

Rexroth IndraMotion MLC 14VRS

RegisterControl

Library
R911341509

Edition 01



Title Rexroth IndraMotion
MLC 14VRS
RegisterControl

Type of Documentation Library

Document Typecode DOK-MLC***-REGI*CO*V14-LI01-EN-P

Internal File Reference RS-a51833f0e93de9c00a6846a5017fafb6-1-en-US-3

Record of revision

Edition	Release date	Note
01	2014-11	First edition

Copyright © Bosch Rexroth AG 2014

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

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

Editorial department System Development Automation Motion Logic Control, HoS (FrWe/PiGe)

Table of Contents

	Page
1 About this documentation.....	3
1.1 Validity of the documentation.....	3
1.2 Required and supplementing documentation.....	3
1.3 Information representation.....	9
1.3.1 Using safety instructions.....	9
1.3.2 Descriptions.....	11
1.3.3 Abbreviations.....	11
1.4 Customer feedback.....	11
2 Important instructions on use.....	13
2.1 Intended use.....	13
2.1.1 Introduction.....	13
2.1.2 Areas of use and application.....	13
2.2 Unintended use.....	14
3 RMB_TechRegiExtended.....	15
3.1 RMB_TechRegiExtended library subject to license.....	15
3.2 General information.....	17
3.3 Function blocks to communicate with the mark stream sensor.....	18
3.3.1 Introduction and overview.....	18
3.3.2 MB_RegiSensor01ConfigType01.....	22
3.3.3 MB_RegiSensor01ConfigType01.....	39
3.3.4 MB_RegiSensor01SendRecvDataType01.....	56
3.3.5 MB_RegiSensor01CycComType01.....	59
3.3.6 MB_RegiSensor01CycComType02.....	63
3.3.7 MB_RegiSensor01ConfigType02.....	67
3.3.8 MB_RegiSensor01ConfigType02.....	85
3.3.9 MB_RegiSensor01SendRecvDataType02.....	102
3.3.10 MB_RegiSensor01CycComType04.....	106
3.3.11 MB_RegiSensor01ReadDiagType01.....	110
3.3.12 MB_RegiSensor01SendSpeedAccType01.....	112
3.3.13 MB_RegiSensor01TeachDataType01.....	117
3.3.14 MB_RegiSensor01SetIPAddressType01.....	119
3.4 Function block to create defined velocity characteristics.....	121
3.4.1 Introduction and overview.....	121
3.4.2 MB_ReachVelocityType01.....	122
4 RMB_TechRegiAdvanced.....	127
4.1 RMB_TechRegiAdvanced library subject to license.....	127
4.2 General information.....	129
4.3 Function blocks to control and optimize the printing process.....	129
4.3.1 Introduction and overview.....	129
4.3.2 MB_RegisterControllerAdvType01.....	130

Table of Contents

	Page
4.3.3 MB_RegiAccelerationCompensationType01.....	140
4.3.4 MB_RegiGravureRegulateType01.....	146
4.3.5 MB_RegiGravureRegulateType02.....	152
 5 Service and support.....	 159
 Index.....	 161

1 About this documentation

1.1 Validity of the documentation

This documentation is intended for users who want to determine the position of registration marks using an evaluation unit (mark stream sensor). The function blocks for register control (automatic controller parameterization, acceleration compensation and decoupling network) are described.

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

Example: In the product phase "Engineering", the target group "programmer" can execute the activity "parameterize" using this documentation.

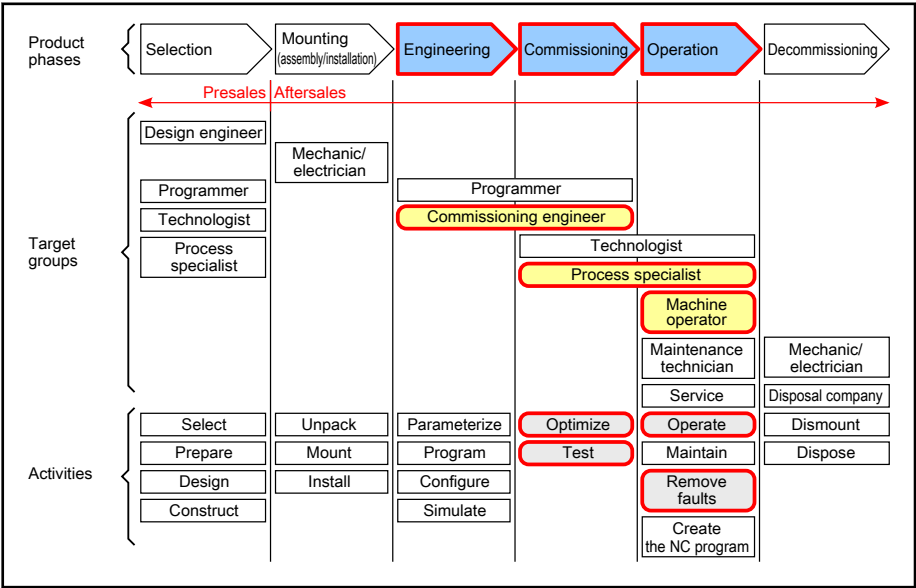


Fig. 1-1: Assigning this documentation to the target groups, product phases and target group activities

1.2 Required and supplementing documentation

Documentation titles with type codes and part numbers

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

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

About this documentation

First steps
Rexroth IndraMotion MLC 14VRS First Steps DOK-MLC***-F*STEP**V14-CORS-EN-P, R911341517 This documentation describes the first steps of the IndraMotion MLC and the RobotControl. It includes the hardware and software prerequisites as well as the creation of a project.
Rexroth IndraLogic XLC 14VRS First Steps DOK-XLC***-F*STEP**V14-CORS-EN-P, R911341489 This documentation describes the first steps of the IndraLogic XLC. It includes the hardware and software prerequisites as well as the creation of a project.
Rexroth IndraMotion MLC 14VRS Project Conversion DOK-XLCMLC-PROCONV*V14-APRS-EN-P, R911341494 This documentation describes the project conversion of the IndraMotion MLC.

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

Engineering
Rexroth IndraWorks 14VRS Software Installation DOK-IWORKS-SOFTINS*V14-CORS-EN-P, R911344286 This documentation describes the IndraWorks installation.
Rexroth IndraWorks 14VRS Engineering DOK-IWORKS-ENGINEE*V14-APRS-EN-P, R911343566 This documentation describes the application of IndraWorks in which the Rexroth Engineering tools are integrated. It includes instructions on how to work with IndraWorks and how to operate the oscilloscope function.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Function Description DOK-XLCMLC-FUNC****V14-APRS-DE-P, R911341700 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraMotion MLC.

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

Diagnostics and service
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481 This documentation includes all control parameters implemented in the control systems IndraLogic XLC and IndraMotion MLC.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Parameters DOK-XLCMLC-PARAM***V14-RERS-EN-P, R911341479 This documentation describes the parameters of the XLC/MLC systems as well as the interaction between parameterization and programming. It includes the axis parameters, control parameters, kinematic parameters, touch probe parameters and programmable limit switch parameters.

