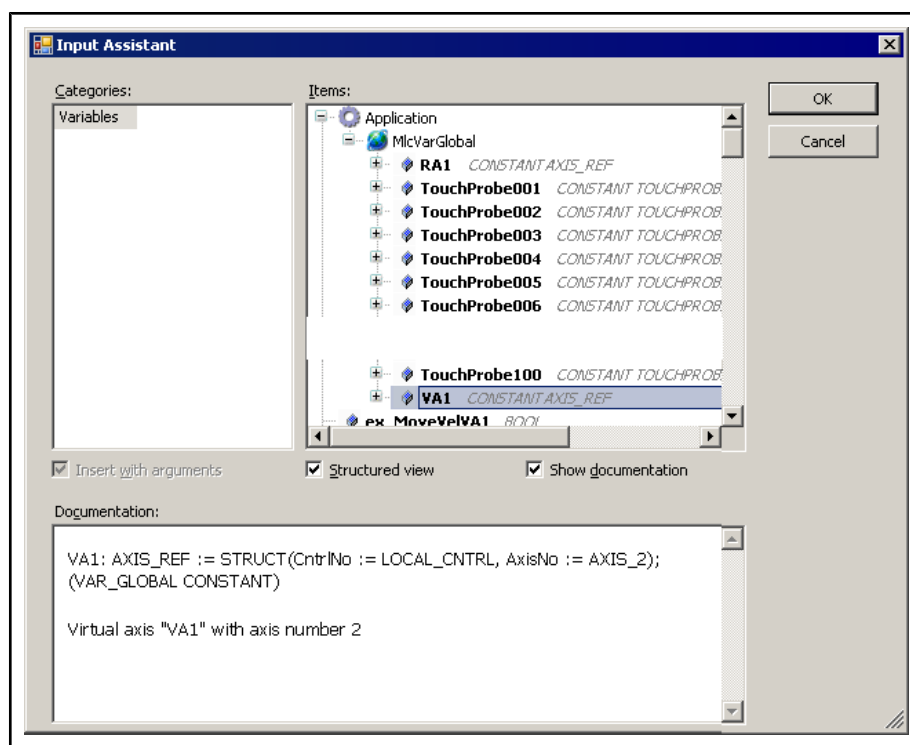


Fig.5-23: Selecting the axis instance when assigning a function block of the type "MC_MoveVelocity"

- Confirm by clicking on "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

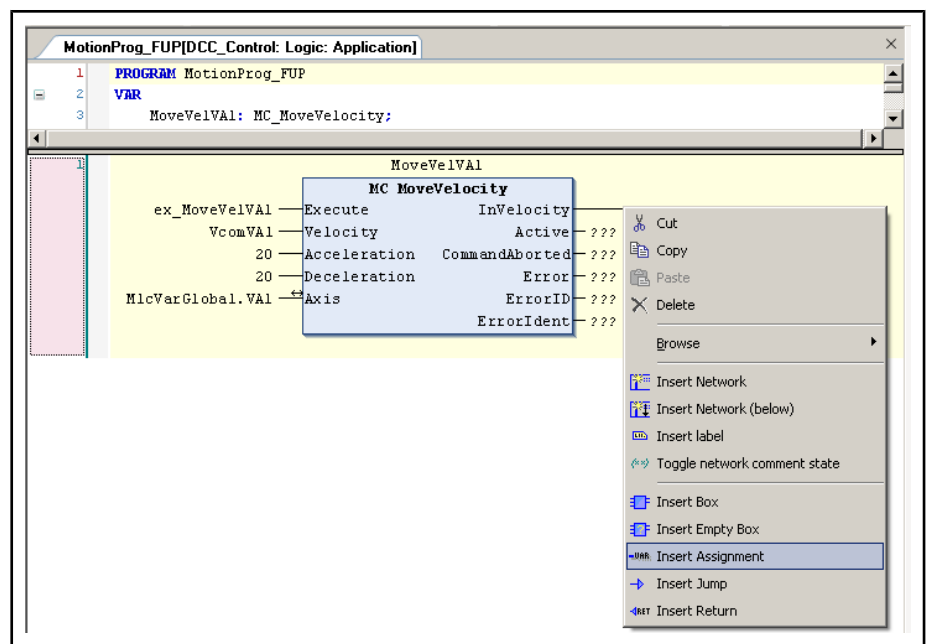


Fig. 5-24: Assignment at the function block output

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

- Highlight the 3 question marks **???** at the output "InVelocity".
- Enter the name "in_MoveVelVA1", then <Enter>.
- The open window "Variable Declaration" can be confirmed immediately with <Enter>.
- Assign "ca_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.

First Steps - Motion

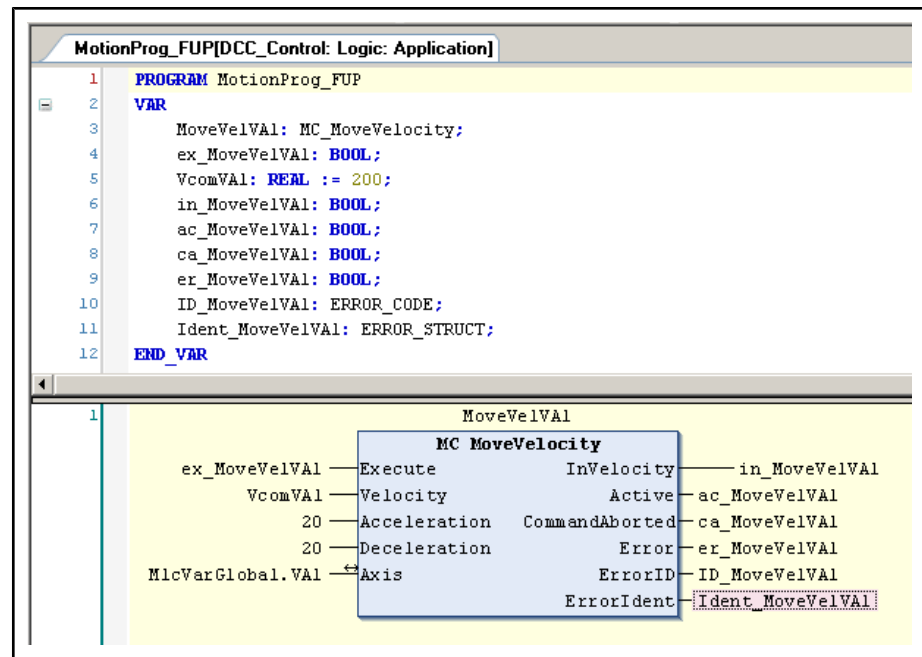


Fig.5-25: PLCopen FB, "MC_MoveVelocity" type is inserted and assigned



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

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

First Steps - Motion

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 **???**, press <Enter>, select type "REAL" in the declaration window, press <Enter>.
- Go to the left **???** and click on **...** using the input assistance to navigate to **Variables ► ML_Base ► Global Variable List ► GVL_Base ► AxisData**:

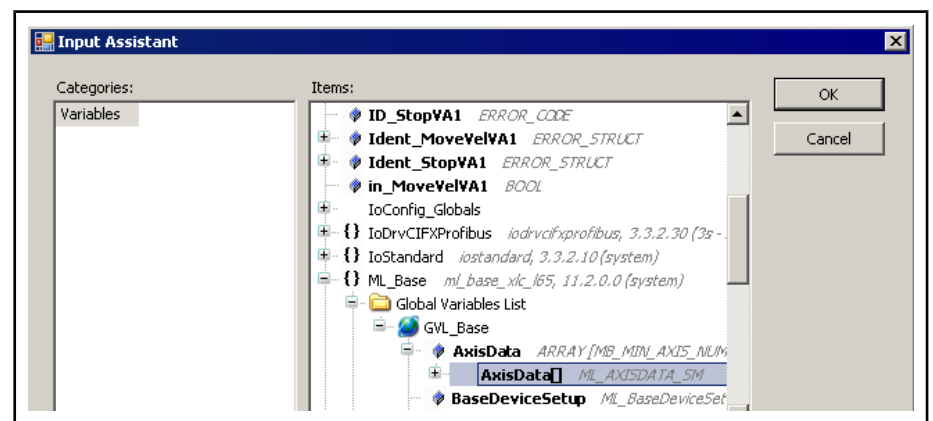


Fig. 5-26: Actual velocity value VA1 from the global variable list of the library ML_Base.library

Further input is supported by the Intellisense function. This means that after entering "." for "GVL_Base.AxisData[2].", all available elements are open and can be selected.

Confirm "rActualVelocity_i" with <Enter>.

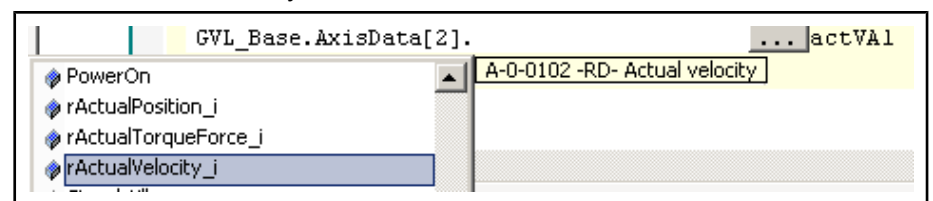


Fig. 5-27: Selecting the variables "ActualVelocity"

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

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

First Steps - Motion

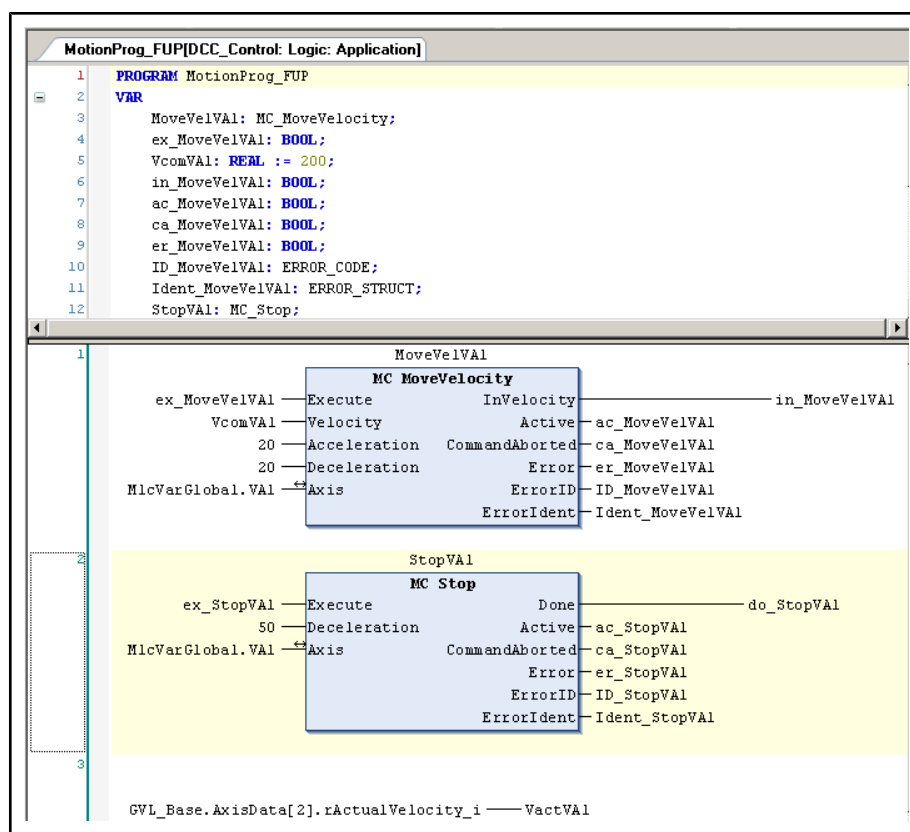


Fig.5-28: Implementing the virtual axis VA1

5.4.3 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.

The compilation can be accessed via

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

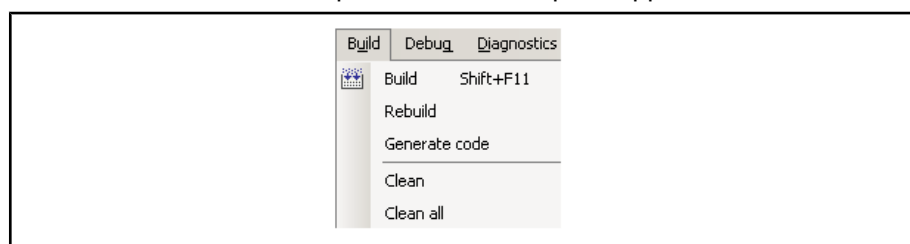


Fig.5-29: Build application

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

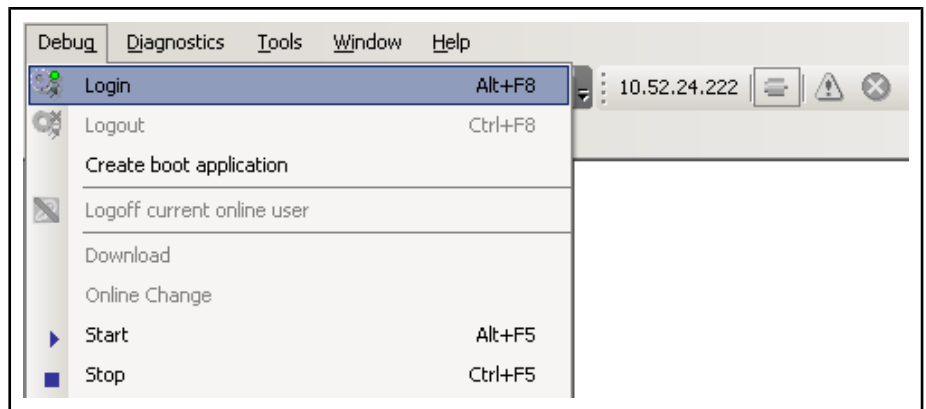


Fig.5-30: Transferring compiled data to the control

A prompt asking for agreement to download appears 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.

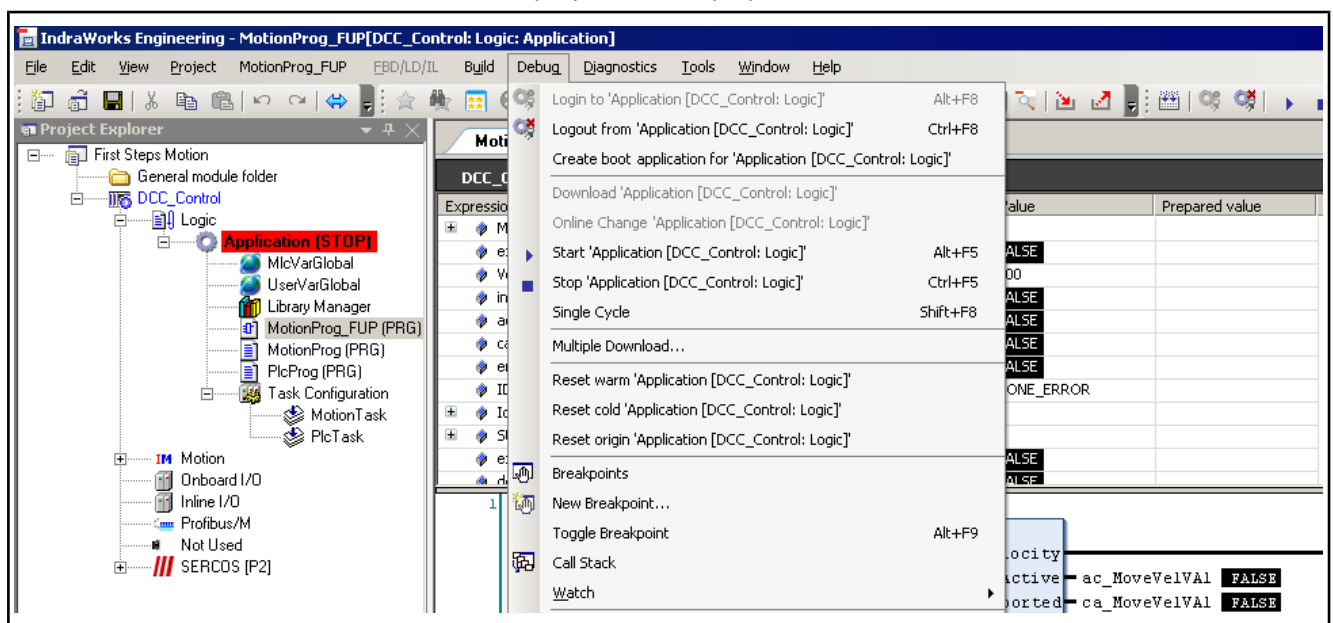


Fig.5-31: Starting the program processing in the control (1)

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

First Steps - Motion

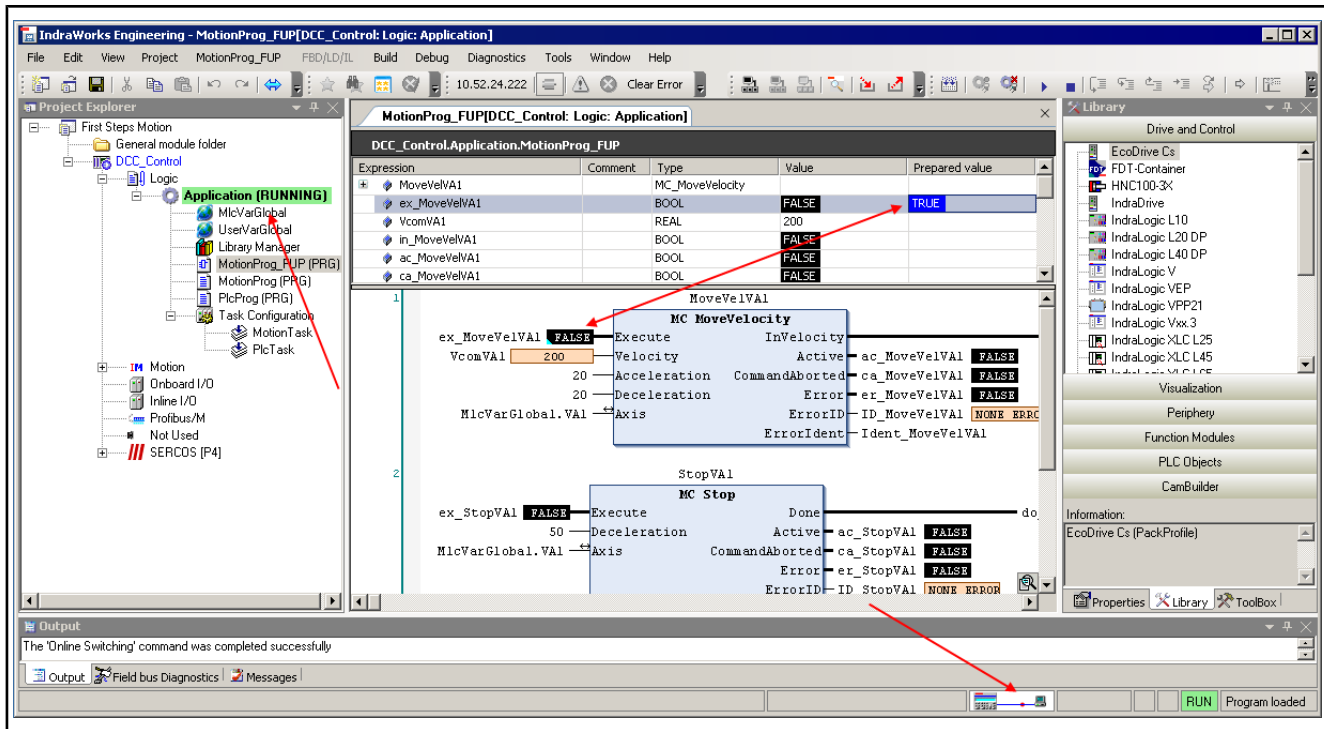


Fig.5-32: Starting the program processing in the control (2)

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).

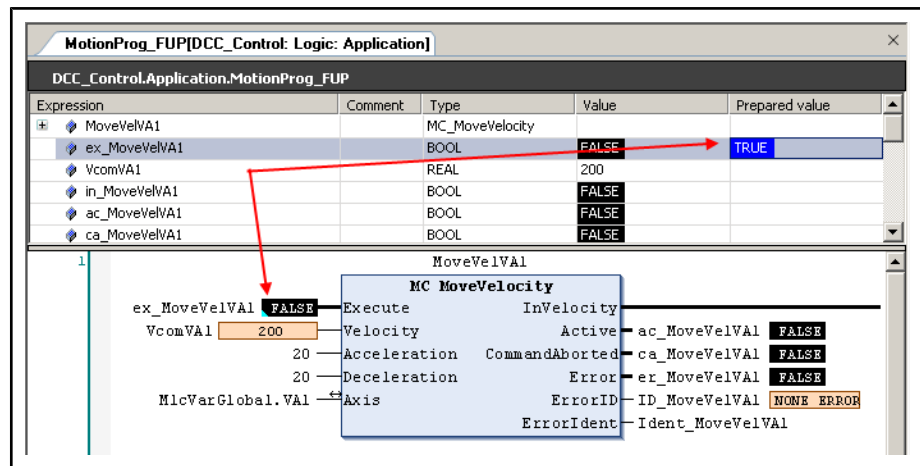


Fig.5-33: Preparation of "Write values"

In the control, <ex_MoveVelVA1> can be set to <TRUE> via **Debug ► Write Values "Ctrl"+"F7"**.



- "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.