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

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

Drive controllers
Rexroth IndraDrive Firmware for Drive Controllers MPH-, MPB-, MPD-, MPC-08 DOK-INDRV*-MP*-08VRS**-APxx-EN-P-DE-P, R911332643 This documentation describes all functional properties of the IndraDrive firmware in the variants MPH-08, MPB-08, MPD-08 and MPC-08.
Rexroth IndraDrive Drive Controllers MPx-02 to MPx-08 DOK-INDRV*-GEN-**VRS**-PAxx-EN-P, R911297317 This documentation describes all parameters implemented in the firmware for drive controllers of the IndraDrive family. It supports the parameterization of the drive controllers.
Rexroth IndraDrive MPx-02 to MPx-08 and HMV DOK-INDRV*-GEN-**VRS**-WAxx-EN-P, R911297319 This documentation describes all diagnostics implemented in the following firmwares: • Drive controller firmwares from MPx-02 to MPx-08 and • firmwares of the supply devices of the type "HMF". It supports the operating crew as well as the programmer during troubleshooting.
Rexroth IndraDrive Control Units CSB02, CSE02, CSH02, CDB02 DOK-INDRV*-CXX02*****-PRRS-EN-P, R911338962
Rexroth IndraDrive MPx-18 Functions DOK-INDRV*-MP*-18VRS**-APRS-EN-P, R911338673 This documentation describes all functional properties of the IndraDrive firmware in the variants MPB-18, MPM-18, MPC-18 and MPE-18.

xx Corresponding edition

Tab. 1-5: XCL/MLC documentation overview - Drive controllers

About this documentation

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

Tab. 1-6: XCL/MLC documentation overview - PLC

Field buses
Rexroth IndraWorks 14VRS Field Buses DOK-IWORKS-FB*****V14-APRS-EN-P, R911341485 This documentation describes the field bus and local periphery connections supported by the IndraLogic XLC, IndraMotion MLC and IndraMotion MTX systems. The focus of this documentation is on the configuration, parameterization, commissioning and the diagnostics of the different periphery connections. This documentation is the basis for the online help.
Rexroth IndraWorks 14VRS Field Buses Libraries DOK-IWORKS-FB*LIB**V14-LIRS-EN-P, R911343575 This documentation describes the field bus libraries: RIL_ProfibusDP_02, RIL_ProfibusDP Slave, RIL_ProfinetIO, RIL_ProfinetIODevice, RIL_EtherNetIPAdapter, RIL_MappingList, RIL_SERCOSIII, RIL_Inline including their diagnostics and error reactions of the function blocks.
Rexroth IndraWorks 12VRS FDT Container DOK-IWORKS-FDT*CON*V12-APRS-EN-P, R911334398 This documentation describes the IndraWorks FDT Container functionality. It includes the activation of the functionality in the project and working with DTMs.
SERCOS System Manual for I/O Devices DOK-CONTRL-ILS3*****-APxx-EN-P, R911333512 This documentation describes the configuration, parameterization, commissioning and diagnostics of I/O devices with a SERCOS interface.

xx Corresponding edition

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

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

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

Technology
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Generic Application Template DOK-XLCMLC-TF*GAT**V14-APRS-EN-P, R911341487 This documentation provides a structured template to the IndraLogic PLC programmer. This template can be used to add and edit the PLC programming code. It includes the template, the template wizard and example applications.
Rexroth IndraMotion MLC 14VRS Technology Libraries DOK-MLC***-TF*LIB**V14-LIRS-EN-P, R911341511 This documentation describes the function blocks, functions and data types of the "ML_TechInterface.library", "ML_TechMotion.library", "RMB_TechCam.library" and "ML_TechBase.library". It also includes libraries for the winder functionality, register controller functionality and CrossCutter functionality.
Rexroth IndraWorks Energy Efficiency Management 14VRS DOK-IWORKS-4EE*****V14-APRS-EN-P, R911339229 This documentation describes the HMI connection of the energy management system and the function blocks, functions and data types of the "RIL_4EE " library. The documentation also includes the error reactions of the function blocks.
Rexroth IndraMotion MLC 14VRS RegisterControl (Library) DOK-MLC***-REGI*CO*V14-LIRS-EN-P, R911341509 This documentation describes the inputs and outputs of the individual function blocks and provides notes on their use.
Rexroth IndraMotion MLC 14VRS RegisterControl (Application Description) DOK-MLC***-REGI*CO*V14-APRS-EN-P, R911341507 This documentation describes the application of the integrated register control for a rotogravure printing machine. The components of the mark stream sensor, the HMI application and the error recovery options are described. This instruction provides information on how to operate the register control, react to errors and query diagnostics. This documentation is written for machine setters and machine operators.
Rexroth IndraMotion MLC 14VRS Winder Function Application DOK-MLC***-TF*WIND*V14-APRS-EN-P, R911341513 This application-related system documentation describes the application of the winder technology functions.

About this documentation

Technology
Rexroth IndraWorks 13VRS CamBuilder DOK-IWORKS-CAMBUIL*V13-APRS-EN-P, R911336291 This documentation describes the basic principles and operation of the CamBuilder, the cam editing tool.
Rexroth IndraMotion MLC 14VRS Robot Control V2 DOK-MLC***-ROCO****V14-RERS-EN-P, R911341588 This documentation provides information about the Robot Control V2. The focus is on PLCopen programming. The program structure, variables, functions, motion statements and the required system parameters are described. This documentation also includes the description of the "ML_Robot" library. Robot Control V2 is supported by a kinematic interface. The components of this interface are described in the "ML_KinTech" library.

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

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

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

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

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

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

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

Hydraulics
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Sequential Programming DOK-XLCML-SEQPROG*V14-LIRS-ENE-P, R911341909 This documentation includes the library for the sequential programming.
Rexroth IndraMotion MLC 13VRS MH_TechHydrBase DOK-MLC***-TF*HBASEV13-LIRS-EN-P, R911336312 This documentation describes the function blocks, functions and data types of the MH_TechHydrBase libraries. The documentation also includes the error reactions of the function blocks.
Rexroth IndraMotion MLC 14VRS MH_TechHydrMotion DOK-MLC***-TF*HMOT*V14-LIRS-EN-P, R911341906 This documentation describes the function blocks, functions and data types of the MH_TechHydrMotion libraries. The documentation also includes the error reactions of the function blocks.
Rexroth IndraMotion MLC 13VRS MH_HydrControl DOK-MLC***-TFHCLIB*V13-LIRS-EN-P, R911336328 This documentation describes the function blocks, functions and data types of the MH_HydrControl library. The documentation also includes the error reactions of the function blocks.
Rexroth IndraMotion MLC 14VRS MH_SynchControl DOK-MLC***-TFHSLIB*V14-LIRS-EN-P, R911341911 This documentation describes the function blocks, functions and data types of the MH_SynchControl library. The documentation also includes the error reactions of the function blocks.

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

1.3 Information representation

1.3.1 Using safety instructions

Safety instructions – Structure

The safety instructions are structured as follows:

About this documentation

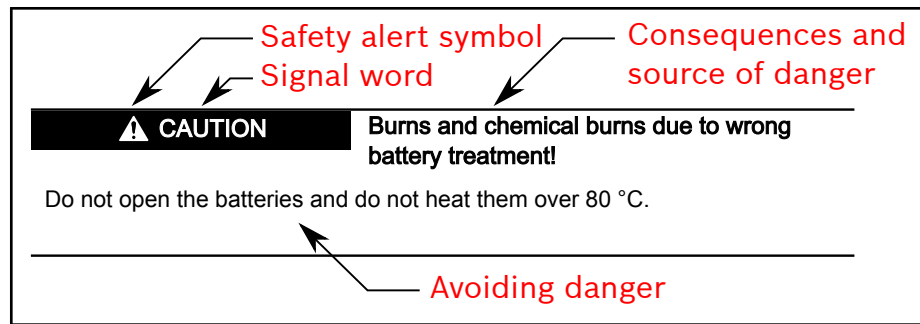


Fig. 1-2: Safety instructions – structure

Explaining signal words and safety alert symbol

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

The signal word is used to draw attention to the safety instruction and also provides information on the severity of the hazard.

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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

Symbols used

Pointers are displayed as follows:



This is a note.

Tips are displayed as follows:



This is a tip.

1.3.2 Descriptions

Designation	Explanation
Runtime license	A check for the license is carried out at runtime. For each control, on which the function blocks are used, one license is required (runtime license).
Mark stream sensor	Evaluation unit, sampling a mark stream and measuring the mark positions.
Teach-In	Process during which the mark stream sensor measures the threshold values for the mark measurement.
Measurement	Mark positions of the defined marks are determined and send to the control.
Bar code	Sequence of at least 3 blocks that can be found automatically by the sensor.
S-curve	Jerk-limited acceleration phase

Tab. 1-14: Used descriptions

1.3.3 Abbreviations

Abbreviation	Significance
OWG	Optical waveguide
i.e.	That is
mm	Millimeter
e.g.	For example
n.def.	Not defined
cf.	Compare
MLC	MotionLogic Control
s	Seconds
m	Meter
%	Percent

Tab. 1-15: Used abbreviations

1.4 Customer feedback

Customer requests, comments or suggestions for improvement are of great importance to us. Please email your feedback on the documentations to

About this documentation

Feedback.Documentation@boschrexroth.de. Directly insert comments in the electronic PDF document and send the PDF file to Bosch Rexroth.

2 Important instructions on use

2.1 Intended use

2.1.1 Introduction

Bosch Rexroth products are developed and manufactured according to the state-of-the-art. The products are tested prior to delivery to ensure operating safety and reliability.

The products may only be used as intended. If they are not used as intended, situations occur that can result in damage to property and injury to persons.



Bosch Rexroth shall not assume any warranty, liability or payment of damages in case of damage resulting from a non-intended use of the products; the use shall solely bear all risks from unintended use of the products.

Before using Bosch Rexroth products, the following requirements have to be met to guarantee the intended use of the products:

- Anybody handling Bosch Rexroth products in any way is obliged to read and consent to the relevant safety instructions and the intended use.
- The original condition of hardware products may not be altered; in other words, no structural modifications are permitted. It is not permitted to decompile software products or alter source codes.
- Do not install damaged or defective products or use them in operation.
- It has to be ensured that the products have been installed as described in the relevant documentation.

2.1.2 Areas of use and application

Rexroth IndraLogic XLC/
IndraMotion MLC

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



The IndraLogic XLC/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 applies to cables and lines.

It may only be operated with the hardware component configurations and combinations expressly specified and with the software and firmware indicated and specified in the respective documentation and functional descriptions.

The IndraLogic XLC/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 areas of application for IndraLogic XLC/IndraMotion MLC and its function modules are:

- Handling and mounting systems
- Packaging and food machines
- Printing and processing machines

Important instructions on use

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

2.2 Unintended use

Using the Rexroth IndraLogic XLC/IndraMotion MLC and its function modules outside of the above-referenced areas of application or under operating conditions other than described in the document and the technical data specified is defined as "unintended".

The Rexroth IndraLogic XLC/IndraMotion MLC and its function modules may not be used if

- subjected to operating conditions that do not comply with the specified ambient conditions. For example, operation under water, under extreme temperature fluctuations or extreme maximum temperatures is prohibited.
- Furthermore, the IndraLogic XLC/IndraMotion MLC and its function modules must not be used in any applications not expressly approved by Bosch Rexroth. Therefore, please read the information given in the general safety instructions!

3 RMB_TechRegiExtended

3.1 RMB_TechRegiExtended library subject to license



The use of the functions blocks of the RMB_TechRegiExtended library is subject to a license. Depending on the application, one of the three licenses, either FWS-IM*MLC-TEC-NNVRS-NN-REGI*EXT, FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV1 or FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV2 has to be purchased to use the function blocks. For each control on which the function blocks are used, one license is required each (runtime license).

Einzellizenz Technologiebibliothek
Berechtigt zur Verwendung mit einer einzigen Steuerungsfirmware.

Single licence for technology library
Authorize the use with exactly one controller firmware.

TechLib RegisterControl Extended TechLib RegisterControl Extended		
Licence key (MNR): R911xxxxxx	Serial number: 1234567	Type code: FWS-IM*MLC-TEC-NNVRS-NN-REGI*EXT
		
Registration key WAHOF2-FBJYZF-8CSB0F-9DQO0D-9E8ZER		

Fig. 3-1: FWS-IM*MLC-TEC-NNVRS-NN-REGI*EXT license key

Einzellizenz Technologiebibliothek
Berechtigt zur Verwendung mit einer einzigen Steuerungsfirmware.

Single licence for technology library
Authorize the use with exactly one controller firmware.

TechLib RegisterControl Advanced 1 TechLib RegisterControl Advanced 1		
Licence key (MNR): R911xxxxxx	Serial number: 1234567	Type code: FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV1
		
Registration key WAHOF2-FBJYZF-8CSB0F-9DQO0D-9E8ZER		

Fig. 3-2: FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV1 license key

RMB_TechRegiExtended

Einzellizenz Technologiebibliothek Berechtigt zur Verwendung mit einer einzigen Steuerungsfirmware.		
Single licence for technology library Authorize the use with exactly one controller firmware.		
TechLib RegisterControl Advanced 2 TechLib RegisterControl Advanced 2		
Licence key (MNR): R911xxxxxx	Serial number: 1234567	Type code: FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV2
		
Registration key WAHOF2-FBJYZF-8CSB0F-9DQO0D-9E8ZER		

Fig. 3-3: FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV2 license key

The figures show the corresponding license keys. One license has to be purchased for each control authorizing the use of function blocks of the RMB_TechRegiExtended library or the RMB_TechRegiAdvanced on this control system.