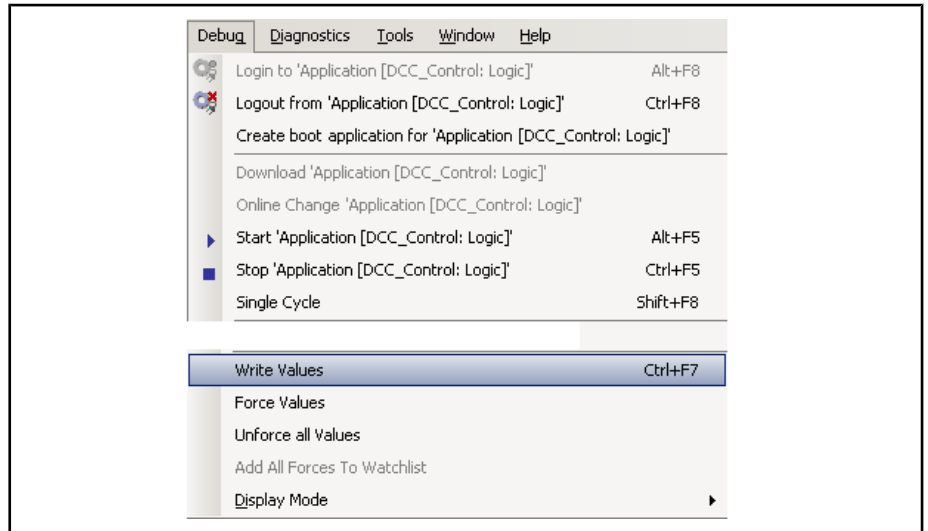


Fig.5-34: Write values

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.

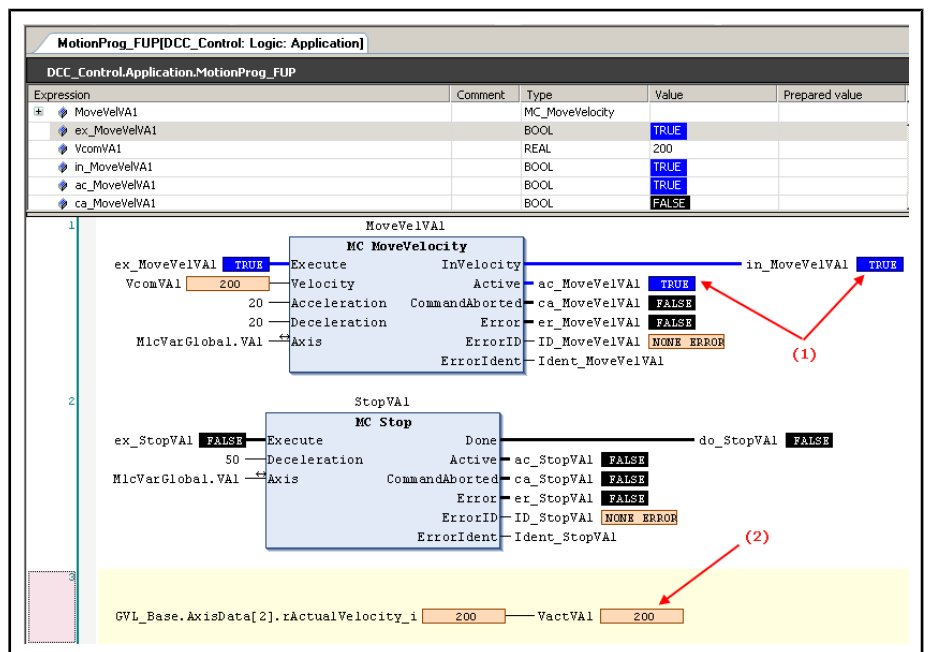


Fig.5-35: After "Write Values" at the "Execute" input

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.

First Steps - Motion

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

IndraLogic provides a Trace tool to trace motion processes. It can be used to record

- 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.

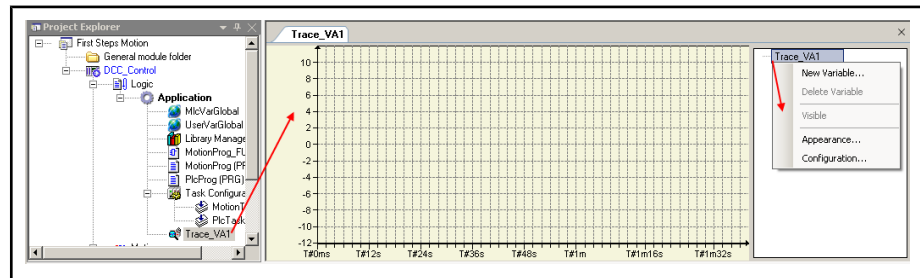


Fig.5-36: Dialog: New Trace object "Trace_VA1" with context menu

The following can be done with the Trace object via the context menu on the right side of the page:

- Configuration...
- Appearance... and
- New Variable...

Configuration... The new Trace object must firstly be configured:

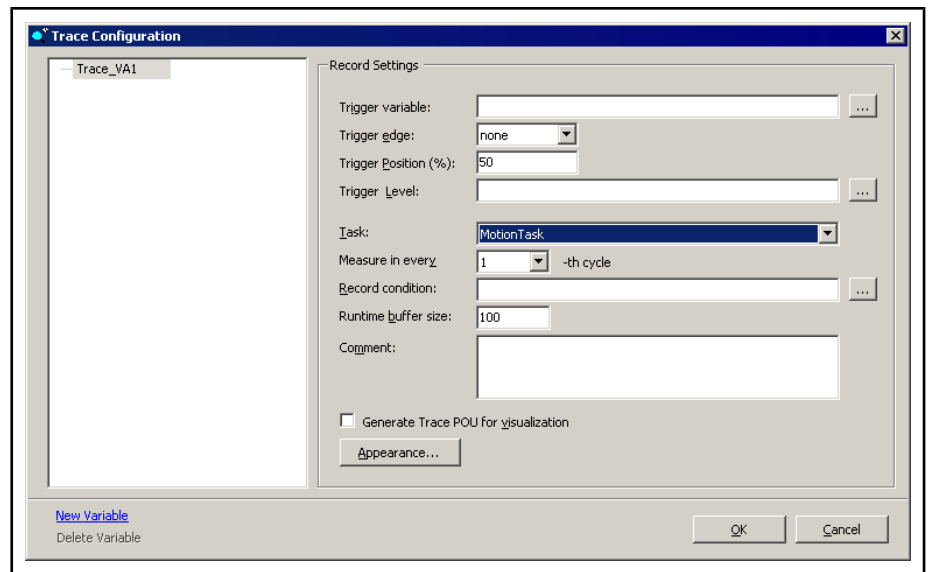


Fig. 5-37: Dialog: New Trace object "Trace_VA1", minimum configuration

The required settings are

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

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

Measure in every ... -th cycle: 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.

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.

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.

First Steps - Motion

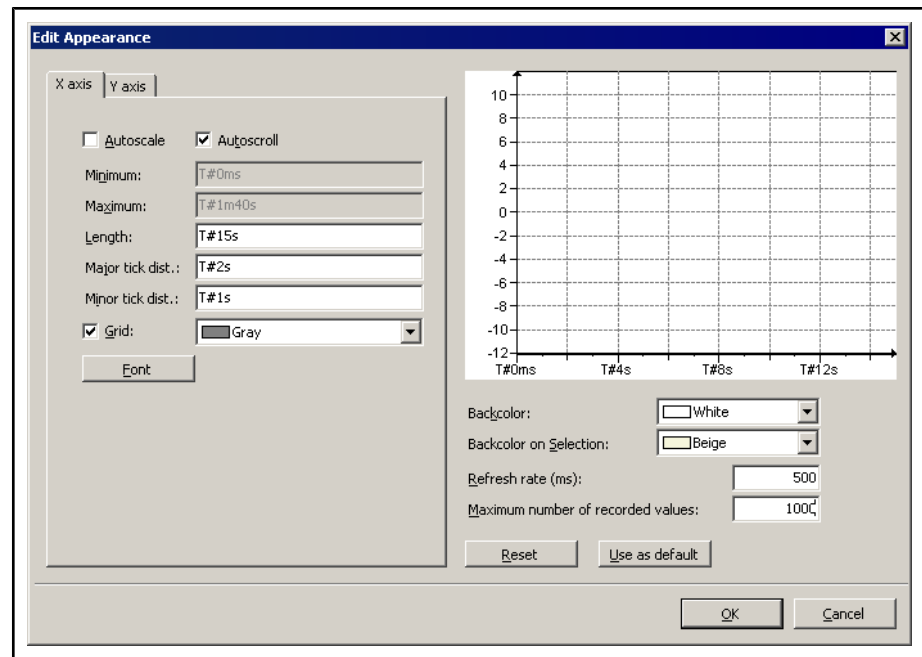


Fig.5-38: Dialog: Display settings (appearance)

Select **Autoscroll**, positioning a **Length** 15s window over the curve. The **Scale division** is set out in a 2s grid with a **scale division** of 1s.



The process under scrutiny should not be longer than 15s.

New Variables...

Now the variables required must be named and their appearance adapted:

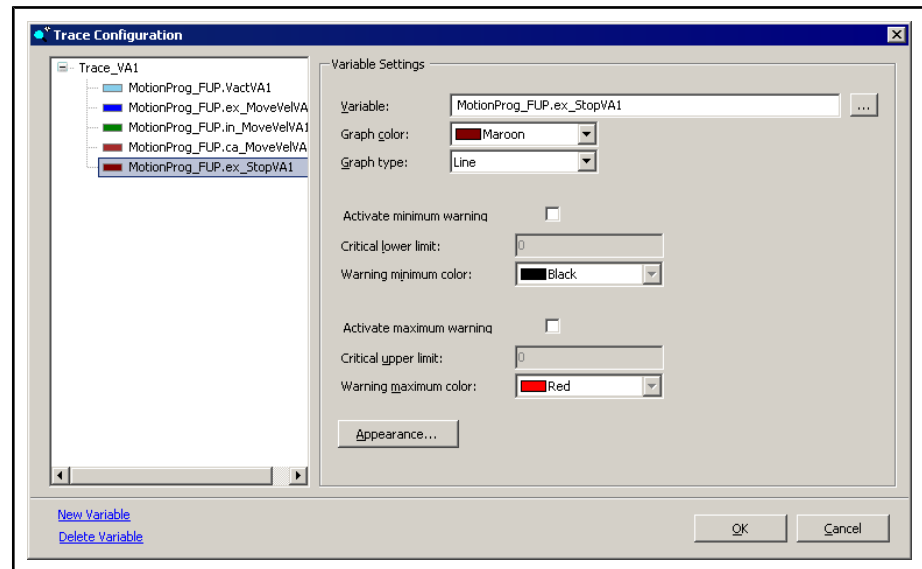


Fig.5-39: Dialog: Creating new variables and adapting appearance

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 appears by right-clicking into the left section of the Trace window.

First Steps - Motion

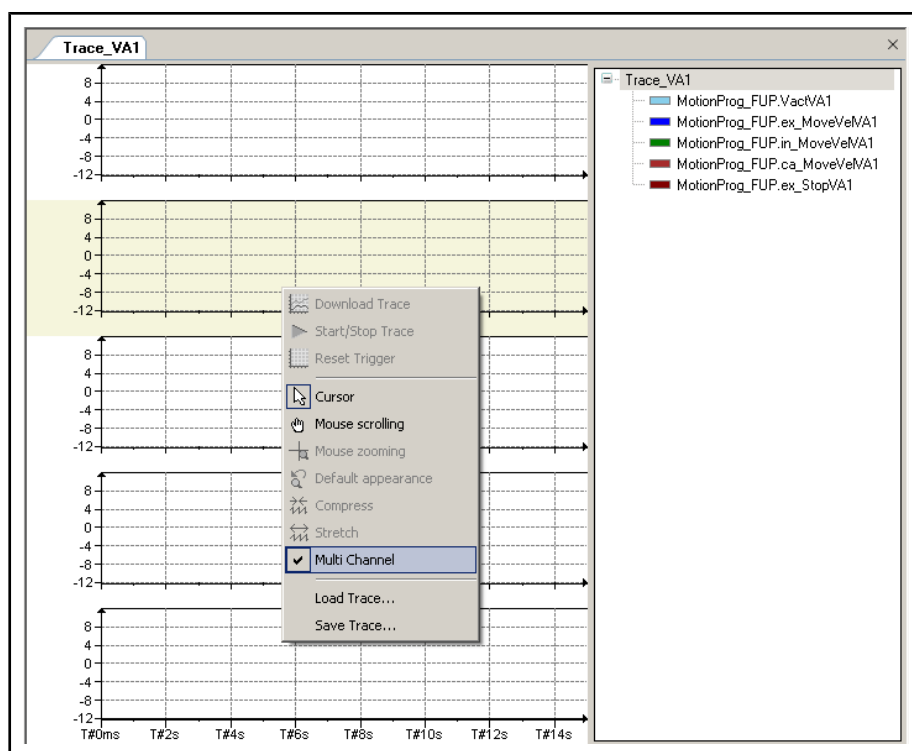


Fig. 5-40: Menu to control the Trace recording

Login to Trace: Before a configured trace record on the control can be enabled for the application running there, the "Trace application" must be loaded there explicitly

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 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, for example.

- Performing the Trace:**
1. Login to application
Connect to the control.

First Steps - Motion

2. Login to Trace

When the application is running, the data required is transmitted to the control via the "Login to 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 104,.

- 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
<Alt>+<left/right> arrow keys	Horizontal scrolling of X-axis in all diagrams
<up/down >arrow keys	Horizontal scrolling of only the selected diagram
<+/-> keys	Simultaneous zooming of the coordinate system along the X- and Y-axis of all diagrams
<Shift>+<+/-> keys	Horizontal scrolling of all diagrams along the X-axis
<Ctrl>+<+/-> keys	Vertical scrolling of all diagrams along the Y-axis
<left/right >arrow keys	Left/right scrolling of cursor

Fig.5-41: Trace Editor shortcuts when Multi Channel is enabled

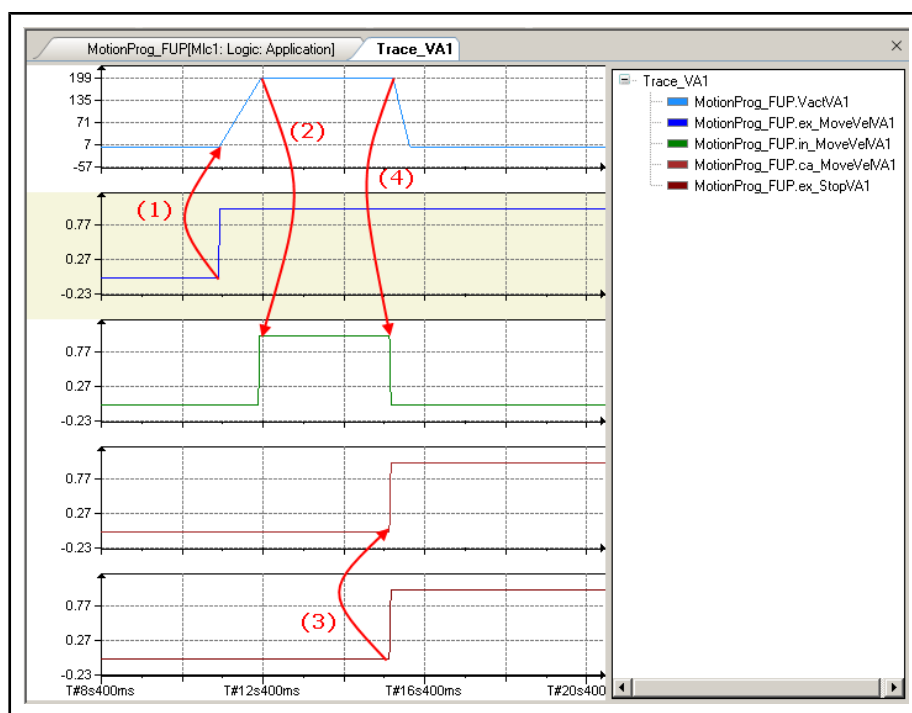


Fig.5-42: Trace recording for the virtual axis VA1

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 characteristics, acceleration at 20 rad/s² and deceleration at 50 rad/s²).

5.4.5 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, page 119](#).

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

First Steps - Motion

"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, click on network 4 with the right mouse button,

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, click on network 5 with the right mouse button,

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, click on network 6 with the right mouse button,

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.

The variable is provided with the name

GVL_Base.AxisData[1].rActualVelocity_i,

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

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

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

First Steps - Motion

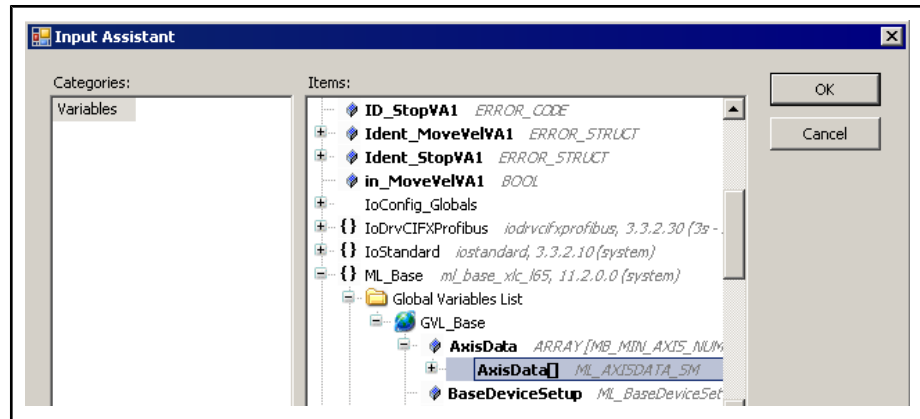


Fig.5-43: Actual velocity value VA1 from the global variable list of the library *ML_Base.library*

Further input is supported by the Intellisense function. This means that after entering "." at "GVL_Base.AxisData[1]." all available elements scroll up and can be selected.

Confirm "rActualVelocity_i" with <Enter>.

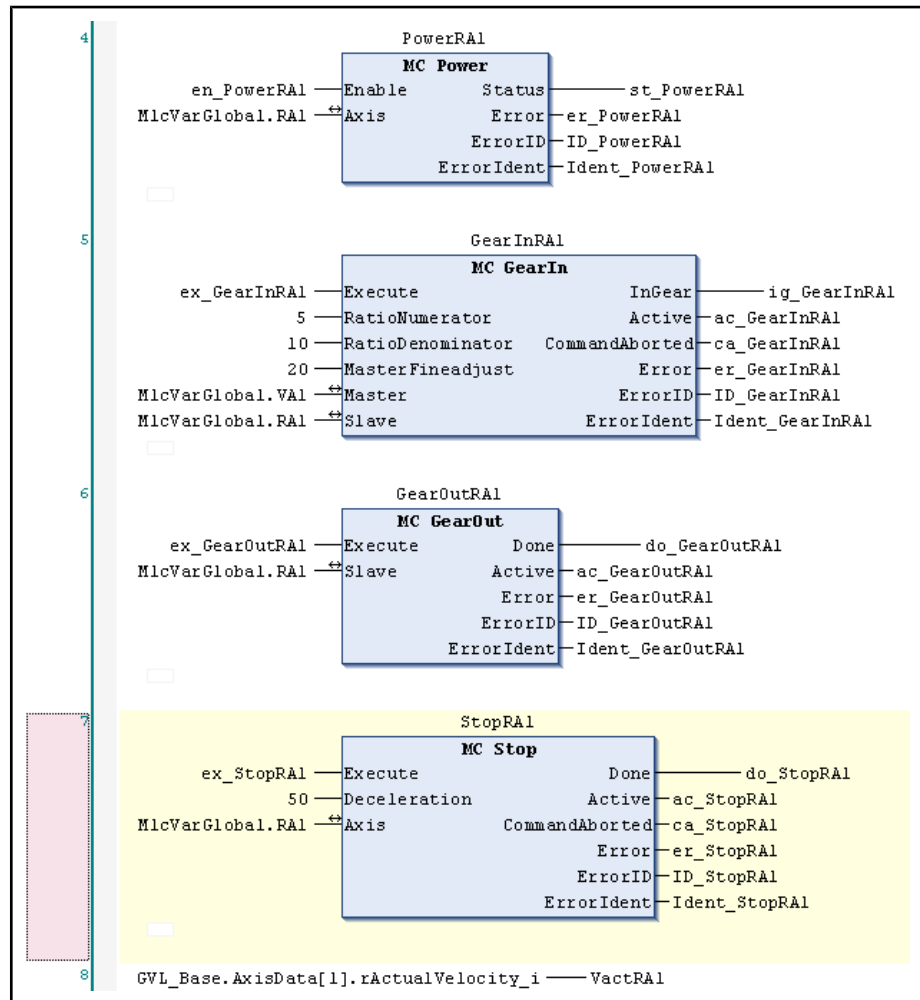


Fig.5-44: Implementing the real axis

5.4.6 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 102.

The new project overwrites the old after sending a prompt.

5.4.7 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.
3. Follow 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.
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).

First Steps - Motion

11. Stop the real axis
Enable "ex_StopRA1" (MC_Stop).
The real axis is decelerated to VactRA1= 0 rpm. After it 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 for the real axis
Disable "en_PowerRA1" (MC_Power) by "Write Values".