The libraries RMB_TechRegiExtended and RMB_TechRegiAdvanced are provided with the three licenses FWS-IM*MLC-TEC-NNVRS-NN-REGI*EXT, FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV1 and FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV2. The content of the licenses is shown in the following:

*FWS-IM*MLC-TEC-NNVRS-NN-REGI*EXT*

- HMI for narrow web applications (up to twelve printing units, no decoupling and acceleration compensation)
- Firmware Update Tool
- Documentations: Application Manual and Library Description
- Libraries: RMB_TechRegiExtended and RMB_TechRegiAdvanced

*FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV1*

- HMI for applications with clamping units (such as gravure printing) (up to twelve printing units, with decoupling and acceleration compensation)
- Firmware Update Tool
- Documentations: Application Manual and Library Description
- Libraries: RMB_TechRegiExtended and RMB_TechRegiAdvanced

*FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV2*

- HMI for applications with clamping units (such as gravure printing) (up to 24 printing units, with decoupling and acceleration compensation)
- Firmware Update Tool
- Documentations: Application Manual and Library Description
- Libraries: RMB_TechRegiExtended and RMB_TechRegiAdvanced

If either the RMB_TechRegiExtended library or the RMB_TechRegiAdvanced library is used, one of the licenses FWS-IM*MLC-TEC-NNVRS-NN-REGI*EXT, FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV1 or FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV2 is required for each control on which one of the function blocks is used.

3.2 General information

The RMB_TechRegiExtended library provides function blocks required to communicate with the mark stream sensor. The mark stream sensor consists of an evaluation unit connected with an optics via optical fiber cables.



Fig. 3-4: Mark stream sensor

The mark stream sensor determines mark positions of up to 12 registration marks simultaneously and returns their measured values. Apart from the hardware of the mark stream sensor, it is a library that has to be purchased. One license is required for each control system used.

To use the functionality of the sensor, the following is required:

- 1 evaluation unit of the mark stream sensor
- 1 sensor optics
- 1 cable set (2 optical fibers, 1 Ethernet cable, 1 incremental encoder cable and 1 electricity cable)
- 1 drive system with MEM card

This documentation describes the inputs and outputs of the individual function blocks and provides notes on their use.

The following overview shows all function blocks of the RMB_TechRegiExtended library requiring a license:

Function block	Description
Function blocks for sensor firmware version <= 11	
MB_RegiSensor01CommType01	Closure function block, communicating with the mark stream sensor up to sensor firmware 11
MB_RegiSensor01ConfigType01	Mark stream sensor configuration up to sensor firmware 11
MB_RegiSensor01SendRecvDataType01	Executes an acyclic communication with the mark stream sensor up to sensor firmware 11
MB_RegiSensor01CycComType01	Evaluates the mark distances and widths of two marks up to sensor firmware 11
MB_RegiSensor01CycComType02	Evaluates all mark distances to the zero pulse as well as all mark widths up to sensor firmware 11
Function block from sensor firmware >= 12	
MB_RegiSensor01CommType02	Closure function block, communicating with the mark stream sensor from sensor firmware 12
MB_RegiSensor01ConfigType02	Mark stream sensor configuraton from sensor firmware 12

RMB_TechRegiExtended

Function block	Description
MB_RegiSensor01SendRecvDataType02	Executes an acyclic communication with the mark stream sensor from sensor firmware 12
MB_RegiSensor01CycComType04	Evaluates all mark distances to the zero pulse as well as all mark widths from sensor firmware 12
Function blocks irrespective of the sensor firmware version	
MB_RegiSensor01SendSpeedAccType01	Sends velocity information to the mark stream sensor
MB_RegiSensor01ReadDiagType01	Reads diagnostic data of the mark stream sensor
MB_RegiSensor01TeachDataType01	Reads/writes the latest valid measured data of the mark stream sensor
MB_RegiSensor01SetIPAddressType01	Sets the IP address of the marks stream sensor
MB_ReachVelocityType01	Calculates different velocity characteristics

Tab. 3-1: *Function blocks of the RMB_TechRegiExtended library that requires a license*



The function blocks MB_RegiSensor01SendSpeedAccType01, MB_RegiSensor01CommType01 and MB_RegiSensor01CommType02 have to be called in a motion-synchronous task to ensure a correction calculation of the acceleration. If this is not possible due to the runtime, the "mMotionCyclic()" function of the two function blocks from the Motion-synchronous tasks has to be called and either the MB_RegiSensor01SendSpeedAccType01, MB_RegiSensor01CommType01 or MB_RegiSensor01CommType02 function blocks can be called in a cyclic task.

3.3 Function blocks to communicate with the mark stream sensor

3.3.1 Introduction and overview

MB_RegiSensor01CommType01 sensor firmware version <= 11

It is recommended to exclusively use the MB_RegiSensor01CommType01 and MB_RegiSensor01CommType02 function blocks as the internal interconnection has been implemented. The MB_RegiSensor01CommType01 function block communicates with a mark stream sensor with a sensor firmware version <= 11. The MB_RegiSensor01CommType02 function block communicates with a mark stream sensor with a sensor firmware version > 12.

The function block contains the individual function blocks MB_RegiSensor01ConfigType01, MB_RegiSensor01CycComType02, MB_RegiSensor01ReadDiagType01, MB_RegiSensor01SendRecvDataType01, MB_RegiSensor01SendSpeedAccType01 and MB_RegiSensor01TeachDataType01. I.e., the individual function blocks are internally connected in the MB_RegiSensor01CommType01 function block. Thus, the sensor configuration and the measurement are more user-friendly.

The function blocks are internally connected in the function block and provide a user-friendly interface to execute the individual functions. The mark stream sensor can be completely controlled using the function block. Only the MB_RegiSensor01SetIPAddressType01 function block for setting the IP address is not integrated as it is only required once per machine. Consequently, the communication with the mark stream sensor is rendered more user-friendly, as described in the following.

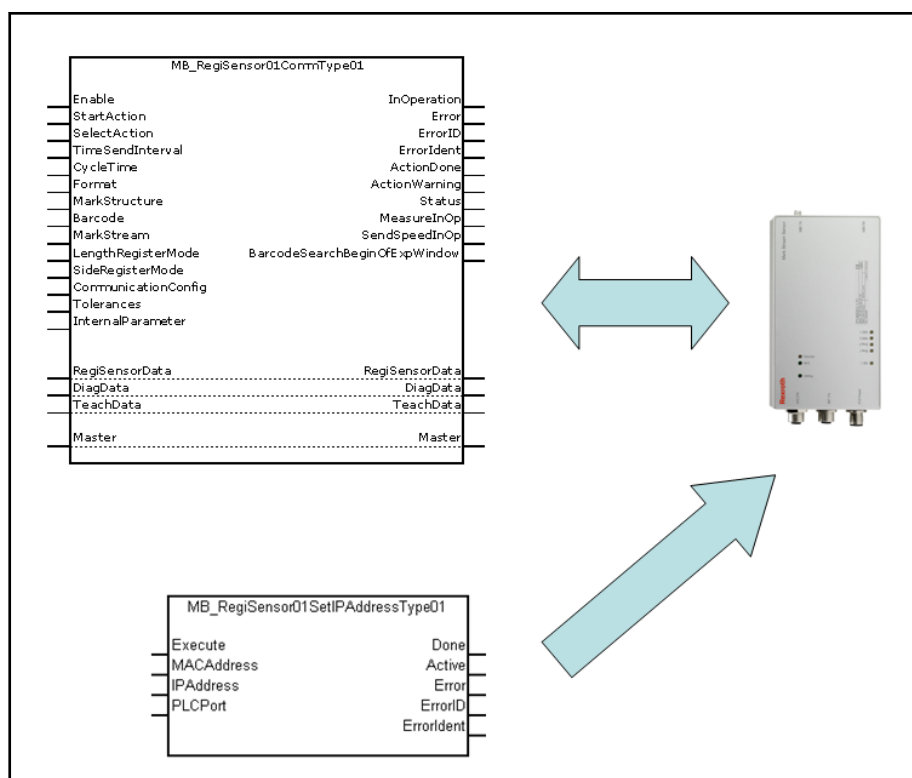


Fig. 3-5: Communication structure with the MB_RegiSensor01CommType01 function block

Only one MB_RegiSensor01CommType01 function block and the MB_RegiSensor01SetIPAddressType01 function block is required. The previously specified connection as well as the connecting order are not required anymore.