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

5.4.8 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 106](#), 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:

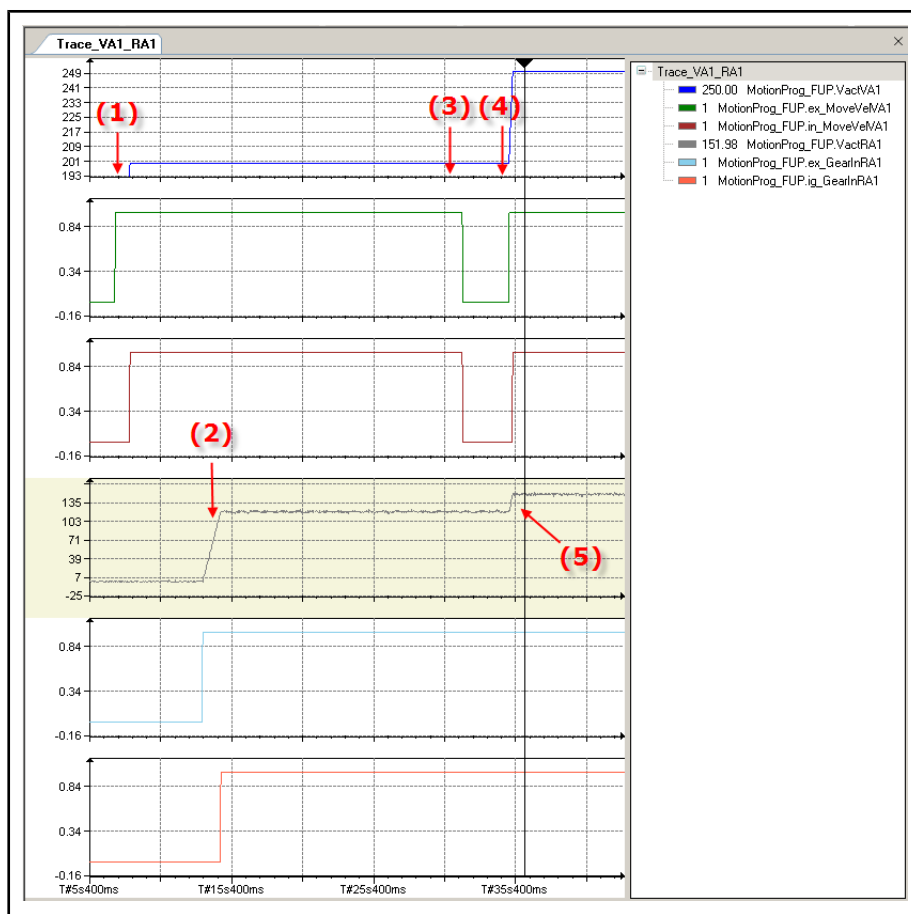


Fig. 5-45: Trace recording for both axes

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,
- ③ - Enable command speed of virtual axis, "ex_MoveVelVA1" FALSE ⇒ TRUE,
- ⑤ - Jump response on real axis.

5.5 Programming the Motion Sequence in Structured Text (ST)

5.5.1 Programming a Motion Sequence in ST, General Information

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

When creating a project with the DDC_Control control, the real axis RA1 and the virtual axis VA1, the first actions to program the motion sequence were started below the "Logic" folder:

First Steps - Motion

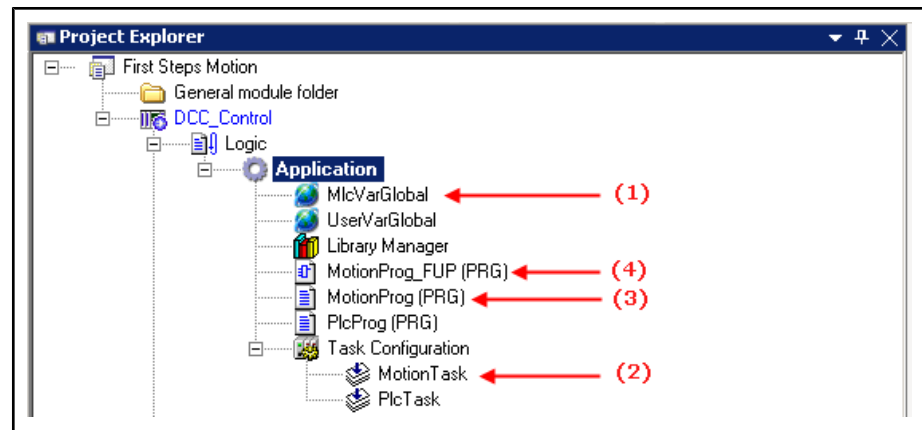


Fig.5-46: Project Explorer after expanding the "Logic" node

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 they are not within this part, the control is stopped.
- **PlcTask**, (low priority cyclic task) for PlcProg, not used.
In the remaining time, the program is processed at least piece by piece, that means during several cycles if required.

Global variables/global constants were created 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
```



We use the MotionTask for "First Steps - Motion".

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

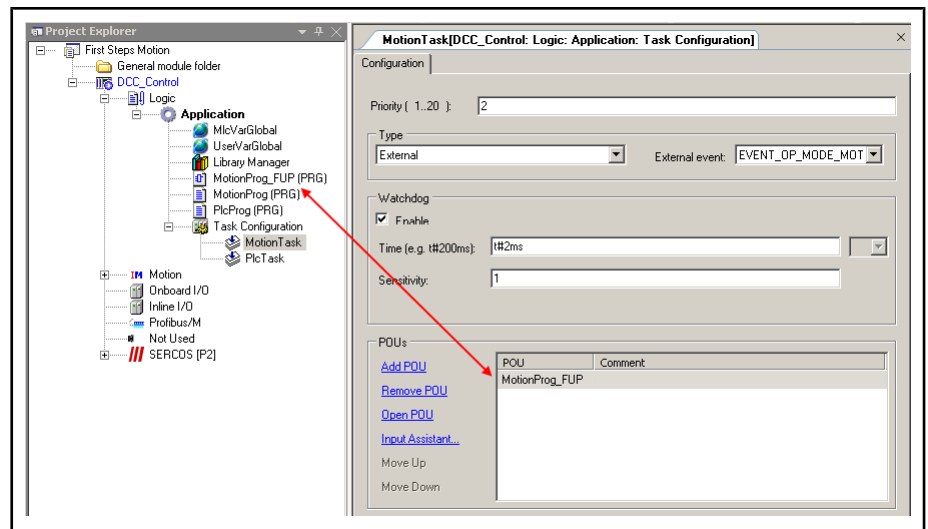


Fig.5-47: Delete call: MotionProg_FBD; add call: MotionProg

5.5.2 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 83](#), 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 described below:

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

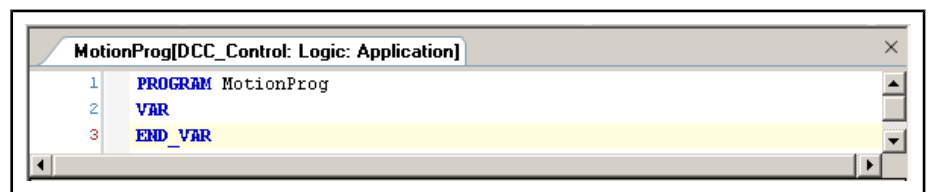


Fig.5-48: New program in ST (Structured Text), after dummy line deletion

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.

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).

First Steps - Motion

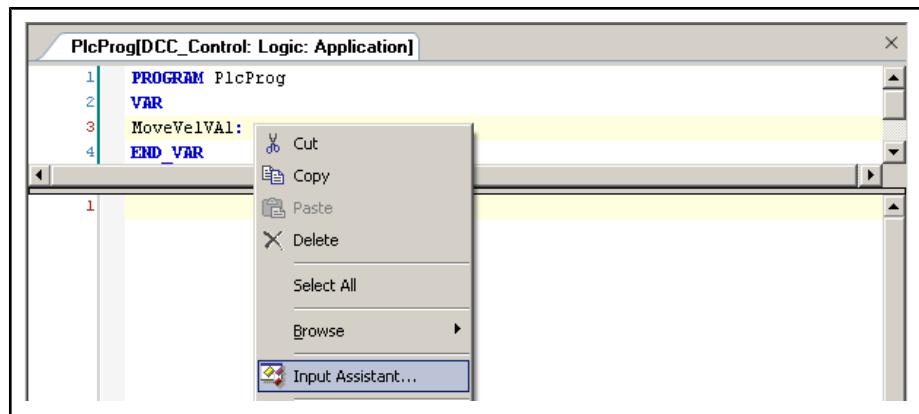


Fig.5-49: Declaration of the instance MoveVelVA1, MC_MoveVelocity type

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

User-defined Types ► ML_PLCOpen ► POU's ► Motion ► MC_MoveVelocity.

Select the option "Insert with arguments".

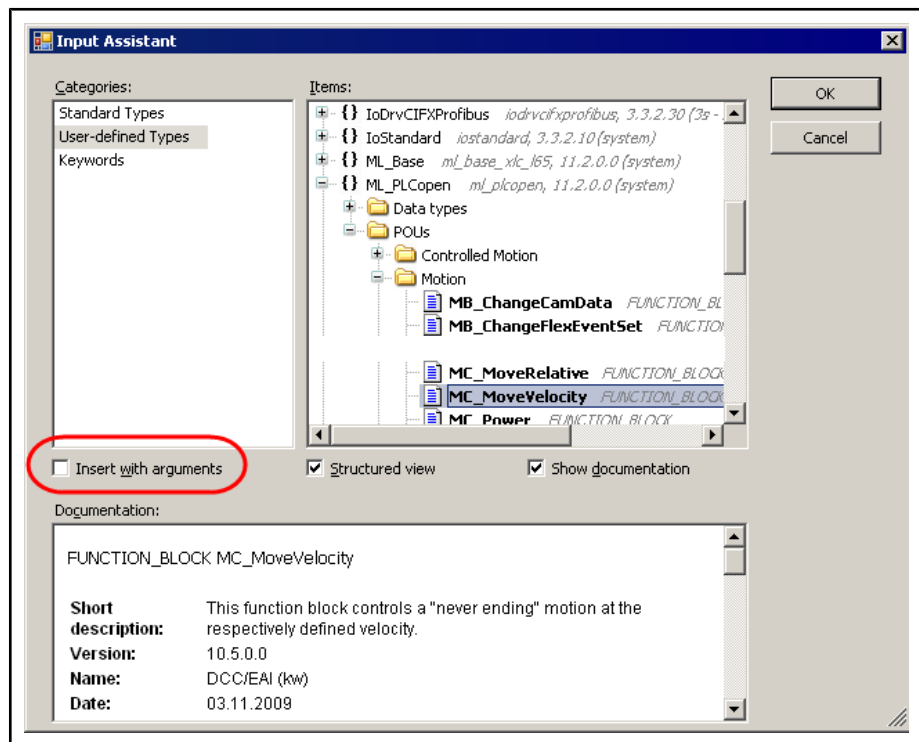


Fig.5-50: Selection window "Function Blocks"

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

Go to line 1 in the implementation section of the Editor and click on <F2>, Input Assistant.

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

Select the option "Insert with arguments".

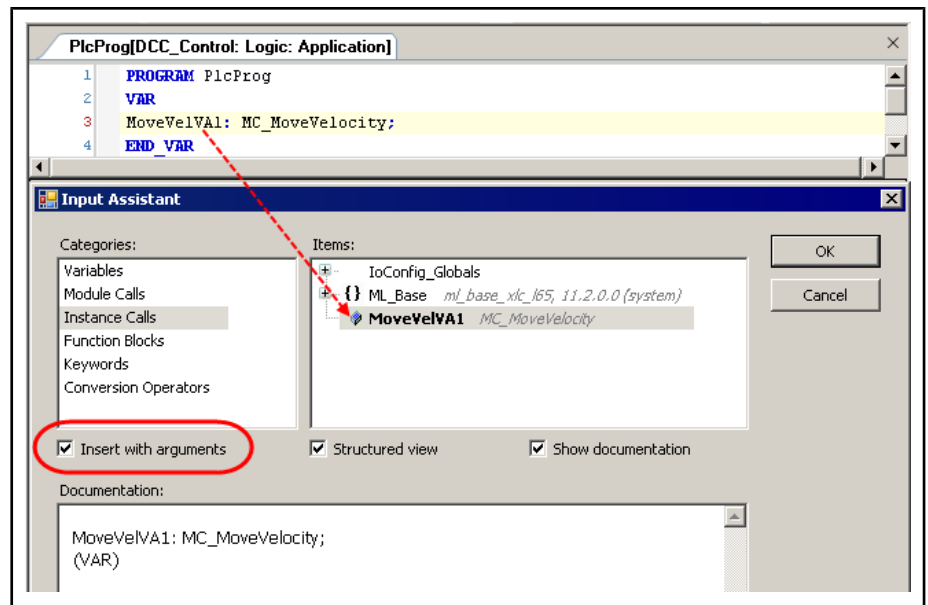


Fig.5-51: Transfer of the instance into the implementation

Confirm the insertion of the instance with "OK".

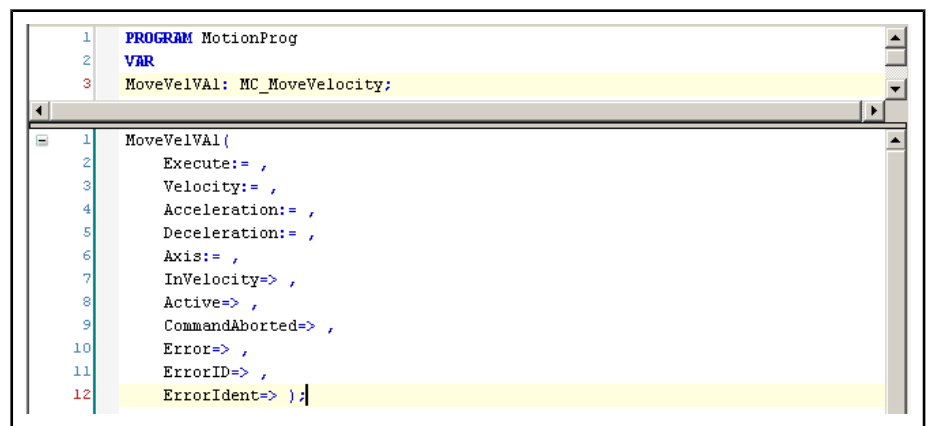


Fig.5-52: Broad design of a function block instance

Assign the FB inputs.

Begin with the input "Execute":

- Enter the name "ex_MoveVelVA1" and complete with "Enter".
- Confirm the "Variable Declaration" window that opens immediately with "Enter".

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

- Enter the name "VcomVA1" and confirm with "Enter".
- Enter "REAL" as type and 200 (rpm) as initial value and confirm with "Enter".

For the "Acceleration" inputs, 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:

- Select the adjacent Input Assistant by right-clicking on it:

First Steps - Motion

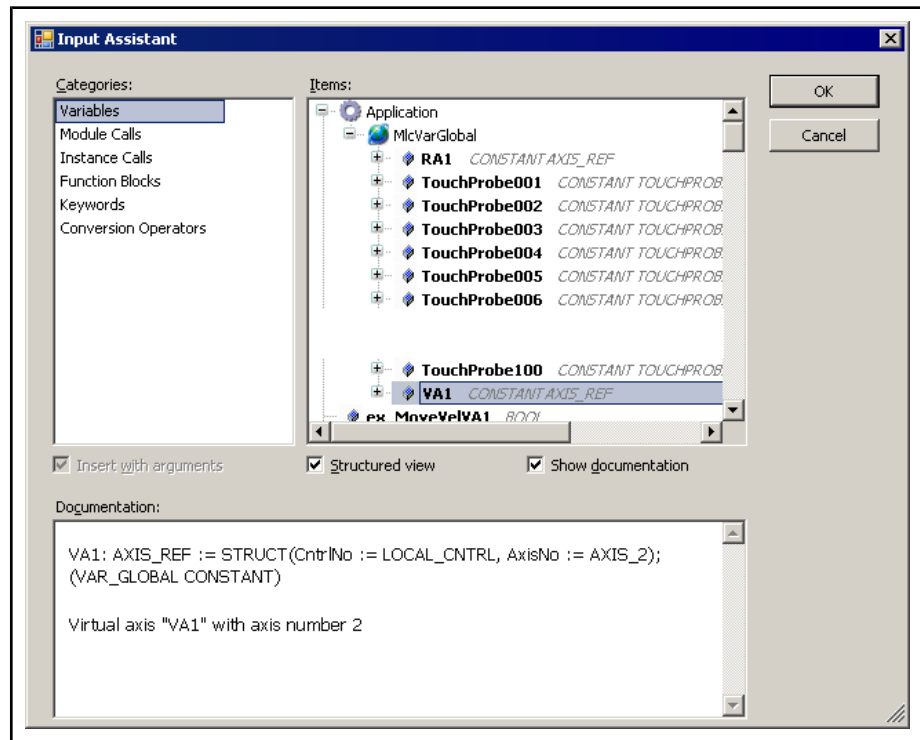
Select **Variables** ► **Application** ► **MlcVarGlobal** ► **VA1**.

Fig.5-53: Selecting the axis instance when assigning a function block of the type "MC_MoveVelocity"

- Confirm by clicking on "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 FB as usual using the "iv_MoveVelVA1" variable, type BOOL:
- Enter the name "iv_MoveVelVA1", then press <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  MoveVelVA1: MC_MoveVelocity;
4  ex_MoveVelVA1: BOOL;
5  VcomVA1: REAL := 200;
6  iv_MoveVelVA1: BOOL;
7  ac_MoveVelVA1: BOOL;
8  ca_MoveVelVA1: BOOL;
9  er_MoveVelVA1: BOOL;
10 ID_MoveVelVA1: ERROR_CODE;
11 Ident_MoveVelVA1: ERROR_STRUCT;
12 END VAR

1 MoveVelVA1(
2   Execute:= ex_MoveVelVA1,
3   Velocity:= VcomVA1,
4   Acceleration:= 20,
5   Deceleration:= 20,
6   Axis:= MlcVarGlobal.VA1,
7   InVelocity=> iv_MoveVelVA1,
8   Active=> ac_MoveVelVA1,
9   CommandAborted=> ca_MoveVelVA1,
10  Error=> er_MoveVelVA1,
11  ErrorID=> ID_MoveVelVA1,
12  ErrorIdent=> Ident_MoveVelVA1);

```

Fig.5-54: 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**:

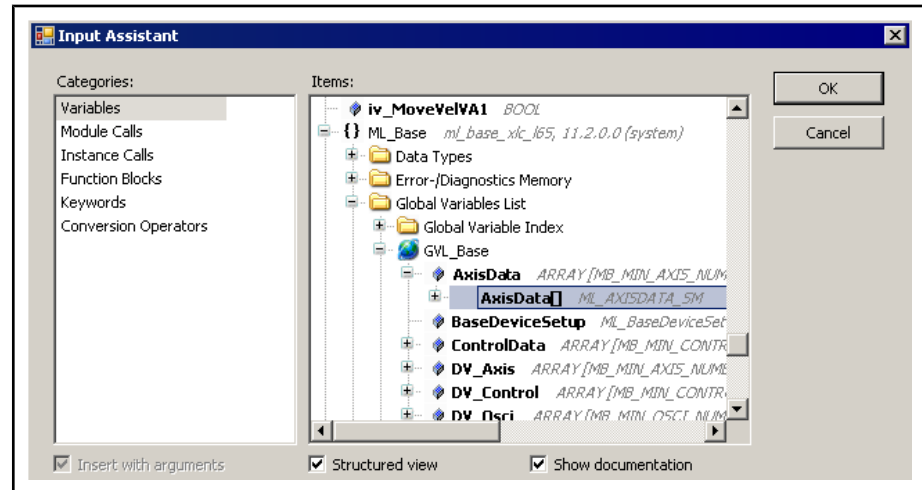


Fig. 5-55: Actual velocity value VA1 from the global variable list of the library *ML_Base.library*

Further input is supported by the Intellisense function. This means that after entering "." for "GVL_Base.AxisData[2].", all available elements are open and can be selected.

Confirm "rActualVelocity_i" with <Enter>.

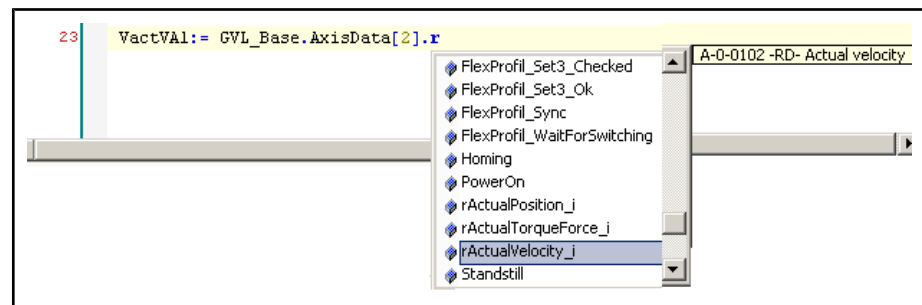


Fig. 5-56: Selecting the variables "ActualVelocity"

- 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 VactVal:= GVL_Base.AxisData[2].rActualVelocity_i;

```

Fig.5-57: Implementing the virtual axis VA1

5.5.3 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 compilation can be accessed via

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

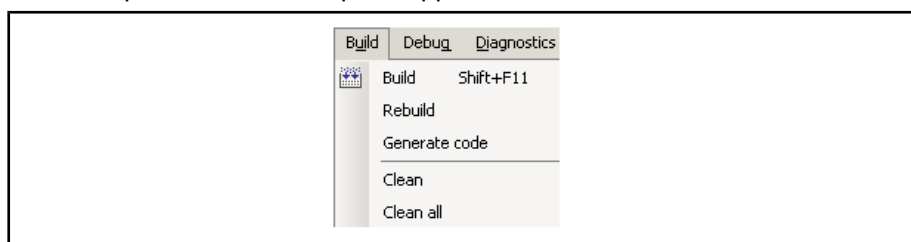


Fig.5-58: Build application

To apply the compiled data to the target control, select the menu item **De- bug ► Login** (see figure below).

First Steps - Motion

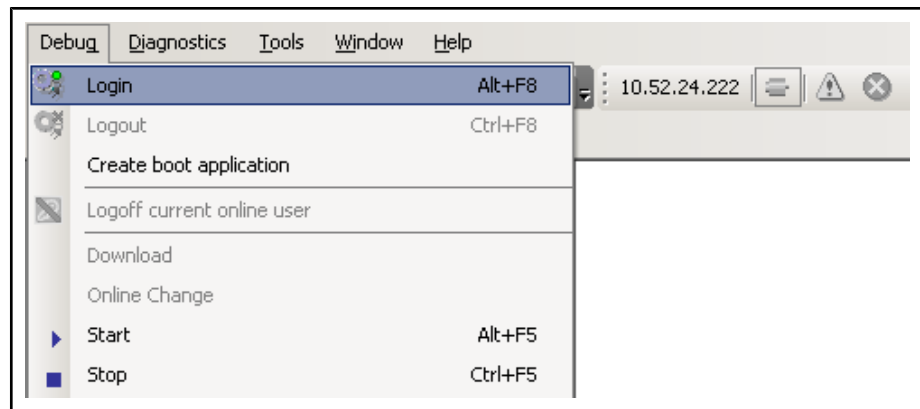


Fig.5-59: Transferring compiled data to the control

A prompt asking for agreement to download appears 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.

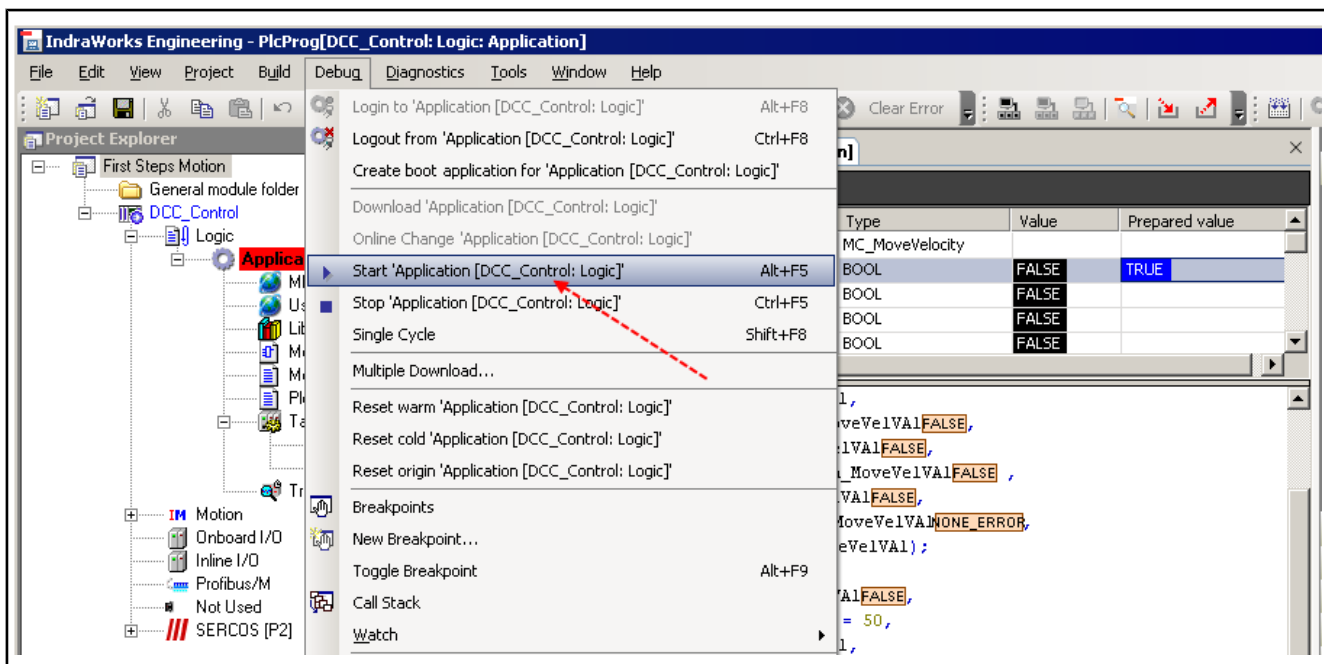


Fig.5-60: Starting the program processing in the control (1)

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

First Steps - Motion

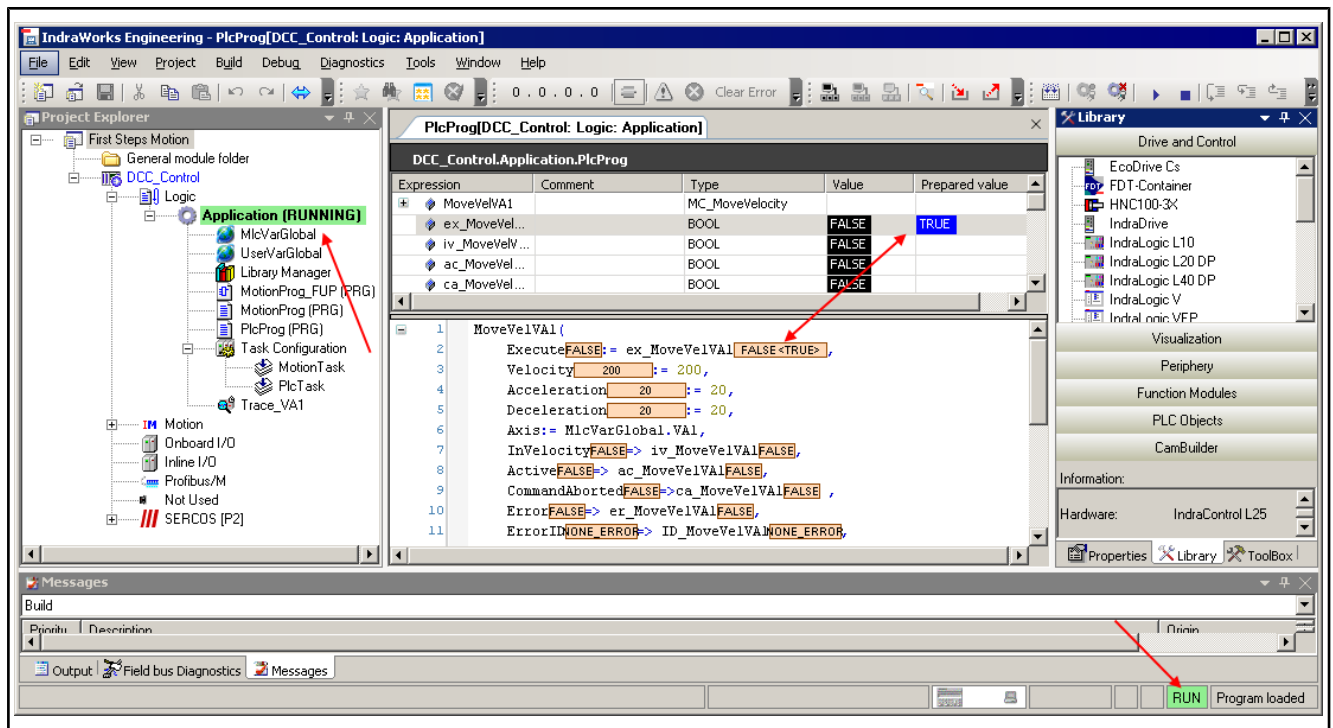


Fig.5-61: Starting the program processing in the control (2)

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 it.

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

First Steps - Motion

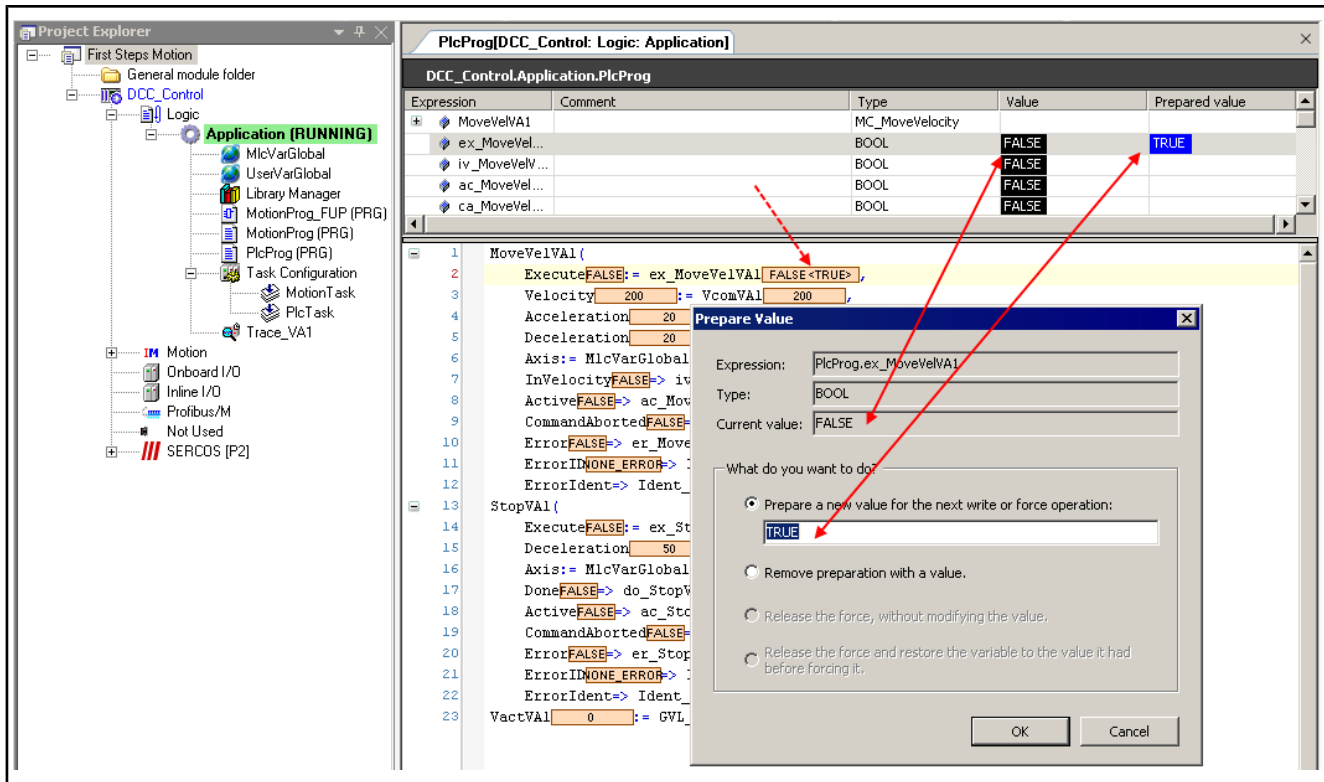


Fig.5-62: Preparation of "Write values"

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.

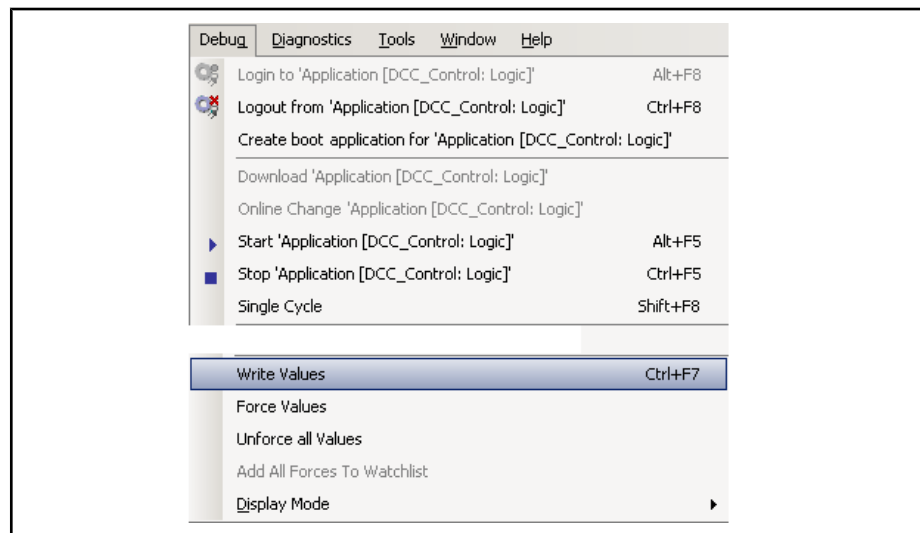


Fig.5-63: Write values

The positive edge thus created sets the outputs "InVelocity" and "Active" of the "MoveVelVA1" function block to "TRUE" (1) (figure below).

The axis travels at 200 rpm (2).

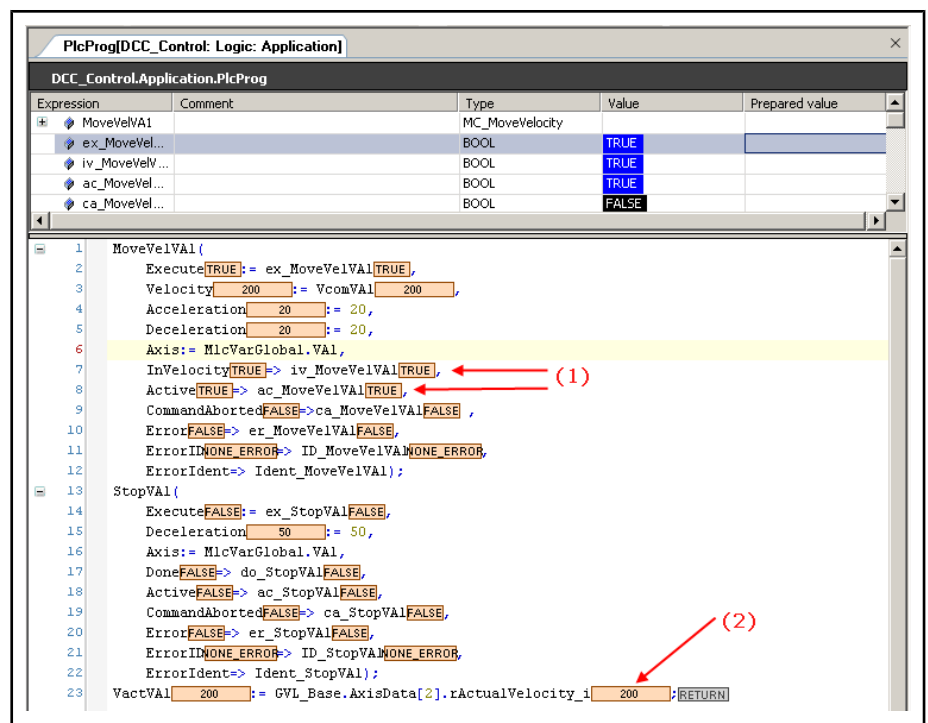


Fig.5-64: After "Write Values" at the "Execute" input

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.

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

IndraLogic provides a Trace tool to trace motion processes. It can be used to record

- 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

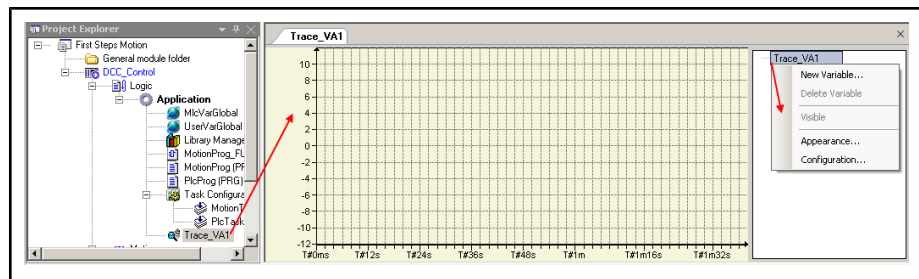


Fig.5-65: Dialog: New Trace object "Trace_VA1" with context menu

The following can be done with the Trace object via the context menu on the right side of the page:

- Configuration...
- Appearance... and
- New Variable...

Configuration...

The new Trace object must firstly be configured:

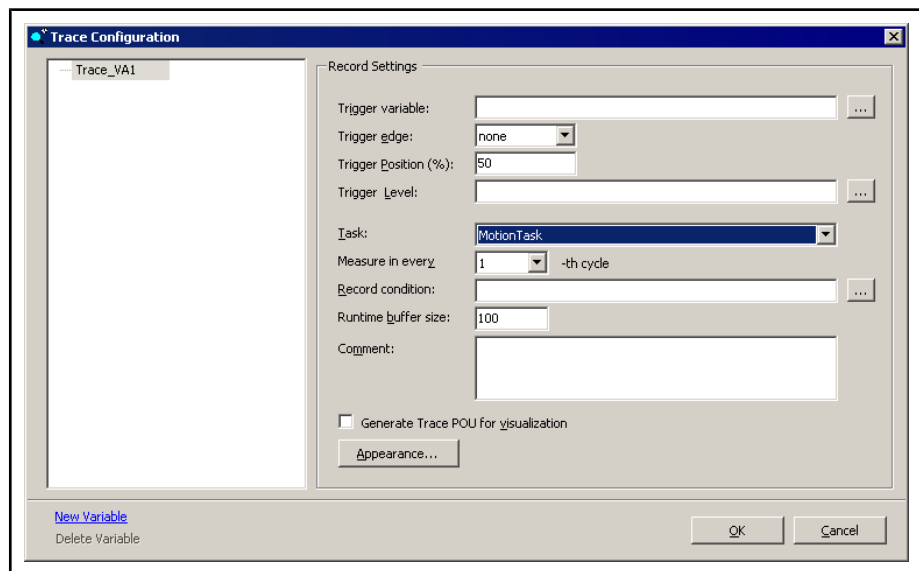


Fig.5-66: Dialog: New Trace object "Trace_VA1", minimum configuration

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 ones for which the values of trace variables are to be read after they are processed. Default: First task in the Task Configuration tree.

Measure in every ... -th cycle: 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.

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

First Steps - Motion

soon as the trace configuration is loaded into the control and the application is running.

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.

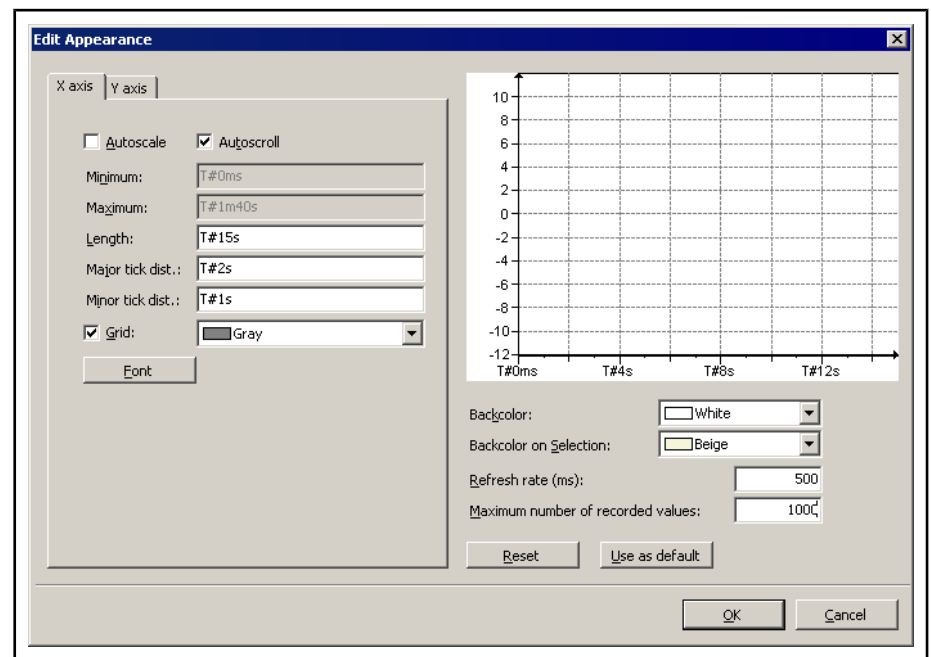


Fig. 5-67: Dialog: Display settings (appearance)

Select **Autoscroll**, positioning a **Length 15s** window over the curve. The **Scale division** is set out in a 2s grid with a **scale division** of 1s.



The process under scrutiny should not be longer than 15s.

New Variables...

Now the variables required must be named and their appearance adapted:

First Steps - Motion

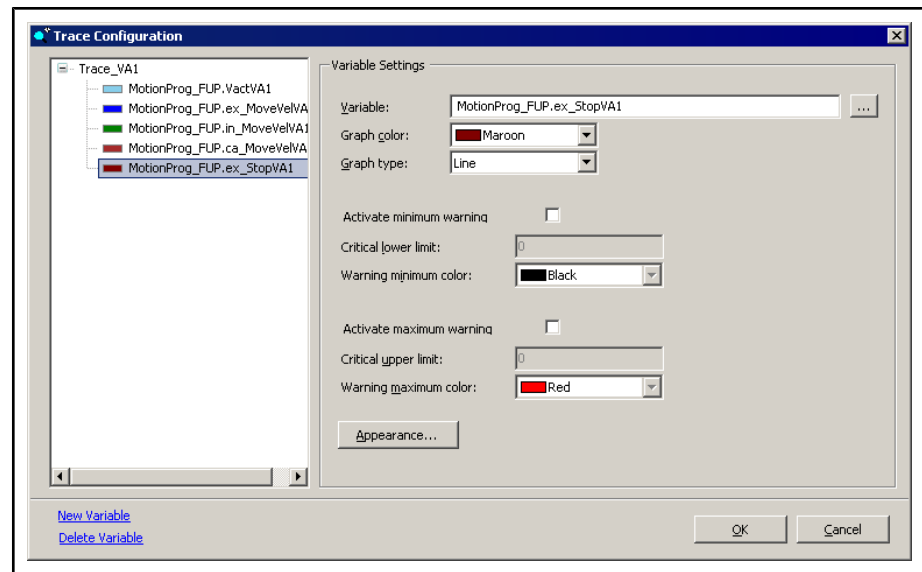


Fig.5-68: Dialog: Creating new variables and adapting appearance

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 appears by right-clicking into the left section of the Trace window.

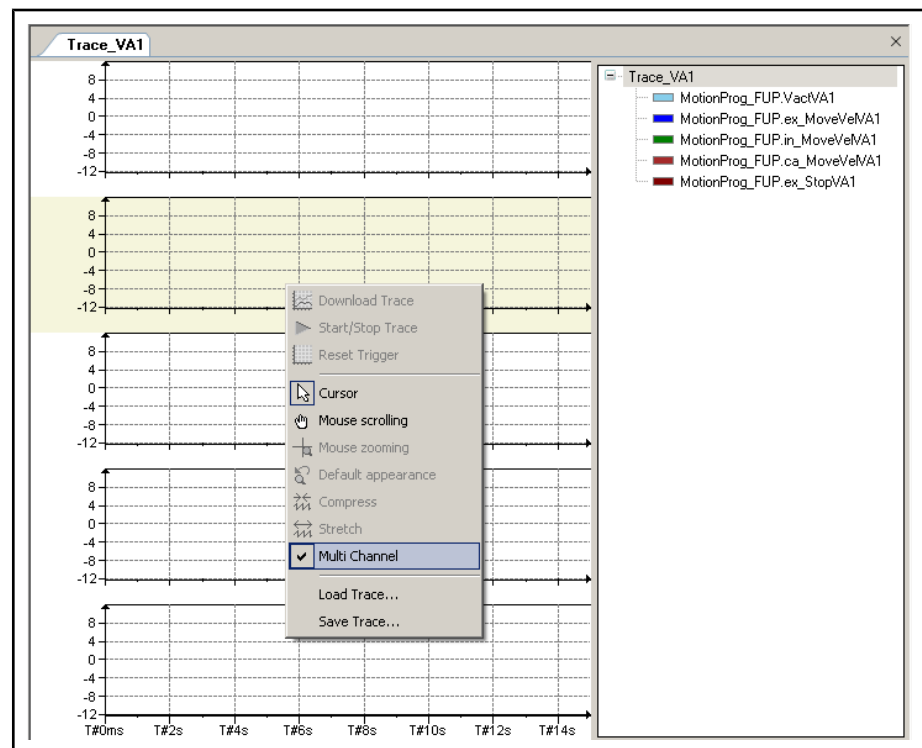


Fig.5-69: Menu to control the Trace recording

Login to Trace: Before a configured trace record on the control can be enabled for the application running there, the "Trace application" must be loaded there explicitly

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, for example.

Performing the Trace:

1. Login to application
Connect to the control.
2. Login to Trace
When the application is running, the data required is transmitted to the control via the "Login to 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 127](#),.
 - 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
<Alt>+<left/right> arrow keys	Horizontal scrolling of X-axis in all diagrams
<up/down >arrow keys	Horizontal scrolling of only the selected diagram
<+/-> keys	Simultaneous zooming of the coordinate system along the X- and Y-axis of all diagrams
<Shift>+<+/-> keys	Horizontal scrolling of all diagrams along the X-axis

Shortcuts	Action in the Trace diagram
<Ctrl>+<+/-> keys	Vertical scrolling of all diagrams along the Y-axis
<left/right >arrow keys	Left/right scrolling of cursor

Fig.5-70: Trace Editor shortcuts when Multi Channel is enabled

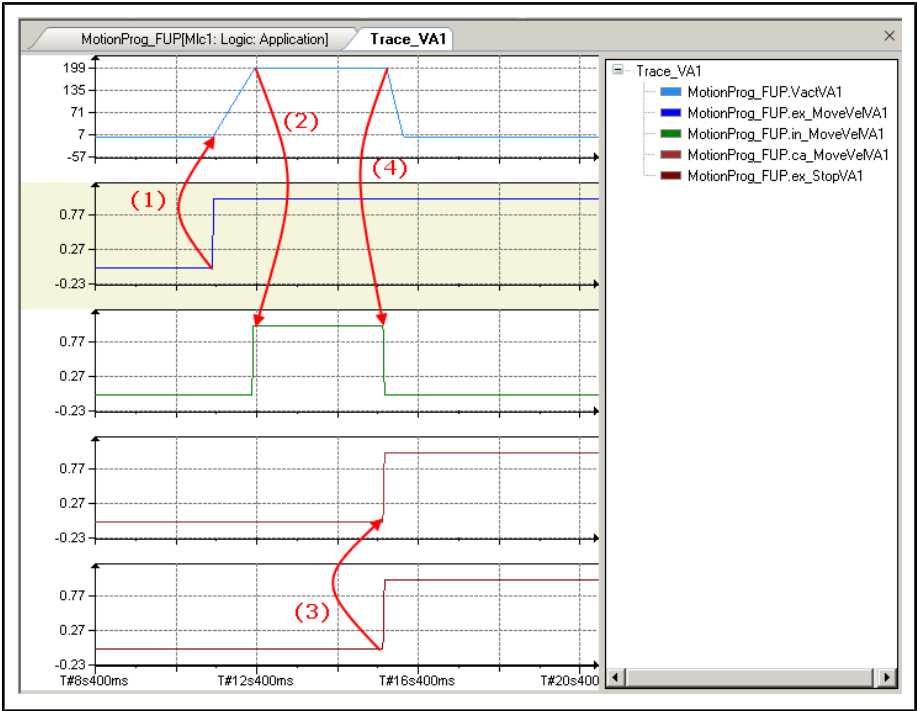


Fig.5-71: Trace recording for the virtual axis VA1

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 characteristics, acceleration at 20 rad/s² and deceleration at 50 rad/s²).

5.5.5 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 95](#).

MC_Power

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

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

"ErrorID" → "ID_GearInRA1" of type ERROR_CODE

"ErrorIdent" → "Ident_GearInRA1" of type ERROR_STRUCT

MC_GearOut

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

First Steps - Motion

"ErrorID" → "ID_GearOutRA1" of type ERROR_CODE

"ErrorIdent" → "Ident_GearOutRA1" of type ERROR_STRUCT

MC_Stop

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,

for the real axis, axis number 1, (axis number 1 has RA1, 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:**

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

First Steps - Motion

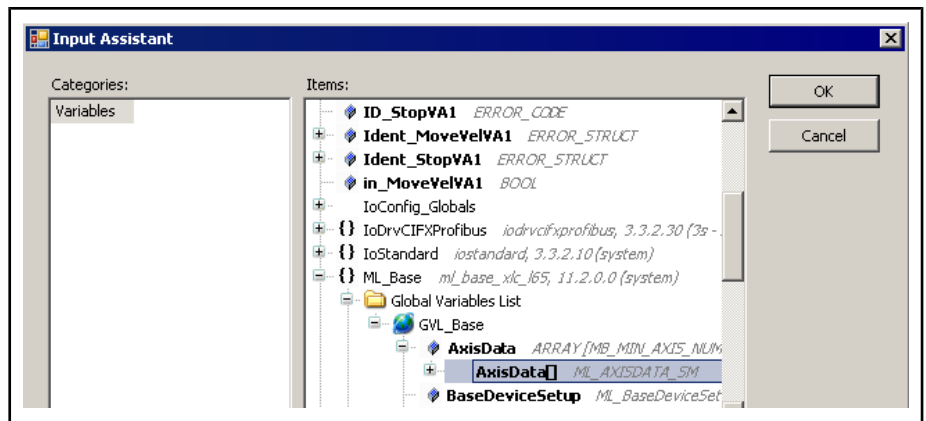


Fig.5-72: Actual velocity value VA1 from the global variable list of the library *ML_Base.library*

Further input is supported by the Intellisense function. This means that after entering "." at "GVL_Base.AxisData[1]." all available elements scroll up and can be selected.

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.5-73: Implementing the real axis

5.5.6 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 125.

The new project overwrites the old after issuing a prompt.

5.5.7 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. Follow 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.
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 it stops, it is actively held in this position.
12. Disable "ex_StopRA1" (MC_Stop).
The active holding of the real axis ends.

First Steps - Motion

13. Power off for the real axis

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



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

5.5.8 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 129](#), 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 $\Rightarrow$ FALSE $\Rightarrow$ TRUE.

The following figure shows the trace result:

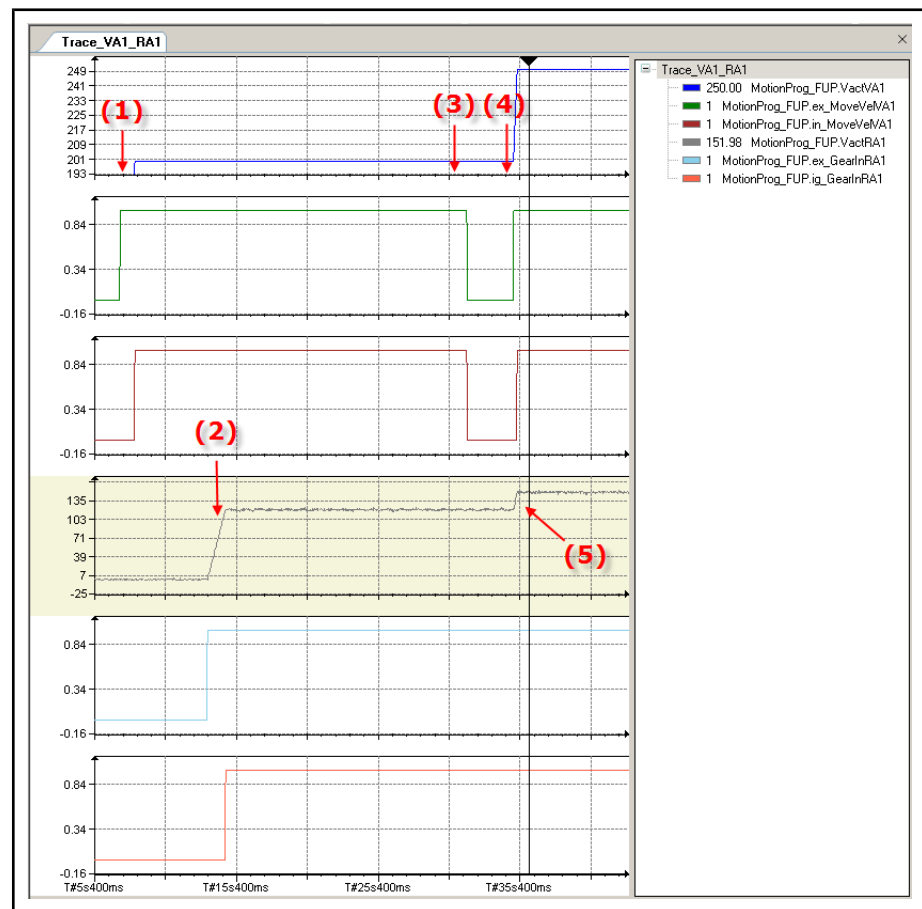


Fig.5-74: Trace recording for both axes

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,
- ③ - Enable command speed of virtual axis, "ex_MoveVelVA1" FALSE ⇒ TRUE,
- ⑤ - Jump response on real axis.

5.6 Restoring the Project

Location Based on the project created, an export file was generated that enables parts of the project to be regenerated independently of its control hardware, such as XLC L25, L45, L65 (with SERCOS III bus) or MLC L25, L45, L65 and the drive engineering.