Due to a fixed, specified internal connection and the simple operation, it is recommended to only use the MB_RegiSensor01CommType01 function block up to sensor firmware version 11 for the mark stream sensor communication.

MB_RegiSensor01CommType02 sensor firmware version ≥ 12

The function block contains the individual function blocks MB_RegiSensor01ConfigType02, MB_RegiSensor01CycComType04, MB_RegiSensor01ReadDiagType01, MB_RegiSensor01SendRecvDataType02, MB_RegiSensor01SendSpeedAccType01 and MB_RegiSensor01TeachDataType01. I.e., the individual function blocks are internally connected in the MB_RegiSensor01CommType01 function block. Thus, the sensor configuration and the measurement are more user-friendly.

The function blocks are internally connected in the function block and provide a user-friendly interface to execute the individual functions. The mark stream sensor can be completely controlled using the function block. Only the MB_RegiSensor01SetIPAddressType01 function block for setting the IP address is not integrated as it is only required once per machine. Consequently, the communication with the mark stream sensor is rendered more user-friendly, as described in the following.

RMB_TechRegiExtended

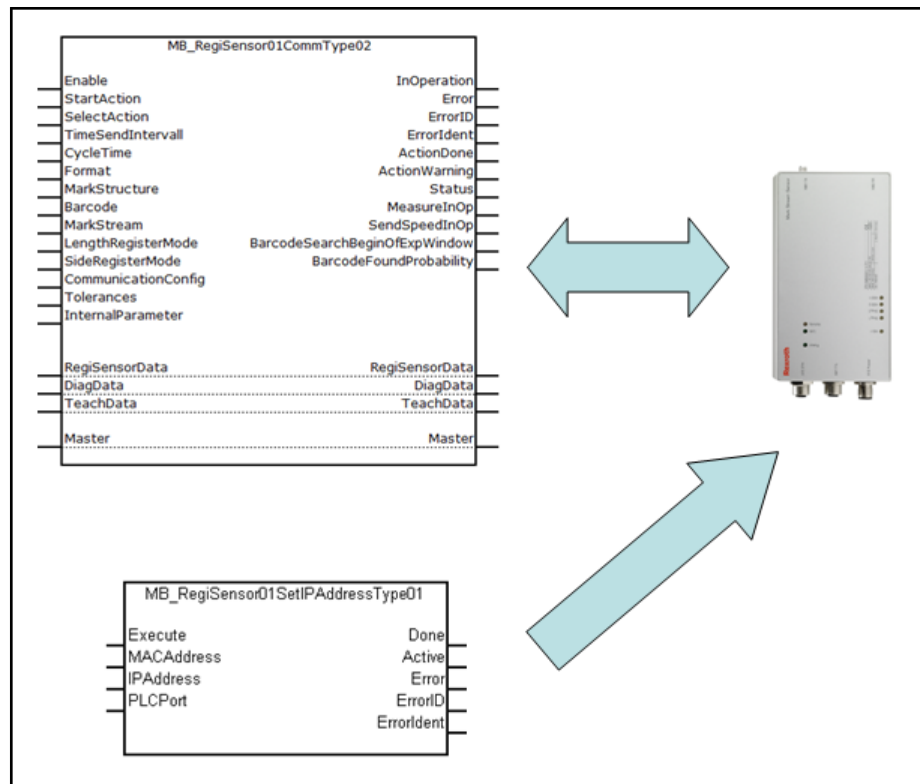


Fig. 3-6: Communication structure with the MB_RegiSensor01CommType02 function block

Only one MB_RegiSensor01CommType02 function block and the MB_RegiSensor01SetIPAddressType01 function block is required. The previously specified connection as well as the connecting order are not required anymore.



Due to a fixed, specified internal connection and the simple operation, it is recommended to only use the MB_RegiSensor01CommType02 function block for the mark stream sensor communication from sensor firmware version 12.

By way of example, the manual interconnection of the function blocks for sensor firmware ≤ 11 is described. However, this procedure is not recommended. It is distinguished between cyclic and acyclic communication. Acyclic communication means that the control is the client and sensor is the server. Thus, the control requests data and the sensor responds. This includes configuration data, process data and diagnostic data.

The following function blocks are part of the acyclic communication:

- MB_RegiSensor01ConfigType01: Sensor configuration, measurement start, teach-in, etc.
- MB_RegiSensor01SendSpeedAccType01: Sends information on velocity and acceleration
- MB_RegiSensor01SendRecvDataType01: It is responsible for the acyclic communication with the sensor
- MB_RegiSensor01ReadDiagType01: Reads the diagnostic data from the sensor
- MB_RegiSensor01TeachDataType01: Reads/writes the latest valid measured data

RMB_TechRegiExtended

Each sensor requires two ports on the control. One port is for the acyclic communication and one port is for the cyclic communication. In the cyclic communication, the sensor sends a data package after each cylinder revolution. This data package contains the measured data defined in the configuration. The control listens on this port and evaluates the measurement results.

Therefore, the following function blocks are used:

- MB_RegiSensor01CycComType01: Evaluates the mark distances and widths of two marks with regard to each other.
- MB_RegiSensor01CycComType02: Evaluates the distances of all marks to the zero pulse as well as all mark widths.

The function block MB_RegiSensor01CycComType01 provides the exact mark distance between the two marks specified and their difference in mark width. The MB_RegiSensor01CycComType02 function block provides all distances of the configured marks to the zero pulse as well as their mark widths. The function blocks can be exchanged with each other depending on the use case. The following communication structure results for each sensor:

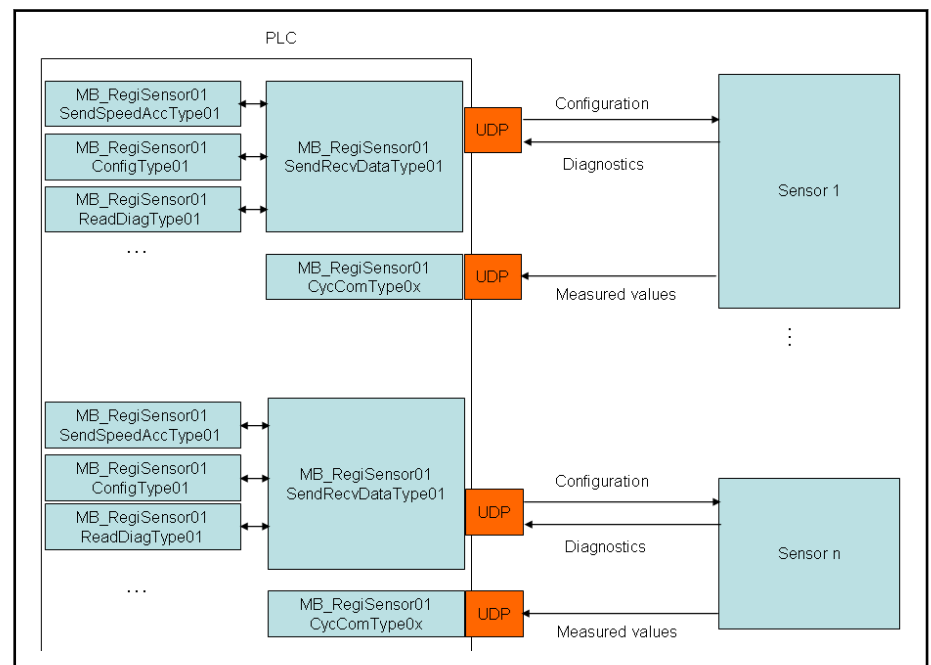


Fig. 3-7: Communication structure

Thus, it can be seen that one function block executes the entire acyclic communication and that it sends all data packages of the other function blocks to the sensor. That means that the MB_RegiSensor01ConfigType01 function block absolutely requires the MB_RegiSensor01SendRecvDataType01 function block to send the data packages.

Another function block in the RMB_TechRegiExtended to set the IP address is:

- MB_RegiSensor01SetIPAddressType01

Only one function block is used per machine.

The following figure shows the minimum wiring of function blocks and their switch-on sequence.

RMB_TechRegiExtended

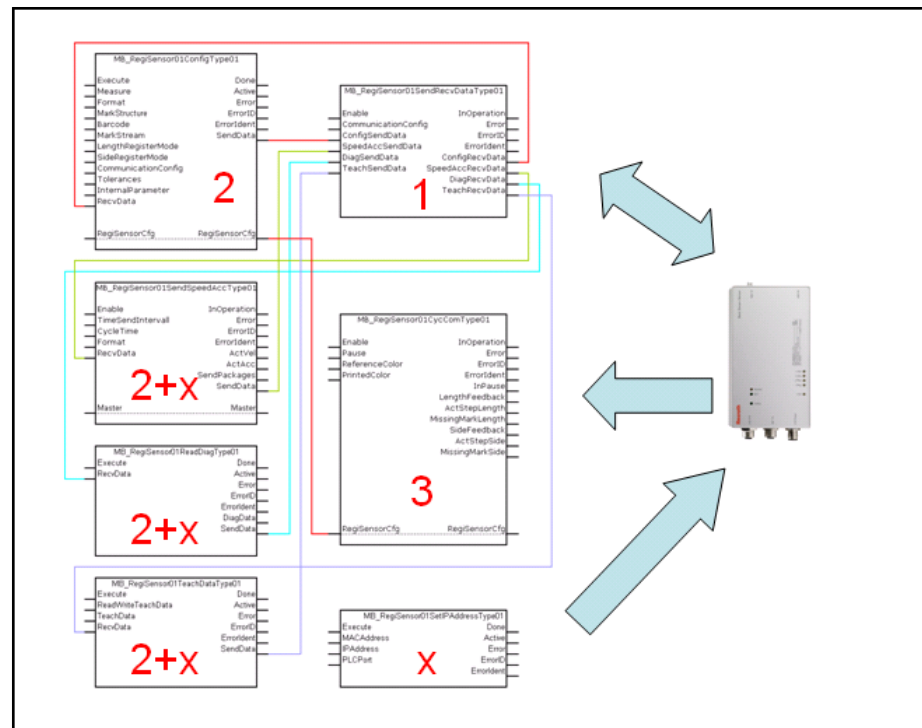


Fig. 3-8: Minimum wiring and switch-on sequence.

To execute a measurement with the mark stream sensor, the MB_RegiSensor01SendRecvDataType01 function block has to be activated first. Subsequently, the MB_RegiSensor01ConfigType01 FB can be used to send a configuration to the sensor and to start the measurement and teach-in. The function blocks MB_RegiSensor01SendSpeedAccType01, MB_RegiSensor01ReadDiagType01 and MB_RegiSensor01TeachDataType01 are not absolutely required for a measurement. They can be activated as desired. The only prerequisite is that the MB_RegiSensor01SendRecvDataType01 has already been activated. If a measurement has been started at the sensor, the on of the function blocks MB_RegiSensor01CycComType01 or MB_RegiSensor01CycComType02 has to be subsequently activated to receive the measured values. The MB_RegiSensor01SetIPAddressType01 function block to set the IP address is only required once for each machine and is thus provided with an individual UDP port. It can be activated at any point in time without being dependent on other function blocks. The colored lines indicate the minimum wiring between the function blocks to allow a correct communication with the sensor.

3.3.2 MB_RegiSensor01ConfigType01

Brief description

The MB_RegiSensor01CommType01 function block contains the individually implemented function blocks:

- MB_RegiSensor01ConfigType01
- MB_RegiSensor01CycComType02
- MB_RegiSensor01ReadDiagType01
- MB_RegiSensor01SendRecvDataType01
- MB_RegiSensor01SendSpeedAccType01 and
- MB_RegiSensor01TeachDataType01

RMB_TechRegiExtended

The function blocks are internally connected in the function block and provide a user-friendly interface to execute the individual functions. The function block facilitates the full control of the mark stream sensor.



The following function block to set the IP address is not integrated as it is only required once per machine:

- MB_RegiSensor01SetIPAddressType01



The MB_RegiSensor01CommType01 can only be used together with the sensor firmware 11 VRS and smaller.

From sensor firmware 12, the MB_RegiSensor01CommType02 function block has to be used.



The function block MB_RegiSensor01CommType01 has to be called in a Motion-synchronous task to ensure a correction calculation of the acceleration. If this is not possible due to the runtime, the "mMotionCyclic()" function of the function block from the Motion-synchronous tasks has to be called and the MB_RegiSensor01CommType01 function block can be called in a cyclic task. The call of the MotionTask has to be provided with the following structure: {CyclicTask}.fbRegiSensor01CommType01.{fbInstanz}.mMotionCyclic(). The elements in the curly brackets have to be replaced by the program name in the cyclic task and the instance name of the function block. Thus, the method call from a program running in the Motion task could be as follows: PlcProg.fbRegiSensor01CommType01.fbRegiSensor01CommType01.mMotionCyclic(). If this call is not executed, a correct calculation of the acceleration cannot be ensured.