The export file was created beginning from the "Application" node and contains the following components:

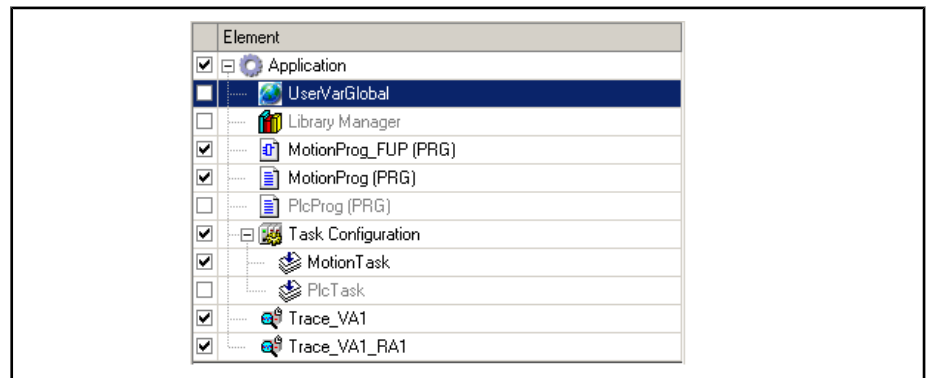


Fig. 5-75: Components of the export file

The given components are overwritten during import.



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

Restore Some involvement by the user is needed to ensure that the project runs on a selected control (XLC L25, L45, L65 (with SERCOS III bus) or MLC L25, L45, L65):

1. Clear old data in the control
Clear the old memory content of the control.
Details are described under [Resetting the Control \(CMLx5\), page 84](#).
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.

First Steps - Motion

3. Create the control
Drag a control, e.g. XLC L25, L45, L65 (with SERCOS III bus) or MLC L25, L45, L65 from the device library 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 in the project.
4. Create the Motion Subproject
Scan the SERCOS bus from the SERCOS III node.
The axis number "1" is assigned to the drive. It has to be renamed in "RA1".
5. Decide on a virtual axis. It gets the axis number "2" and the name "VA1".
In the global variable list "MlcVarGlobal", the axis numbers and names are provided as axis references of the "AXIS_REF" type.
This can be [chapter 5.3.8 "Downloading the Motion Subproject" on page 91](#)
6. Import the export file "Application_FirstSteps_Motion.iwx"
In the Project Explorer, expand the control node to **Logic ► Application**.
context menu "Application" **Import**.
Select the export file "Application_FirstSteps_Motion.iwx" in "Iw:\All Programs\Rexroth\IndraWorks\Projects"; Finish "Overwrite" operation.
The required files were copied. The project can now be run on the control.

6 First Steps - FlexProfile

6.1 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" Motion-Profile.

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.

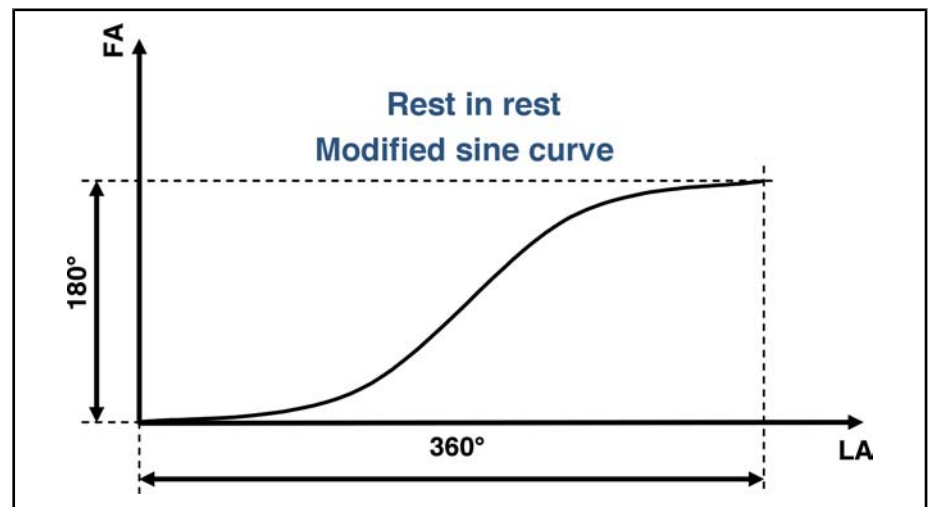


Fig. 6-1: Coupling profile between master axis (VA1) and slave axis (RA1)

The slave axis be provided with up to four MotionProfiles, each with a maximum of 16 (64) steps¹⁾.

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 145](#),
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 147](#).

For programming in ST, see [FlexProfile, data transfer in ST, page 148](#)

For programming in FBD, see [FlexProfile, data transfer in FBD, page 149](#)

The actual processing is started via the ML_FlexProfile function block. ML_FlexProfile links the slave axis to the master axis. A task fulfilled by the MC_GearIn in "First Steps - Motion".

The ML_FlexProfile function block is disabled by the MC_GearOut function block.

- For programming in ST, see [ML_FlexProfile in ST, page 151](#)

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

- For programming in FBD, see [ML_FlexProfile in FBD, page 153](#)

6.2 Technical Requirements

- PC with installed IndraWorks Engineering for IndraMotion MLC, 12VRS, control, type "IndraMotion MLC L25", "IndraMotion MLC L45" or "IndraMotion MLC L65", current firmware is loaded.
- IP address is known,
- control is connected to the programming device via Ethernet.
Drive controller, e.g. type "IndraDrive M", current firmware is loaded
- The drive controller is connected to the control via a SERCOS bus.
- The address of the drive controller is known at the SERCOS bus and if there is no drive, a bridged SERCOS bus exists.



For further information on FlexProfile, refer to the documentation "Rexroth IndraMotion MLC 12VRS Functional Description " chapter 8.7 "FlexProfile" (DOK-IM*MLC-FUNC****V12-RExx-EN-P).

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.

6.3 Creating the Motion Subproject

Some involvement by the user is needed to ensure that the project runs on a control of choice such as IndraLogic XLC IndraMotion L25, L45, L65 (with SERCOS III bus) or MLC L25, L45, L65:

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 84](#).

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 to the empty project. The control name can be chosen at random, although "DCC_Control" is recommended.

Complete device type and IP address.

First Steps - FlexProfile

Perform a connection test.
End adding as usual.
The desired control is now in the project.

4. Create the motion subproject
Scan the SERCOS bus from the SERCOS III node.



Select the "processing" in the control!

The axis number "1" is assigned to the drive. It has to be renamed in "RA1".

5. Decide on a virtual axis. It gets the axis number "2" and the name "VA1".
In the global variable list "MlcVarGlobal", the axis numbers and names are provided as axis references of the "AXIS_REF" type.
The [Downloading the Motion Project, page 91](#), can take place. The Motion component is then in "Ready for operation (BB)" state.
6. Import the export file "Application_FirstSteps_Motion.iwx"
In the Project Explorer, expand the control node to **Logic ► Application**.
context menu "Application" **Import**.
Select the export file "Application_FirstSteps_Motion.iwx" in "Iw:\All Programs\Rexroth\IndraWorks\Projects"; Finish "Overwrite" operation.
The required files were copied. The project can now be run on the control.

6.4 Creating a FlexProfile

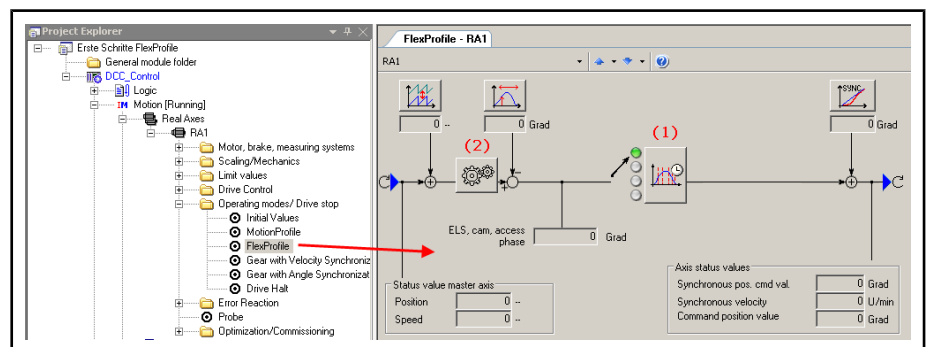
6.4.1 Creating a FlexProfile, General Information



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

6.4.2 Creating the FlexProfile via IndraWorks Dialogs

When clicking on the "FlexProfile" entry in the Project Explorer, the slave axis (RA1), a dialog opens in which the general profile configuration can be made.



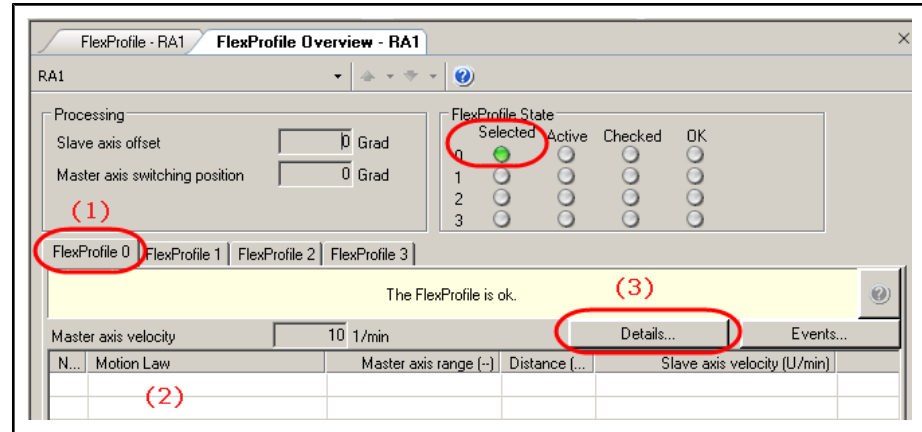
- (1) Transition to "FlexProfile 0"... "FlexProfile 3" dialogs
- (2) Transition to "Gear Settings" dialog

Fig.6-2: FlexProfile dialog with action chain between master and slave axis

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

First Steps - FlexProfile

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



- (1) Registers "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.6-3: Dialog: "FlexProfile Overview"; FlexProfile 0 is selected

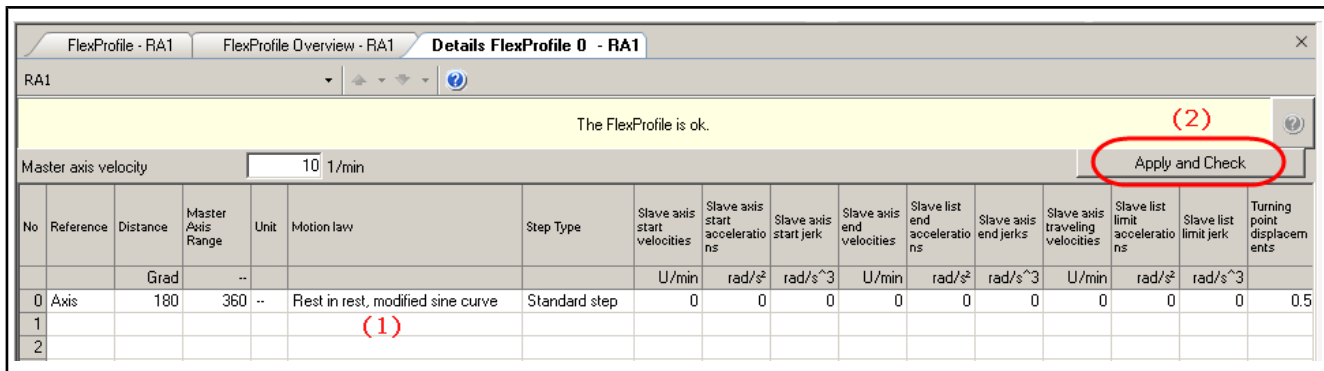
A new MotionProfile can be created via "Details" (1).

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 these, as shown below, and configure them according to the desired motion path.



- (1) Configuration of the individual motion steps [0...15 (63)] of the selected FlexProfile
- (2) Transition, apply the data of the MotionProfile and check whether the profile is correct.

Fig.6-4: Dialog: FlexProfile configuration

For FlexProfile 0, the following data was entered into the dialog above:

Reference	Axis
Distance	180 degrees
Master axis range	360
Unit	-
Motion law	Rest-in-rest, modified sine curve

First Steps - FlexProfile

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.

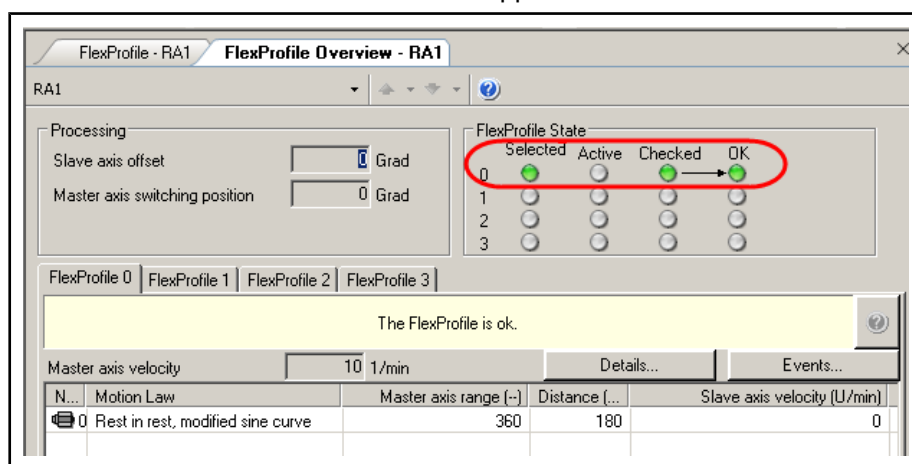


Fig. 6-5: Dialog: "FlexProfile Overview"; FlexProfile 0, checked → ok

The configured MotionProfile "FlexProfile 0" can now be started with the PLC function block "ML_FlexProfile".

6.4.3 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 148, or [PlcProg_FBD](#), page 149. Both programs are alternatively controlled via the PlcTask.

Usage in a SERCOS synchronous task (e.g. "MotionTask") is not recommended for real programs.

Tasks to be solved

1. Preparing data for the motion steps 0...15 (63) of the four FlexProfiles "FlexProfile 0"... "FlexProfile 3".

```
profileAxis: ARRAY[0..15] OF MB_PROFILE_STEP;
```

First Steps - FlexProfile

The ARRAY "profileAxis" thus contains the data of a FlexProfile.

For "First Steps - FlexProfile", the data set "0" of type "MB_PROFILE_STEP" is thus assigned in our program.

This corresponds to [fig. 6-4 "Dialog: FlexProfile configuration " on page 146.](#)

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"-

This corresponds to [fig. 6-5 "Dialog: "FlexProfile Overview"; FlexProfile 0, checked → ok" on page 147.](#)

FlexProfile, data transfer in ST

The variable "profileAxis" is first declared when providing the data set.

Then, 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⁾**.

TYPE MB_PROFILE_STEP				
Name	Type	Address	Initial	Comment
Master	MB_MASTER_TYPE		MASTER_AXIS0	Selection of master axis sources: time or position
Hub	LREAL			Stroke
Range	LREAL			Definition of master axis section
MotionLaw	MC_STEP_MODE			Definition of motion law
StepType	MB_STEP_TYPE		STEP_FIX	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.6-6: Description of the structure MB_PROFILE_STEP in the library manager
Use the "." operator with the following Intellisense function for the input.

²⁾ ML_Base_xxx_yyy: xxx stands for XLC or MLC; yyy stands for L25, L45 or L65

First Steps - FlexProfile

```

1  PROGRAM PlcProg
2  VAR
3      profileAxis: ARRAY[0..15] OF MB_PROFILE_STEP;
4  END_VAR

1  profileAxis[0].Master:= MB_MASTER_TYPE.MASTER_AXIS0;
2  profileAxis[0].Hub:= 180;
3  profileAxis[0].Range:= 360;
4  profileAxis[0].MotionLaw:= MC_STEP_MODE.REST_IN_REST_MOD_SINE;
5  profileAxis[0].StepType:= MB_STEP_TYPE.STEP_FIX;
6  profileAxis[0].StartVel:= 0;
7  profileAxis[0].StartAcc:= 0;
8  profileAxis[0].StartJrk:= 0;
9  profileAxis[0].EndVel:= 0;
10 profileAxis[0].EndAcc:= 0;
11 profileAxis[0].EndJrk:= 0;
12 profileAxis[0].TravelVel:= 0;
13 profileAxis[0].LimitAcc:= 0;
14 profileAxis[0].LimitJrk:= 0;
15 profileAxis[0].TurningPointDisplacement:= 0.5;
    
```

Fig.6-7: "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.

```

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.6-8: 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 FBD

The variable "profileAxis" is first declared when providing the data set.

Then, 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_STEP³⁾**.

³⁾ ML_Base_xxx_yyy: xxx stands for XLC or MLC; yyy stands for L25, L45 or L65

First Steps - FlexProfile

TYPE MB_PROFILE_STEP				
Name	Type	Address	Initial	Comment
Master	MB_MASTER_TYPE		MASTER_AXIS0	Selection of master axis sources: time or position
Hub	LREAL			Stroke
Range	LREAL			Definition of master axis section
MotionLaw	MC_STEP_MODE			Definition of motion law
StepType	MB_STEP_TYPE		STEP_FIX	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.6-9: Description of the structure MB_PROFILE_STEP in the library manager
Use the "." operator with the following Intellisense function to input the assignments.

PlcProg_FUP[DCC_Control: Logic: Application]	
1	PROGRAM PlcProg_FUP
2	VAR
3	profileAxis: ARRAY[0..15] OF MB_PROFILE_STEP;
4	END_VAR
1	MB_MASTER_TYPE.MASTER_AXIS0 — profileAxis[0].Master
2	180 — profileAxis[0].Hub
3	360 — profileAxis[0].Range
4	MC_STEP_MODE.REST_IN_REST... — profileAxis[0].MotionLaw
5	MB_STEP_TYPE.STEP_FIX — profileAxis[0].StepType
6	0 — profileAxis[0].StartVel
7	0 — profileAxis[0].StartAcc
8	0 — profileAxis[0].StartJrk
9	0 — profileAxis[0].EndVel
10	0 — profileAxis[0].EndAcc
11	0 — profileAxis[0].EndJrk
12	0 — profileAxis[0].TravelVel
13	0 — profileAxis[0].LimitAcc
14	0 — profileAxis[0].LimitJrk
15	0.5 — profileAxis[0].TurningPoin...

Fig.6-10: "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.

First Steps - FlexProfile

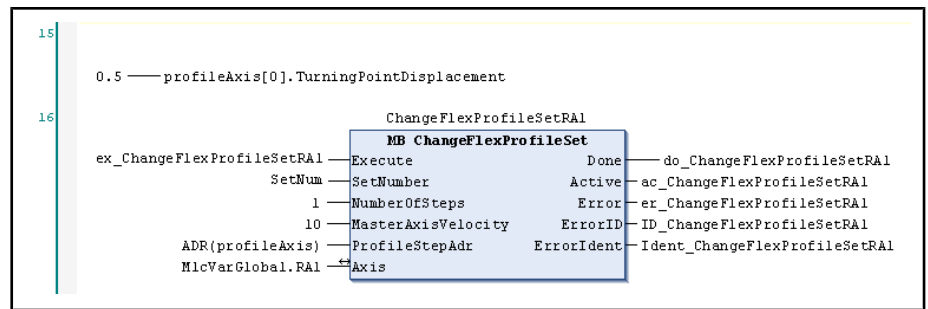


Fig.6-11: Function block to transfer the data array to the control (SetNum = 0 for FlexProfile 0)

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.

6.5 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 creates a 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 example by calling the PLCopen function block "MC_MoveVelocity". The slave axis travels the desired MotionProfile.

Therefore, import the export file "Application_FirstSteps_IndraLogic_Motion.iwx" from the directory ""Iw:\All Programs\Rexroth\IndraWorks\Projects"", **DCC_Control ► Logic ► Application**, context menu "Import".

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. It 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: enter, <F2> input help, **User-defined types ► ML_PLCOpen ► POUs ► Motion ► ML_FlexProfile**, deselect option "Insert with arguments".

First Steps - FlexProfile

3. Complete line:

```
FlexProfileRA1: ML_FlexProfile;
```

Add empty line instead of "GearInRA1"

<F2> input help, **Instance calls** ► **FlexProfileRA1**, option "Insert with arguments"