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V12	RMB_TechRegiExtended

Tab. 3-2: Reference table of the MB_RegiSensor01CommType01 function block

Interface description

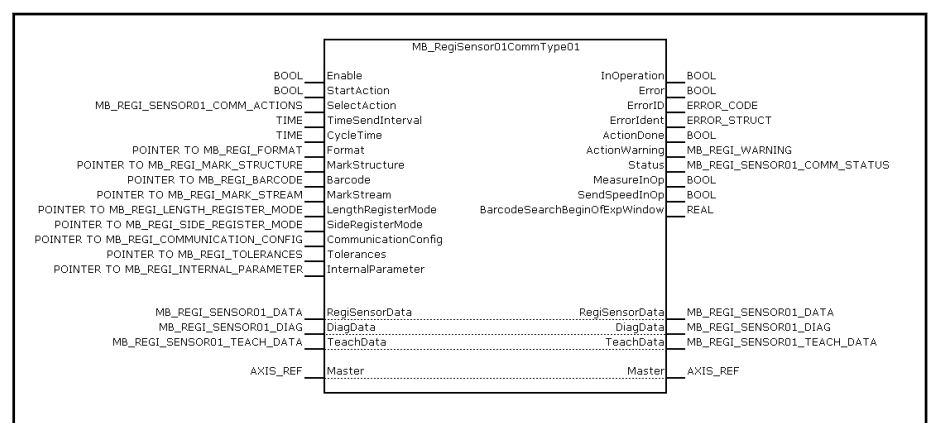


Fig. 3-9: MB_RegiSensor01CommType01 function block

RMB_TechRegiExtended

I/O type	Name	Data type	Explanation
VAR_IN_OUT CONST	Master	AXIS_REF	Addressing the master axis
VAR_IN_OUT	RegiSensor-Data	MB_REGI_SENSOR01_DATA	Contains the respective measured values of the sensor
	DiagData	MB_REGI_SENSOR01_DIAG	Diagnostic data of the sensor
	TeachData	MB_REGI_SENSOR01_TEACH_DATA	Sensor data to be read or written
VAR_INPUT	Enable	BOOL	Processing of function block enabled (level-controlled)
	StartAction	BOOL	Starts the selected action
	SelectAction	MB_REGI_SENSOR01_COMM_ACTIONS	Selecting the executable actions
	TimeSendInterval	TIME	Data of the velocity and acceleration information is calculated and sent in this interval
	CycleTime	TIME	Cycle time at which the function block is called.
	Format	POINTER TO MB_REGI_FORMAT	Defines the format characteristics
	MarkStructure	POINTER TO MB_REGI_MARK_STRUCTURE	Defines the mark structure
	Bar code	POINTER TO MB_REGI_BARCODE	Defines the bar code
	MarkStream	POINTER TO MB_REGI_MARK_STREAM	Defines the mark stream
	LengthRegisterMode	POINTER TO MB_REGI_LENGTH_REGISTER_MODE	Defines the web measuring principle for the length register
	SideRegisterMode	POINTER TO MB_REGI_SIDE_REGISTER_MODE	Defines the web measuring method for the length register
	CommunicationConfig	POINTER TO MB_REGI_COMMUNICATION_CONFIG	Defines the communication characteristics of the UDP connection
	Tolerances	POINTER TO MB_REGI_TOLERANCES	Defines the tolerances allowed for teaching and measurement
	InternalParameter	POINTER TO MB_REGI_INTERNAL_PARAMETER	Internal sensor parameters that should only be changed if specifically mentioned.

I/O type	Name	Data type	Explanation
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in the event of an error
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics
	ActionDone	BOOL	Action was completed successfully
	ActionWarning	MB_REGI_WARNING	Contains warnings of failed actions or warnings of the sensor system that do not result in a function block error
	Status	MB_REGI_SENSOR01_COMM_STATUS	Returns the current MB_RegiSensor01Comm-Type01 function block status
	MeasurementOp	BOOL	Measurement running
	SendSpeedOp	BOOL	Transmitting velocity and acceleration values.
	Barcode-SearchBeginOfExpWindow	REAL	This value contains the position of the bar code end

Tab. 3-3: I/O interface of the MB_RegiSensor01ConfigType01 function block

Data structures

The following data structures have been described in the previous function blocks of the RMB_TechRegiExtended library:

- [tab. 3-15 "MB_REGI_FORMAT data structure" on page 41](#)
- [tab. 3-16 "MB_REGI_MARK_STRUCTURE data structure" on page 41](#)
- [tab. 3-17 "MB_REGI_BARCODE data structure" on page 42](#)
- [tab. 3-18 "MB_REGI_MARK_STREAM data structure" on page 42](#)
- [tab. 3-19 "MB_REGI_LENGTH_REGISTER_MODE data structure" on page 42](#)
- [tab. 3-20 "MB_REGI_SIDE_REGISTER_MODE data structure" on page 42](#)
- [tab. 3-21 "MB_REGI_COMMUNICATION_CONFIG data structure" on page 42](#)
- [tab. 3-22 "MB_REGI_TOLERANCES data structure" on page 43](#)
- [tab. 3-23 "MB_REGI_INTERNAL_PARAMETER data structure" on page 43](#)
- [tab. 3-41 "MB_REGI_SENSOR01_DATA data structure" on page 65](#)
- [tab. 3-89 "MB_REGI_SENSOR01_DIAG data structure" on page 111](#)
- [tab. 3-98 "MB_REGI_SENSOR01_TEACH_DATA data structure" on page 118](#)

RMB_TechRegiExtended

MB_REGI_WARNING		
Warning	BOOL	Shows that a warning is pending
WarningAdditional1	DWORD	Shows Additional 1.
WarningAdditional2	DWORD	Shows Additional 2.

Tab. 3-4: MB_REGI_WARNING data structure

Enumeration type

The enumeration type **MB_REGI_SENSOR01_COMM_STATUS** can assume the following values:

Element	Value	Description
WAIT_FOR_NEXT_ACTION	0	Function block waits for the subsequent action
CONFIGURATION_IN_WORK	1	Configuration is transmitted
BARCODE_SEARCH_IN_WORK	2	Bar code search is running
MARKSTREAM_SEARCH_IN_WORK	3	Mark stream search is running
TEACH_IN_IN_WORK	4	TeachIn is running
EASY_TEACH_IN_IN_WORK	5	EasyTeachIn in running
READ_DIAG_IN_WORK	6	ReadDiagData is running
READ_TEACH_DATA_IN_WORK	7	ReadTeachData is running
WRITE_TEACH_DATA_IN_WORK	8	WriteTeachData is running
MEASUREMENT_STARTING	9	Measurement is started
MEASUREMENT_STOPPING	10	Measurement is stopped
SEND_SPEED_STARTING	11	SendSpeed is started
SEND_SPEED_STOPPING	12	SendSpeed is stopped
SENSOR_OFF	13	The function block is not in state InOperation.
SENSOR_ERROR	14	Sensor in status Error

Tab. 3-5: Enumeration type MB_REGI_SENSOR01_COMM_STATUS

The enumeration type **MB_REGI_SENSOR01_COMM_ACTIONS** can assume the following values:

Element	Value	Description
SET_CONFIG	1	Start configuration transmission
SEARCH_BARCODE	2	Start bar code search
SEARCH_MARK_STREAM	3	Start mark stream search
TEACH_IN	4	Start TeachIn
EASY_TEACH_IN	5	Start Easy TeachIn
READ_DIAG	6	Read diagnostic data
READ_TEACHING_DATA	7	Read Teach data
WRITE_TEACHING_DATA	8	Write Teach data
START_MEASURING	9	Start measurement

Element	Value	Description
STOP_MEASURING	10	Stop measurement
START_SEND_SPEED	11	Start sending velocity information
STOP_SEND_SPEED	12	Stop sending velocity information