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

```

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.VAL,
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.6-12: "ML_FlexProfile" is inserted and assigned (ST)

ML_FlexProfile in FBD

As described above, add a function block of type "ML_FlexProfile" and assign it as described in the following.

Add ML_FlexProfile and declare a "FlexProfileRA1" instance:

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

First Steps - FlexProfile

"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

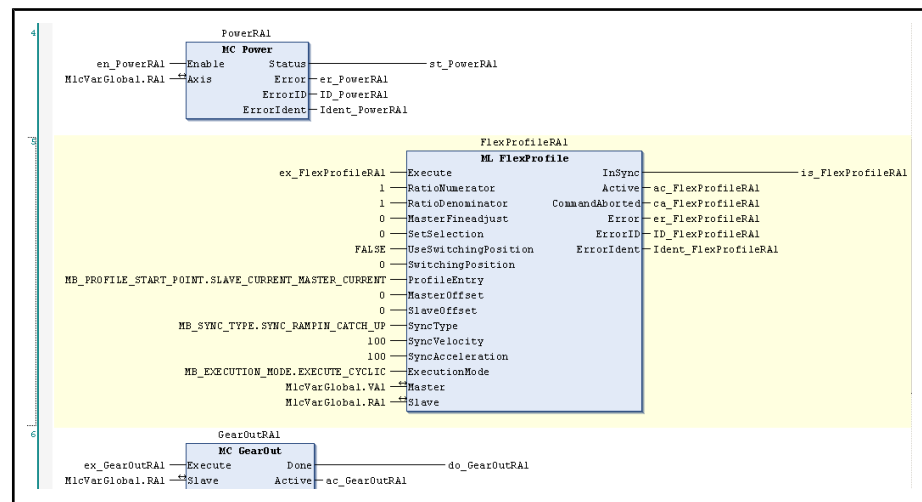


Fig.6-13: "ML_FlexProfile" is inserted and assigned (FBD)

6.6 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

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

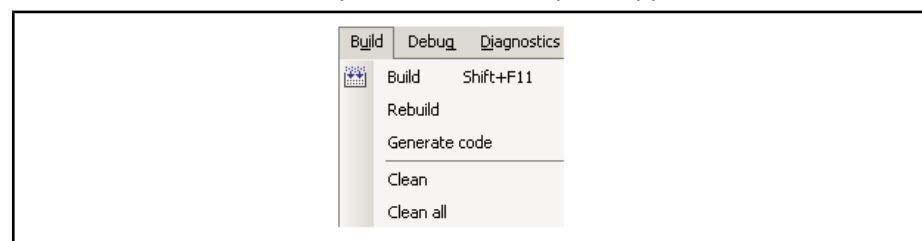


Fig.6-14: Build application

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

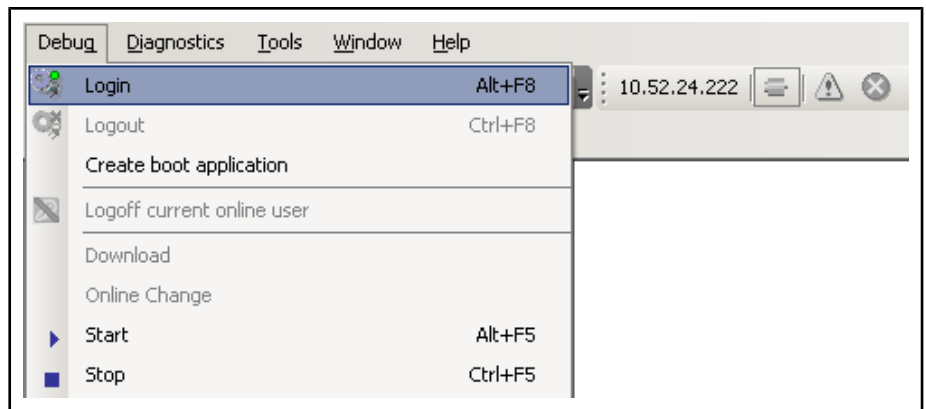


Fig. 6-15: Transferring compiled data to the control

A prompt asking for agreement to download appears 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**.

6.7 Workflow "Virtual Axis → FlexProfile 0 → Real Axis "

The following procedure shows how to start and stop the axes and the Flex-Profile:

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:

First Steps - FlexProfile

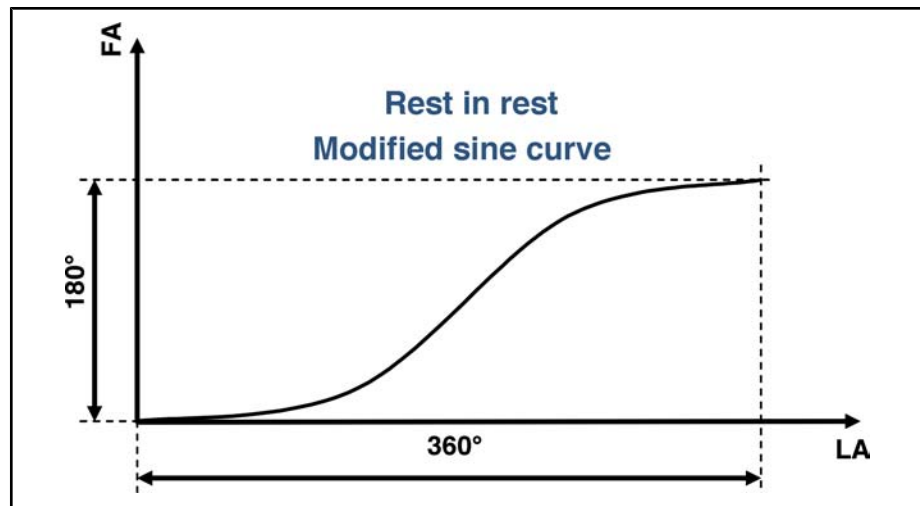


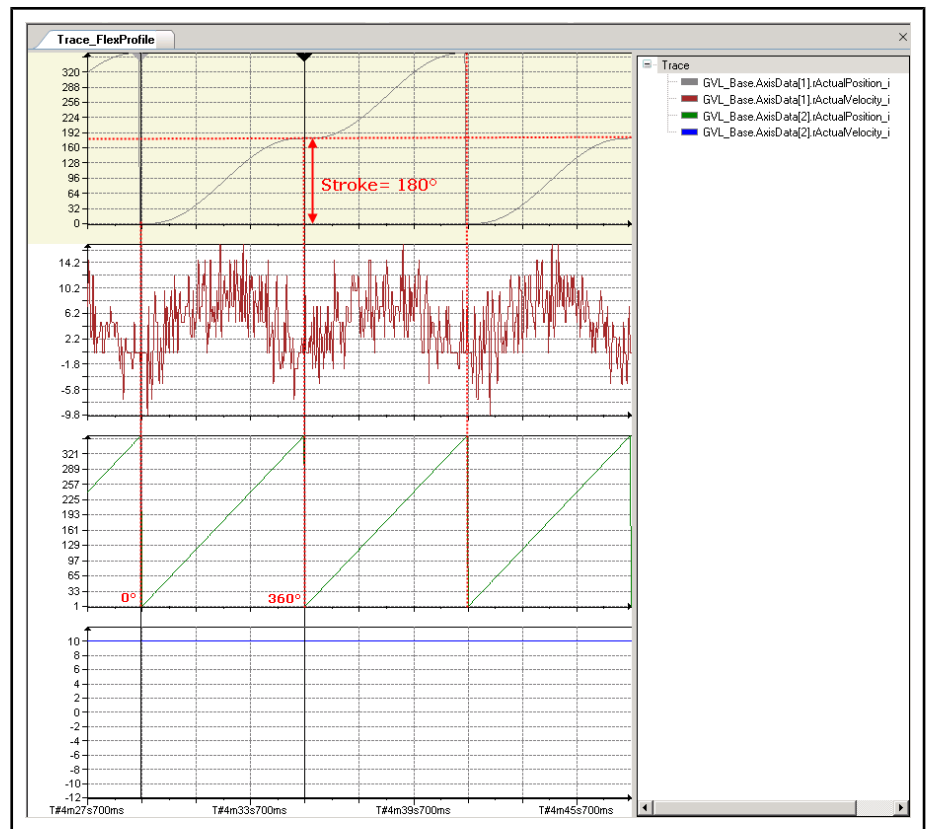
Fig.6-16: Coupling profile between the master axis (VA1) and slave axis (RA1)

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. It is held enabled in this position after reaching standstill.
8. Stop the virtual axis
Enable "ex_StopVA1" (MC_Stop)
The virtual axis is decelerated to VactVA1 = 0 rpm.
9. Power off for the real axis
Disable "en_PowerRA1" (MC_Power) by "Write Values".

6.8 Recording the Motion of Both Axes (Trace)

The motions of master axis and slave axis can be displayed in the trace recording. The following results:

First Steps - FlexProfile



Grey Slave axis ("RA1"), position
Red Slave axis ("RA1"), velocity
Green Master axis ("VA1"), position
Blue Master axis ("VA1"), velocity

Fig.6-17: Recording the motion of both axes

The trace recording shows the velocity characteristics of the master axis.

It 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.

6.9 Restoring the Project

Based on the project created, an export file was generated enabling parts of the project to be regenerated independently of its control hardware [IndraLogic XLC L25, L45, L65 (with SERCOS III bus) or IndraMotion MLC L25, L45, L65] and the drive engineering.

The export file was created beginning from the "Application" node and contains the following components:

First Steps - FlexProfile

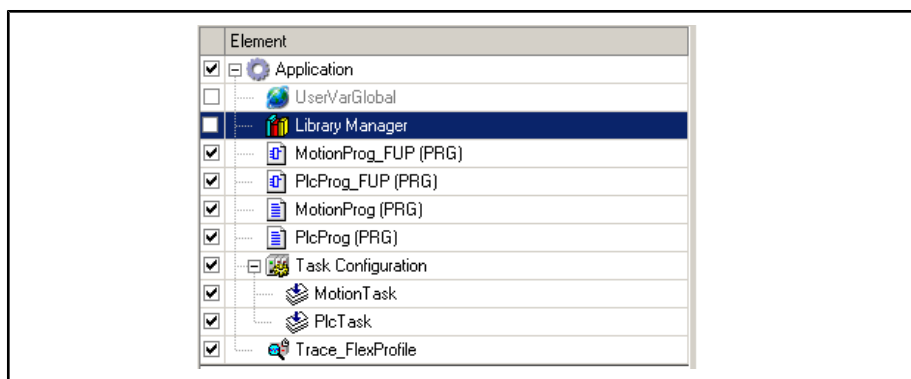


Fig. 6-18: Components of the export file

The given components are overwritten during import.

The export file "Application_FirstSteps_FlexProfile.iwx" is located in the directory "Iw:\All Programs\Rexroth\IndraWorks\Projects".

7 First Steps - RobotControl

7.1 Project Goal "First Steps - RobotControl"

A simple example will show 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 will be addressed. It covers the procedure from the project creation up to the execution of the RoCo source code.

In the example project "RoCoTestV12", 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 and
2. the axis numbers 1, 2 and 3 are assigned in the same sequence.

Thus, it is guaranteed that the PLC program runs without any adaptations.



The user can either load the example project from the IndraWorks installation DVD from the directory "\\AddOns\\Mlc\\Example-Project\\RobotControl" which contains

- the parameter export file (*.par),
- the RCL program (*.qll),¹⁾
- the PLC program and
- 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.

7.2 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 module (CFL01.1 Y1)

Required software IndraWorks installation using MLC as target system.

¹⁾ RCL means **RobotControlLanguage**

First Steps - RobotControl

7.3 Procedure when Creating an RoCo Project

1. Starting IndraWorks and creating a new project

After starting IndraWorks, create the new project via **File ► New ► Project**.

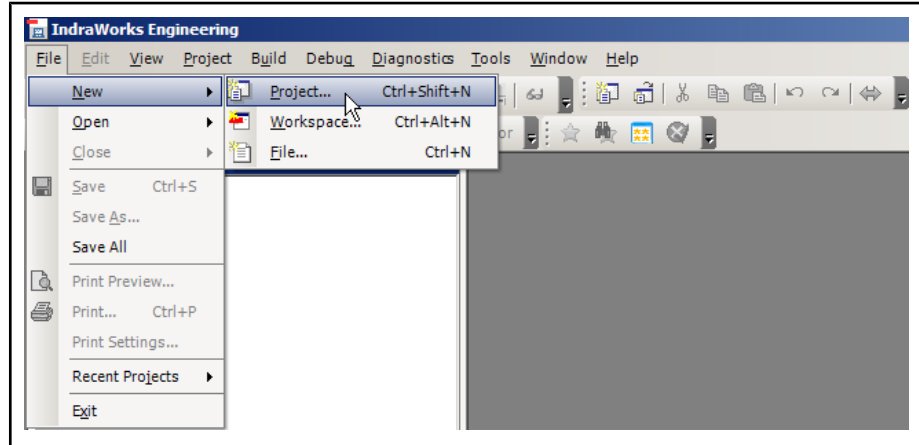


Fig. 7-1: Create a new IndraWorks project

2. Creating and configuring the 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 "Configuration", the function module SRAM module (CFL01.1-Y1) - memory extension for the MLC Lx5 can be created. (This is also possible by "dragging" from the **Device Library ► FM** and "dropping" into the Project Explorer of the control.)

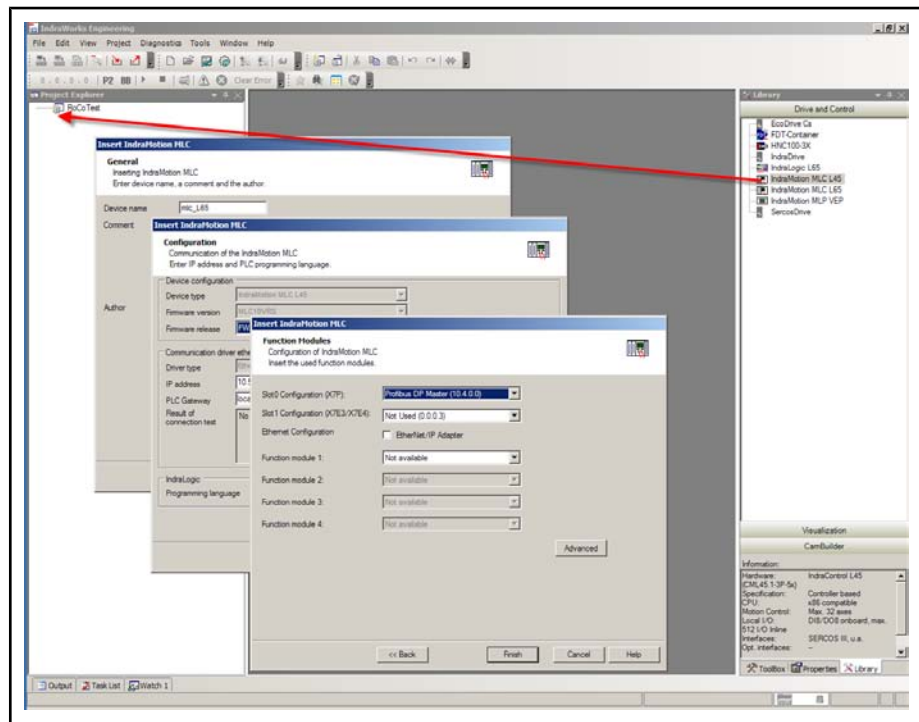


Fig. 7-2: Creating and configuring an MLC

First Steps - RobotControl

3. Creating and configuring **three** axes

Two possible scenarios are described:

Scenario 1 with virtual axes:

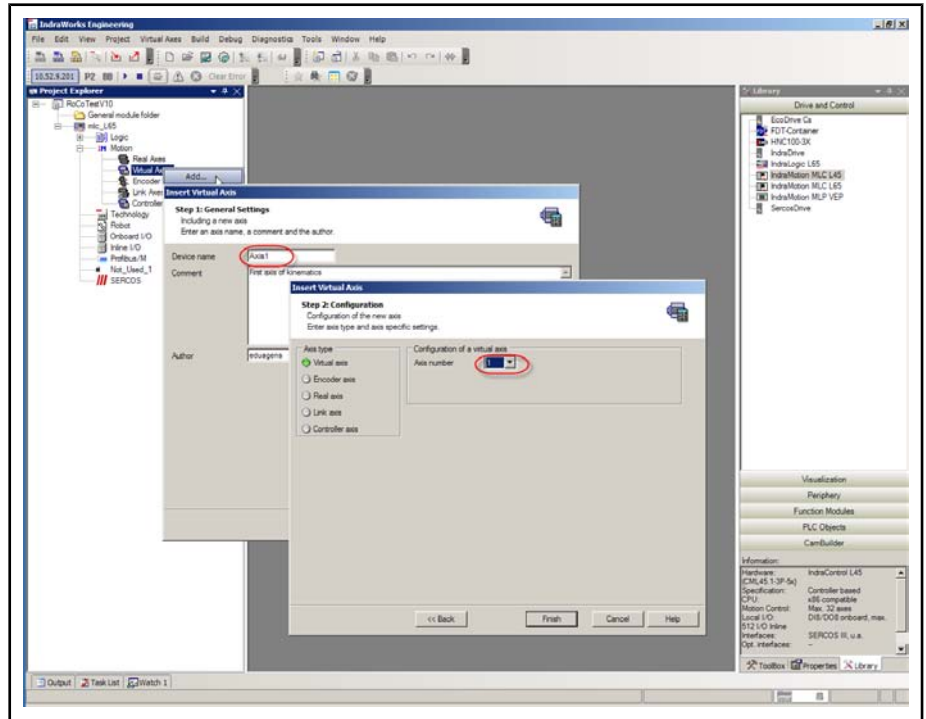
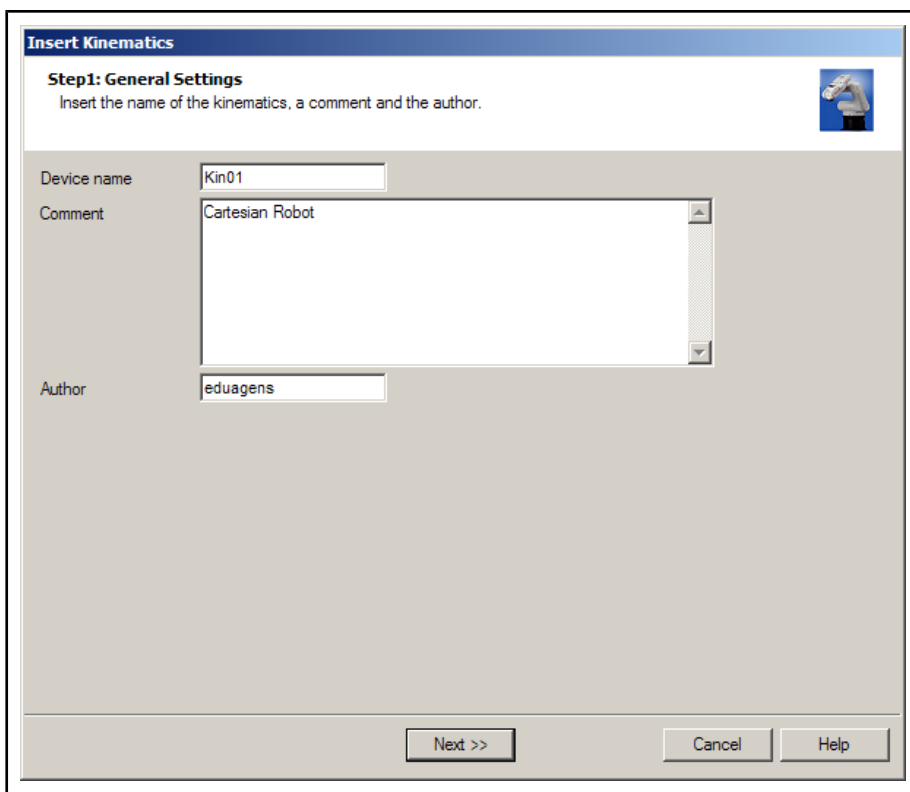


Fig.7-3: Project with virtual axes

Scenario 2 with real axes:

Creating real axes is executed by "dragging" from the **Device Library ► Drive and Control** and "dropping" into the Project Explorer on the SERCOS node.

First Steps - RobotControl



Insert Kinematics

Step1: General Settings
Insert the name of the kinematics, a comment and the author.

Device name: Kin01

Comment: Cartesian Robot

Author: eduagens

Next >> Cancel Help

Fig.7-6: 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 - RobotControl

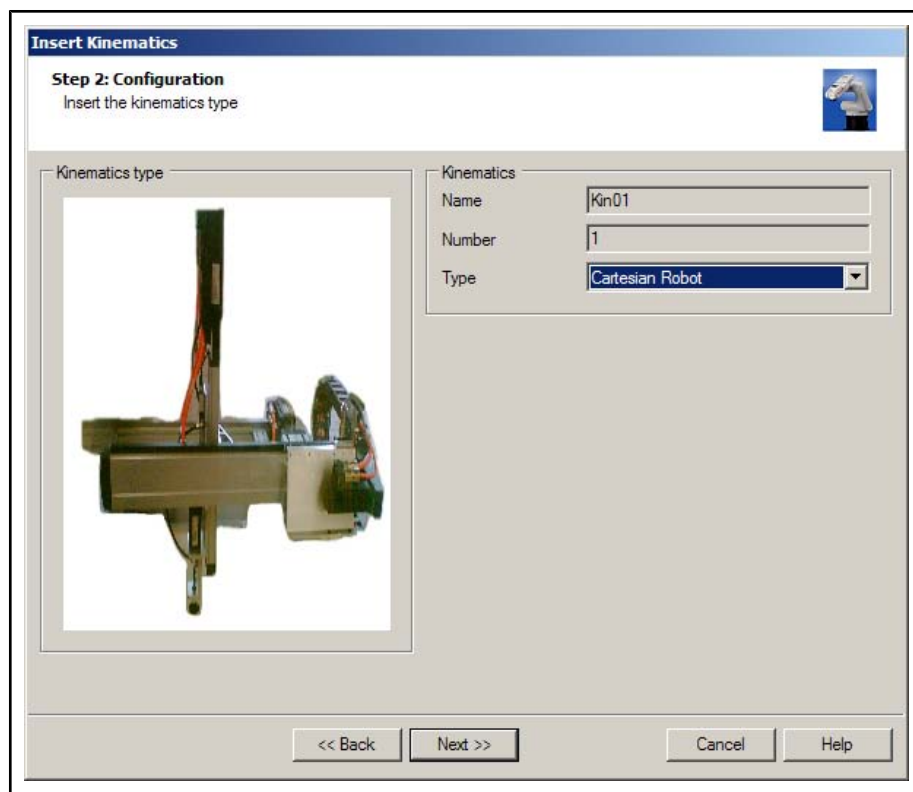


Fig.7-7: Specifying the kinematic type

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.

First Steps - RobotControl

Insert Kinematics

Step 3: Axis Assignment
Assign the required axes for the kinematics.

Cartesian Robot

Axis assignment

Name	Axis type	Control	Axis
A1	Positioning axis	0	Axis1
A2	Positioning axis	0	Axis2
A3	Positioning axis	0	Axis3
A4	---	0	
A5	---	0	
A6	---	0	
A7	---	0	
A8	---	0	
A9	---	0	
A10	---	0	
A11	---	0	
A12	---	0	
A13	---	0	
A14	---	0	

<< Back Next >> Cancel Help

Fig.7-8: Assigning kinematic axes.

All axes belonging to the kinematics (real or virtual) have to be defined and assigned here.

Insert Kinematics

Step 4: Axis Length and Coupling
Insert axis length and axis coupling.

Cartesian Robot

Axis length/exclusion area

Axis	Length	Unit
L1	0	mm
L2	0	mm
L3	0	mm
L4	0	mm
L5	0	mm
L6	0	mm
L7	0	mm
L8	0	mm
L9	0	mm
L10	0	mm
L11	0	mm
L12	0	mm
L13	0	mm
L14	0	mm

Axis coupling

Axis	Coupling
K1	0
K2	0
K3	0
K4	0
K5	0
K6	0
K7	0
K8	0
K9	0
K10	0
K11	0
K12	0
K13	0
K14	0

<< Back Next >> Cancel Help

Fig.7-9: Axis length and axis coupling

First Steps - RobotControl

The axis couplings and axis lengths typical for the robots are entered here. This is not required for the example described.

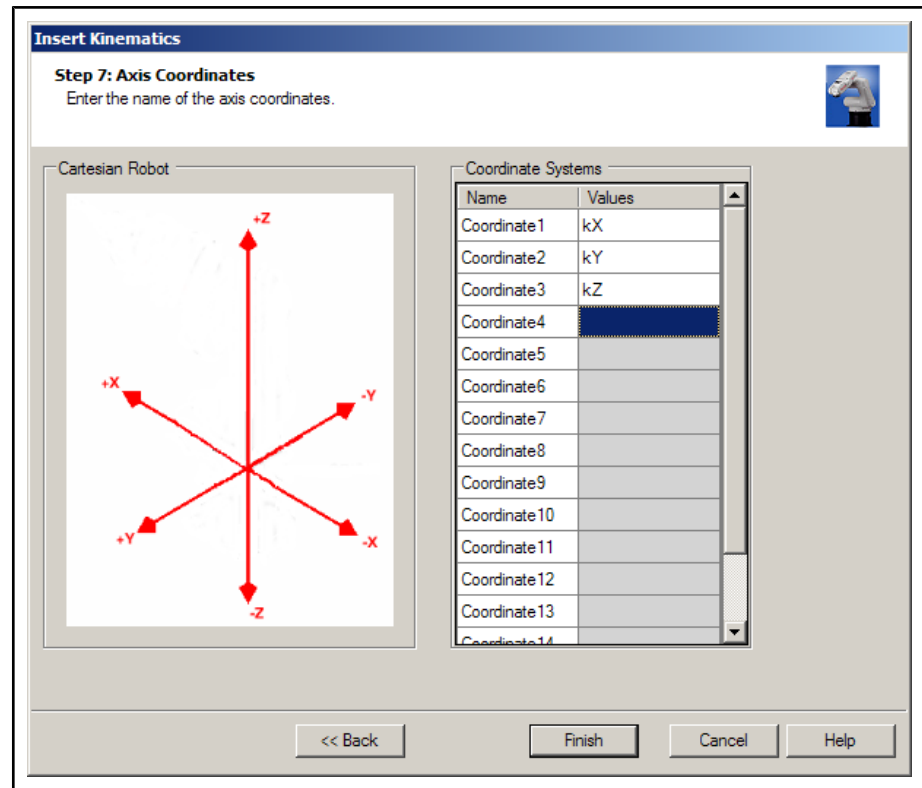


Fig. 7-10: Specifying coordinate names

The coordinate names are defined in this step.



The coordinate designations defined are used in the RoCo programming carried out later.

5. Go 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 "RoCoTestV12\RoCoTestV12virtual.par" for virtual axes or "RoCoTestV12\RoCoTestV12real.par" for real axes.

After a successful parameter import, it should be possible to switch the control to BB (P4).

First Steps - RobotControl

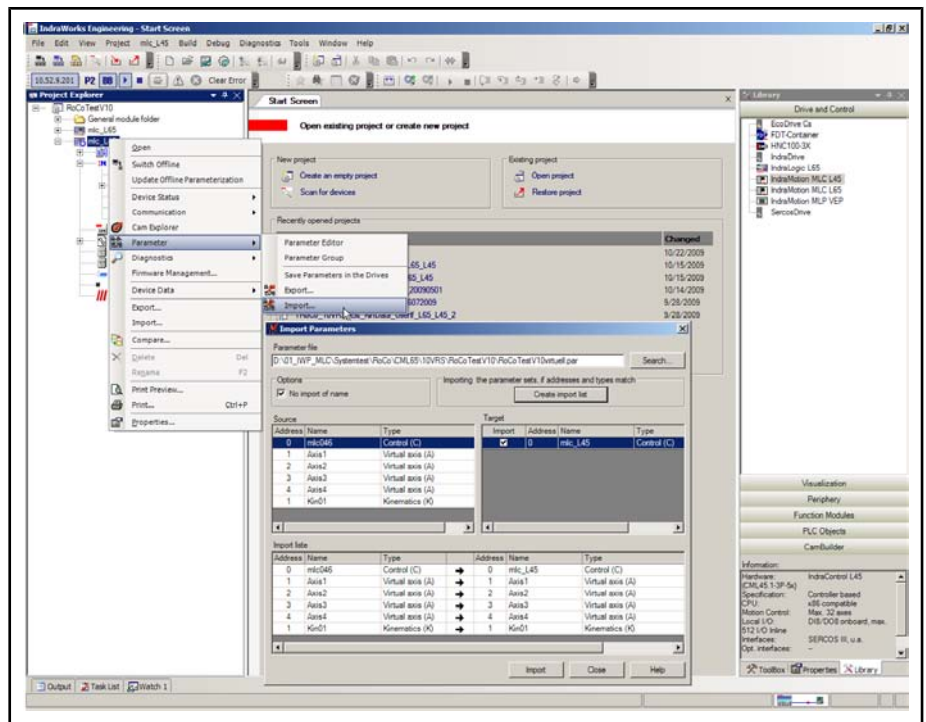


Fig.7-11: Importing parameters

7. Creating a motion program

A new RoCo motion program is created for the kinematics.

It can also be imported from the project **RoCo-TestV12** ► **mlc_Lx** ► **_Emb_Prog1.qll** ► **Prog1.qll** using

- 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 - RobotControl

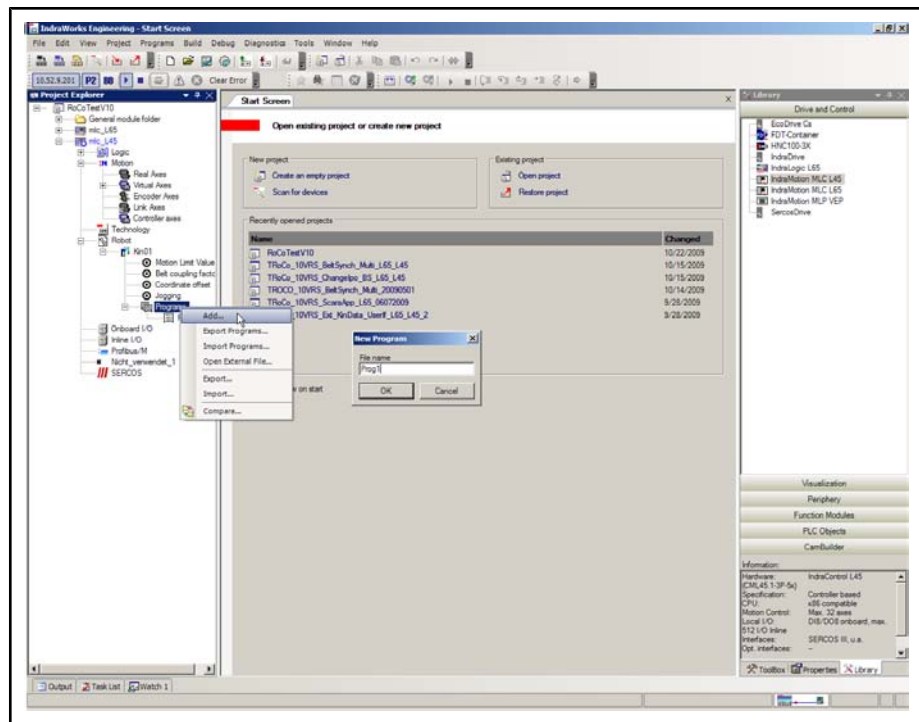


Fig.7-12: Creating new RoCo program

After editing the RoCo motion program (*.QLL), it can be compiled.

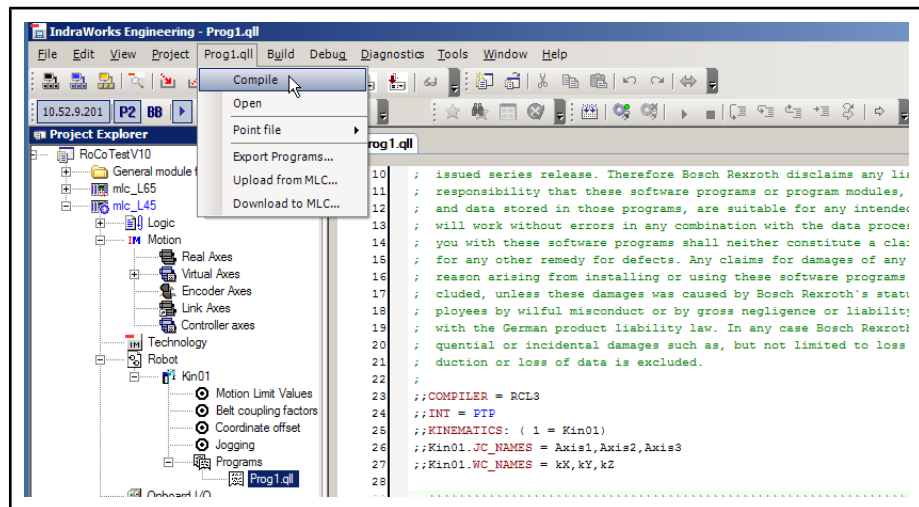


Fig.7-13: Editing the RoCo motion program

After the compilation, a point table (*.PKT) is created automatically and can now be edited.

First Steps - RobotControl

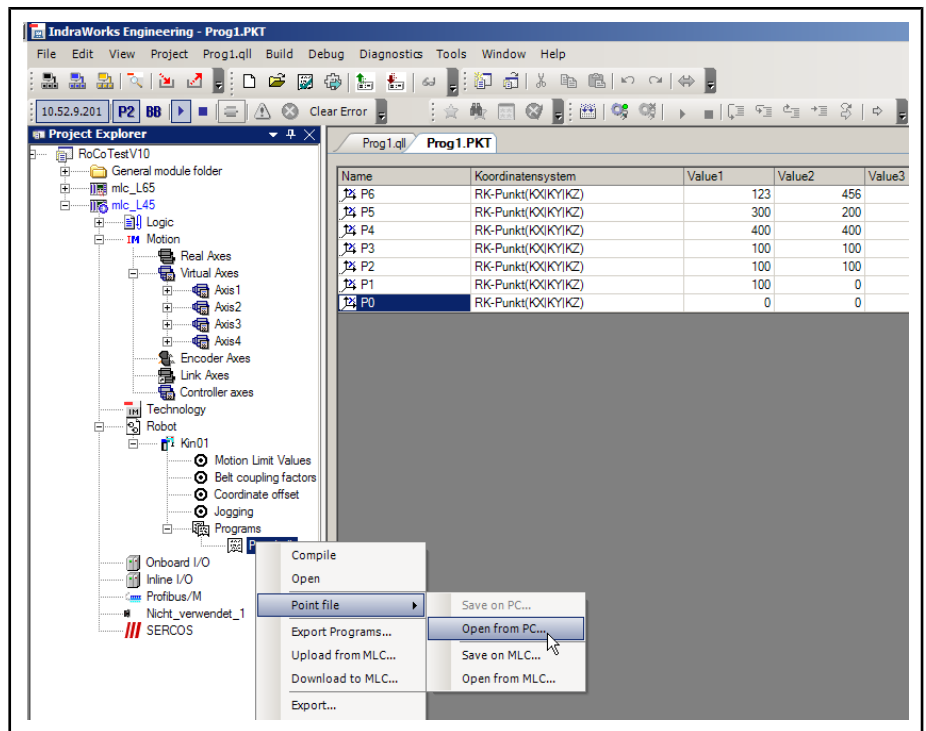


Fig.7-14: Editing the coordinate point table.

The motion program, point file etc. are loaded in the control when operating online.



If no SRAM function module exists, this step is to be repeated after each control reboot since the data is located in the volatile memory.

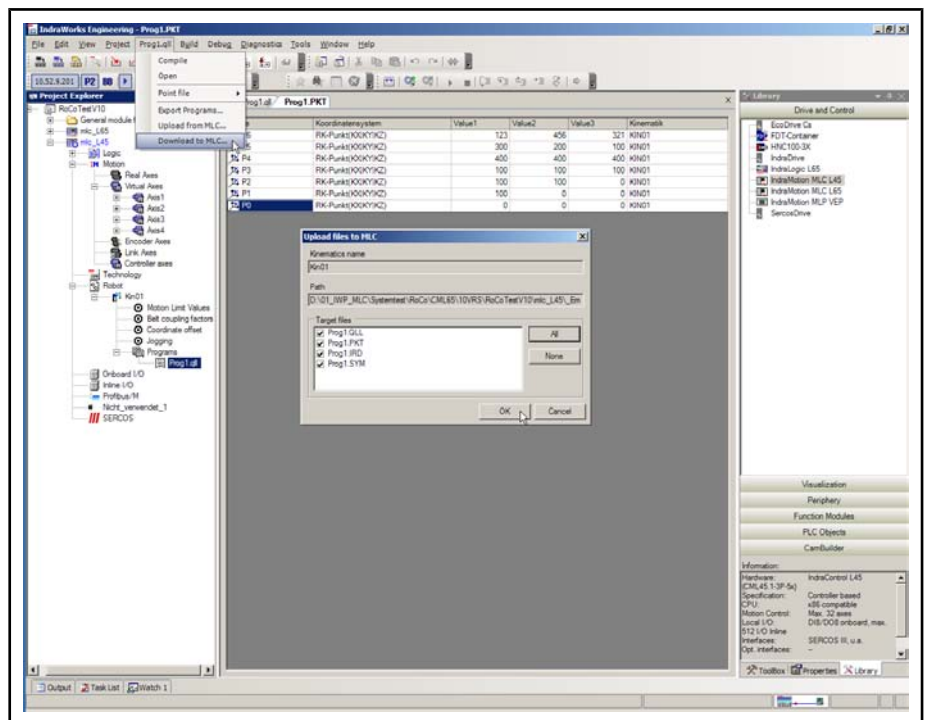


Fig.7-15: Loading RoCo program in the control.

First Steps - RobotControl

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 connecting the power supply for real axes up to the RoCo program start and kinematic state can be read as well.

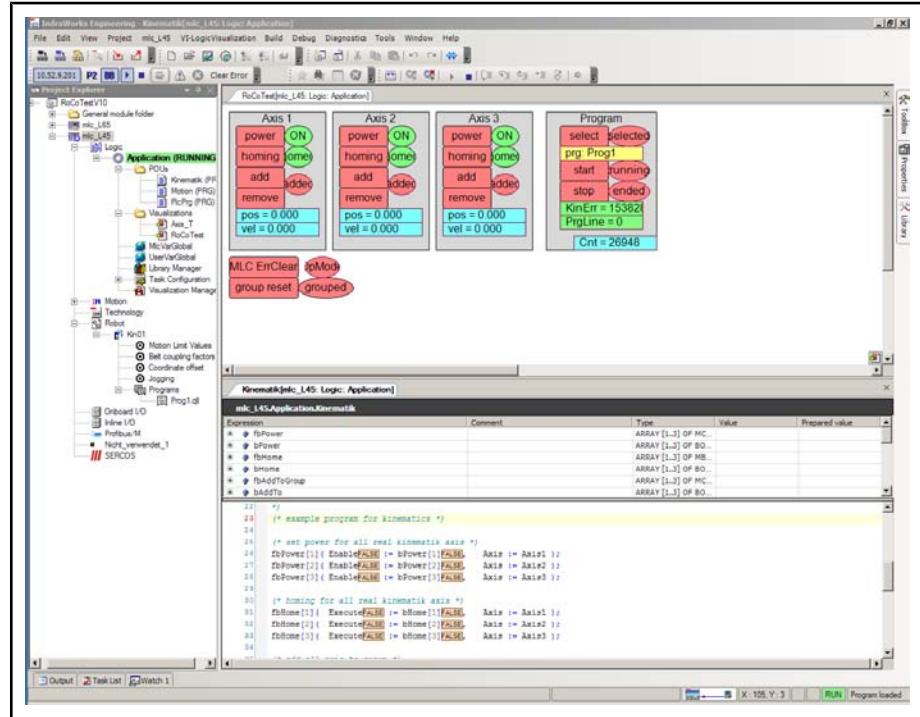


Fig. 7-16: RoCo in the PLC program

Brief description of the PLC functionality in the sequence programmed.

1. Connect power.

This command is only to be executed for real axes.

This bit is always set for virtual axes.

Whether power was or was not 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_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.

This is only possible if all axes belonging to the kinematics are grouped.

Whether all axes are required in Coordinated Motion, can be seen under the status "AxisData[x].CoordinatedMotion".

4. Remove axes from the kinematics
This is required to travel the individual axes outside the RoCo motion program.
5. Select RoCo motion program
To execute a RoCo motion program, it has to be selected. This is to be carried out again after each program end and after each program stop.
6. Start RoCo motion program
This command starts the program selected.
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.

7.4 Details on the Project "RoCoTestV12"

Parameter presettings (minimum changes)

C Parameter:

The C-parameter assignment results automatically when the MLC is implemented in the Project Explorer 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-0-0056, 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
Kin01.V_PTP = 0.4 ; velocity for point to point movement
Kin01.A = 1000 ; acceleration
Kin01.V = 1200 ; velocity
```

First Steps - RobotControl

```
;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
```

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

```

7.5 RobotControl - Application examples

7.5.1 RoCo Basics for Pick & Place Applications

General

In general, the motion sequence for the Tool Center Point (hereafter referred to as TCP) for simple Pick & Place applications is as follows.

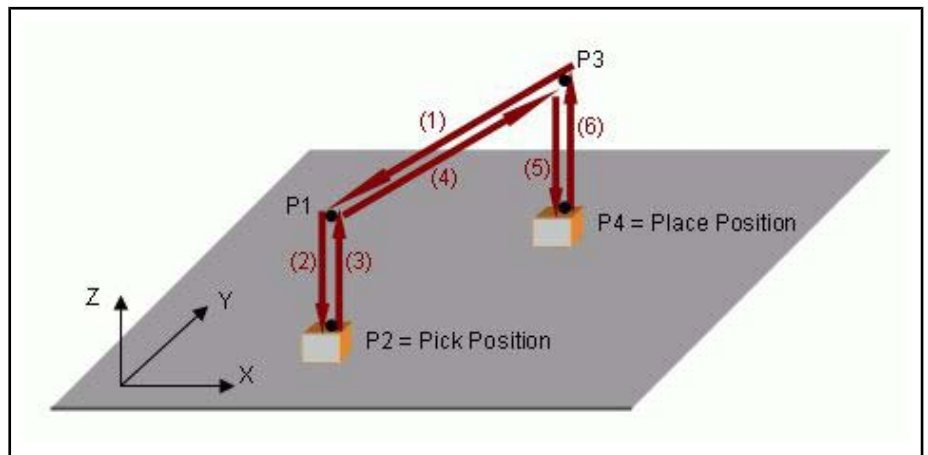


Fig. 7-17: Simple Pick & Place application

- Description**
- horizontal motion (1) above the object
 - vertical motion (2) down to the object to grip it
 - 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

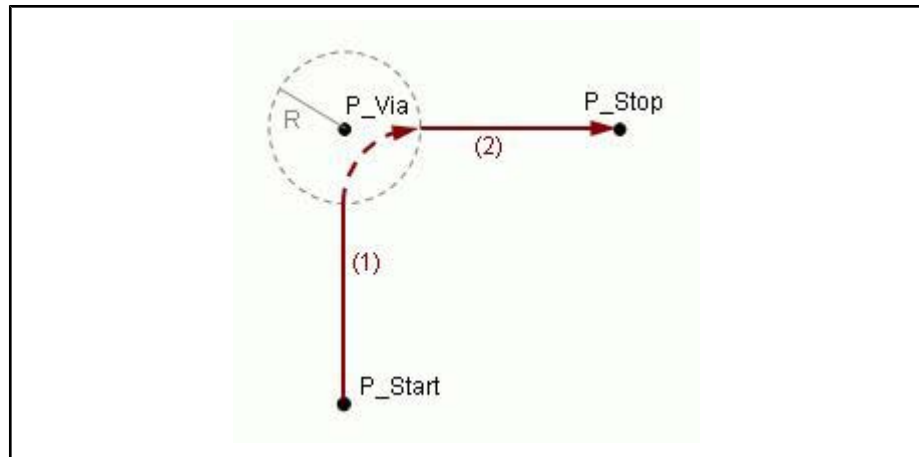
To carry out the previously traced motion sequence more harmonically and therefore more quickly, it is possible to blend the individual motion blocks between the object pick-up and placement and vice versa.



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 ball results from the superimposing of individual motions.

If the points of the blending motion block are not located on a straight line, they are not traveled through directly.

First Steps - RobotControl



R Blending radius

Fig. 7-18: 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".

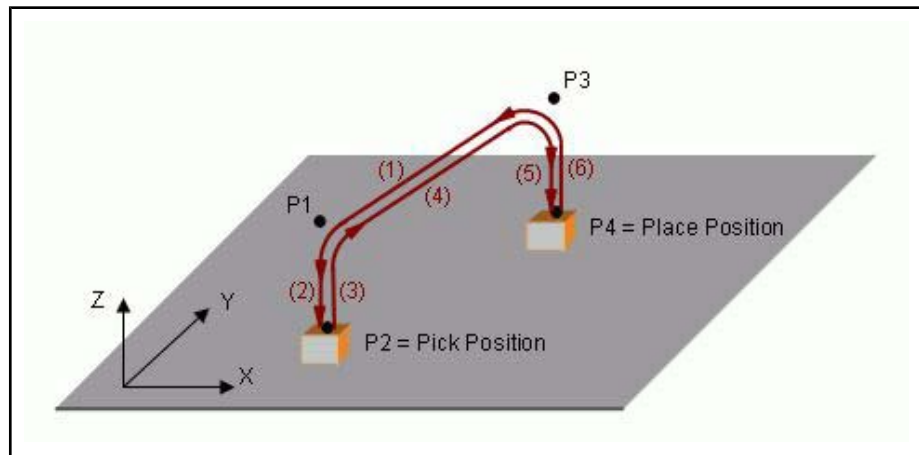


Fig. 7-19: Pick & Place application with blending motions

Description

- Horizontal motion (1) across the object, which changes immediately into vertical motion (2) down towards the object when the blending radius is entered.
- Stooping 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.
- 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 if a new object exists.
- 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 working area, the execution is delayed until the position enters the belt-synchronous working area.



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:

First Steps - RobotControl

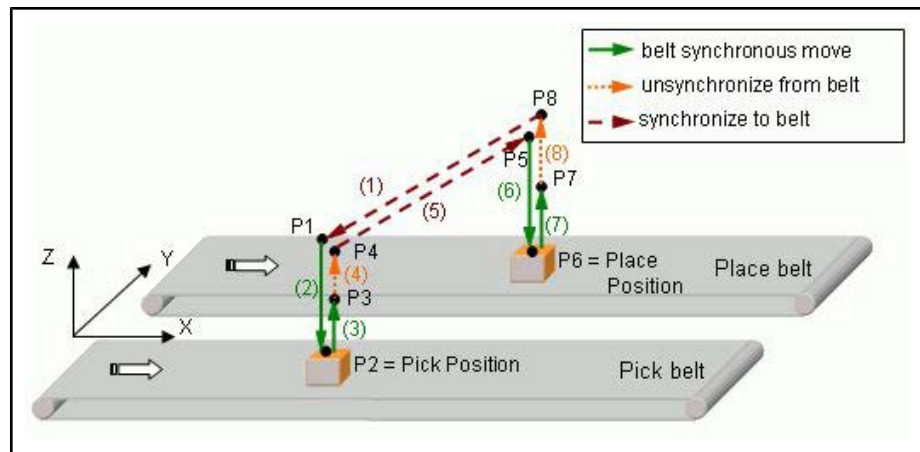


Fig.7-20: Pick & Place application with belt synchronization

Key

- 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
- 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 no 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:

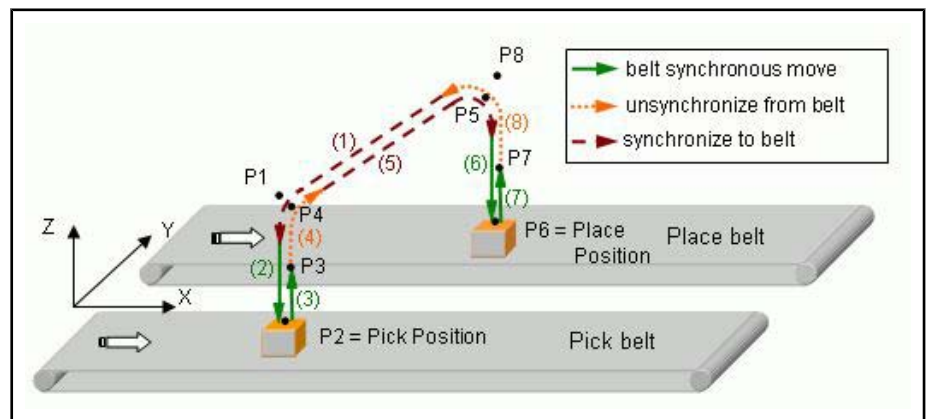


Fig. 7-21: Pick & Place application with belt synchronization and blending motions

Key

- | | |
|-----------|--|
| 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 | <ul style="list-style-type: none"> • Synchronization motion (1) on the Pick Belt across the synchronization point P1 <ul style="list-style-type: none"> – 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 <ul style="list-style-type: none"> – Gripping the object – Determining the complete path from P2 (Pick) to P6 (Place) |
|--------------------|---|

First Steps - RobotControl



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 no 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)

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 begins.
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 de-synchronization P4 or P8 (not during a synchronous phase).

Cause:

Belt counters are synchronous inputs (Belt), so they prevent blending

Implementation:

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 with 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 -
;               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
```

First Steps - RobotControl

```

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 :                     ; world coordinate point definition
  Home,                     ; home position
  PickApp,                  ; point above point Pick to synchronize to B1
  Pick,                    ; pick position
  PickDep,                 ; point above point Pick to move synchronous to
  PickDepAsyn,             ; point above POS to unsynchronize from B1
  PlaceApp,               ; point above point Place to synchronize to B2
  Place,                  ; place position
  PlaceDep,               ; point above point Place to move synchronous to
  PlaceDepAsyn            ; 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
  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.)

```

First Steps - RobotControl

```

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

;-----
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)

```

First Steps - RobotControl

```
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
; of the pick belt
@Base,
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
; of the place belt
@Base,
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
```

8 Service and Support

Our service helpdesk at our headquarters in Lohr, Germany and our worldwide service will assist you with all kinds of enquiries. You can reach us **around the clock - even on weekend and on holidays**.

	Helpdesk	Service Hotline Worldwide
Phone	+49 (0) 9352 40 50 60	Outwith Germany please contact our sales/service office in your area first.
Fax	+49 (0) 9352 40 49 41	
E-mail	service.svc@boschrexroth.de	For hotline numbers refer to the sales office addresses on the Internet.
Internet	http://www.boschrexroth.com You will also find additional notes regarding service, maintenance (e.g. delivery addresses) and training.	

Preparing Information

For quick and efficient help please have the following information ready:

- Detailed description of the fault and the circumstances
- Information on the type plate of the affected products, especially type codes and serial numbers
- Your phone, fax numbers and e-mail address so we can contact you in case of questions.

Index

A

About this documentation.....	5
Validity of the documentation	5
Appropriate use	
Use cases	11
Appropriate Use	
Introduction	11

D

Drive system.....	13
-------------------	----

E

Electric drive system.....	13
Export file	
First steps - FlexProfile	157
First steps - IndraLogic	81
First steps - Motion	141

F

First steps - FlexProfile.....	143
Create FlexProfile	145
Create FlexProfile via IndraWorks dialog ...	145
Create FlexProfile via PLC	147
Create Motion subproject	144
Download PLC subproject FlexProfile	154
Execute a motion step, ML_FlexProfile	151
Restore project	157
Technical requirements	144
Trace_FlexProfile	156
Workflow virtual axis → Virtual axis → Flex- Profile 0 → Real axis	155
First steps - IndraLogic.....	25
Add and edit an element	76
Add new device to a project	27
Basic settings to the device	28
Communication settings	29
Copy visu-element	73
Create new project	26
Create new visualization	69
Create visualization for PLC program	69
Declare global variables	56
Device settings	28
Edit AS (SFC)	45
Edit PlcProg in CFC	62
Edit PlcProg in ST	60
Edit programs and function blocks – Sec- ond part	60
Edit ST	43
Edit visualization	70
Function module settings	30
New IndraWorks project	26
Rectangle visu-element	74
Restore	81
Restoring a project	81

F

...First steps - IndraLogic	
Start IndraWorks	26
Task configuration	55
Technical requirements	25
Test in simulation mode	57
Test visualization	80
Visualization in IndraLogic	69
Visu-element button	77
Visu-element display of variable values	79
Visu-element ellipse	70
Visu-element labeling	78
First Steps - IndraLogic	
Add function block	35
Add POU	32, 36
Add program	34
Control of a traffic light system	32
Create PLC program	32
Edit FBD	37
Programs and function blocks	32
First steps - Motion.....	83
Add a virtual axis	90
Add new device to project	85
Basic settings for the device	86
Communication settings	87
Create new project	84
Device settings	86
Download of the PLX sub-project VA1 (FBD)	102
Download PLC projects RA1 (FBD)	115
Download PLC projects RA1 (ST)	138
Download the Motion subproject	91
Download the PLC project VA1 (ST)	125
Function module settings	88
Program real axis RA1 (FBD)	111
Program real axis RA1 (ST)	134
Program virtual axis VA1 (FBD)	95
Program virtual axis VA1 (ST)	119
Restore project	141
Scan SERCOS III drives	89
Start IndraWorks	84
Technical requirements	83
Trace_VA1_RA1 (FBD)	116
Trace_VA1_RA1 (ST)	140
Trace_VA1 (FBD)	106
Trace_VA1 (ST)	129
Workflow virtual axis → Electronic gear → Real axis	115, 138
Write values, input variables (FBD)	104
Write values, input variables (ST)	127
First Steps - Motion	
Program motion sequence (FBD)	93
Program motion sequence (ST)	117
First steps - RobotControl.....	159
Application examples	173

Index

F	
...First steps - RobotControl	
Create RoCo project	160
Pick & Place, belt synchronization	175
Pick & Place ,belt synchronization with blending motions	177
Pick & Place, blending	173
Pick & Place Applications	173
Required hardware and software	159
First Steps - RobotControl	
RoCoTestV**	171
 I	
Improper use.....	12
 I	
Inappropriate Use	
Consequences, Exclusion of Liability	11
 P	
PELV.....	18
Protective extra-low voltage.....	18
 S	
Safety instructions for electric drives and con- trols.....	13
Support	
see Service Hotline	183

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 (0)93 52-40-0
Fax +49 (0)93 52-48 85
www.boschrexroth.com/electrics



R911333846