Tab. 3-6: Enumeration type MB_REGI_SENSOR01_COMM_ACTIONS

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
StartAction	BOOL			FALSE	Continuous
SelectAction	MB_REGI_SENSOR01_COMM_ACTIONS	1	12	1 (SET_CONFIG)	In case of rising edge at StartAction and status = WAIT_FOR_NEXT_ACTION
TimeSendInterval	TIME	$\geq 2 \cdot \text{CycleTime}$	n. def	t#32ms	Action START_SEND_SPEED is started
CycleTime	TIME	> 0	n. def	t#8ms	Action START_SEND_SPEED is started
Format	POINTER TO MB_REGI_FORMAT	> 0	n. def	0	Rising edge at "Enable"
MarkStructure	POINTER TO MB_REGI_MARK_STRUCTURE	> 0	n. def	0	Rising edge at "Enable"
Bar code	POINTER TO MB_REGI_BARCODE	> 0	n. def	0	Rising edge at "Enable"
MarkStream	POINTER TO MB_REGI_MARK_STREAM	> 0	n. def	0	Rising edge at "Enable"
LengthRegister-Mode	POINTER TO MB_REGI_LENGTH_REGISTER_MODE	> 0	n. def	0	Rising edge at "Enable"
SideRegister-Mode	POINTER TO MB_REGI_SIDE_REGISTER_MODE	> 0	n. def	0	Rising edge at "Enable"
Communication-Config	POINTER TO MB_REGI_COMMUNICATION_CONFIG	> 0	n. def	0	Rising edge at "Enable"
Tolerances	POINTER TO MB_REGI_TOLERANCES	> 0	n. def	0	Rising edge at "Enable"
InternalParameter	POINTER TO MB_REGI_INTERNAL_PARAMETER	> 0	n. def	0	Rising edge at "Enable"

Tab. 3-7: Overview on min./max. values and initial values of the inputs

In case of a rising edge at enable, it is checked if all pointers > 0 and thus, if the addresses containing the data structures are created. The data of the created pointers are checked when calling the individual function blocks. The data structures have the following default values and are applied or checked during the following actions:

RMB_TechRegiExtended

Name	Type	Min. value	Max. value	Default value	Effective
MB_REGI_FORMAT					
FormatLength	REAL	> 0	n. def	0	Rising edge at "Enable"
Increments	UDINT	> 0	n. def	0	Rising edge at "Enable"
MB_REGI_MARK_STRUCTURE					
MarkType	UDINT	1	7	1 (only the marks 1, 3 and 5 are implemented)	Action SET_CONFIG is started
SingleMark	ARRAY [0...3] OF REAL	> 0 (for required entries)	n. def	0	Action SET_CONFIG is started
MB_REGI_BARCODE					
NumberOfBlocks	UDINT	0 (1 and 2 blocks are invalid entries)	10	0	Action SET_CONFIG is started
BeginExpectationWindow	REAL	0	n. def	0	Action SET_CONFIG is started
SizeOfExpectationWindow	REAL	0	n. def	1	Action SET_CONFIG is started
FreeSpaceBeforeBarcode	REAL	0	n. def	0	Action SET_CONFIG is started
BarcodeBlocks	ARRAY [0...19] OF REAL	> 0 (for required entries)	n. def	0	Action SET_CONFIG is started
MB_REGI_MARK_STREAM					
NumberOfMarks	UDINT	1	12	1	Action SET_CONFIG is started
MarkStream	Array [1...12] OF REAL	> 0 (for required entries)	n. def	0	Action SET_CONFIG is started
MarkDirection	BOOL			FALSE	Action SET_CONFIG is started
MB_REGI_LENGTH_REGISTER_MODE					
ModeSelect	UDINT	1	1	1	Action SET_CONFIG is started
LengthProperties	ARRAY [0...4] OF REAL	n. def	n. def	0	Action SET_CONFIG is started
MB_REGI_SIDE_REGISTER_MODE					
ModeSelect	UDINT	0	1	1	Action SET_CONFIG is started
SelectMarks	ARRAY [0...1] OF UINT	n. def	n. def	0	Action SET_CONFIG is started
MB_REGI_COMMUNICATION_CONFIG					
IpSensor	STRING	LEN > 0	n. def	""	Rising edge at "Enable"

RMB_TechRegiExtended

Name	Type	Min. value	Max. value	Default value	Effective
PortConfiguration	UINT	1	n. def	0	Rising edge at "Enable"
IpPLC	STRING	LEN > 0	n. def	""	Rising edge at "Enable"
PortMeasurement	UINT	1	n. def	0	Rising edge at "Enable"
MB_REGI_TOLERANCES					
TeachZeroPulseToMark	REAL	1	n. def	5	Action SET_CONFIG is started
TeachMarkToMark	REAL	1	n. def	2	Action SET_CONFIG is started
MeasureMarkTolerance	REAL	1	n. def	10	Action SET_CONFIG is started
DeviationOfMarkWidth	REAL	1	100	20	Action SET_CONFIG is started
DeviationOfMarkHeight	REAL	1	100	100	Action SET_CONFIG is started
DeviationOfLastPosition	REAL	1	n. def	10	Action SET_CONFIG is started
ShiftSyncPulse	REAL	1	n. def	5	Action SET_CONFIG is started
MB_REGI_INTERNAL_PARAMETER					
ColorWeighting	REAL	0	< 1	0.9999	Action SET_CONFIG is started
IntensityWeighting	REAL	0	< 1	0.15	Action SET_CONFIG is started
Gain	UINT	0	7	0	Action SET_CONFIG is started

Tab. 3-8: Overview of min./max. and default values of the referenced structures

The VAR_IN_OUTs RegiSensorData and DiagData do not have any min./max. and default values, as only internal function blocks and not the user is allowed to write on them. The corresponding data are stored in TeachData when reading the tech data. The created teach data, e.g. originating from a previously saved production, are written on the sensor when writing the teach data. This data structure cannot be interpreted.

Function description

For the function description, it is also recommended to consider the individual descriptions of the following function blocks:

- [chapter 3.3.3 "MB_RegiSensor01ConfigType01" on page 39](#)
- [chapter 3.3.6 "MB_RegiSensor01CycComType02" on page 63](#)
- [chapter 3.3.11 "MB_RegiSensor01ReadDiagType01" on page 110](#)
- [chapter 3.3.4 "MB_RegiSensor01SendRecvDataType01" on page 56](#)
- [chapter 3.3.12 "MB_RegiSensor01SendSpeedAccType01" on page 112](#)

RMB_TechRegiExtended

- [chapter 3.3.13 "MB_RegiSensor01TeachDataType01" on page 117](#)

Initialization

In case of a rising edge at Enable, it is checked for created pointers. The pointer contents of the respective data structure are directly transferred at the respective function blocks. Only after this is done, it is checked when the corresponding function block is called. Subsequently, the MB_RegiSensor01SendRecvDataType function block is started, executing the communication with the mark stream sensor. It is checked if the mark stream sensor can be contacted under the specified IP address using the "Ping" command. Subsequently, the function block configuration is transmitted using the MB_RegiSensor01ConfigType01 function block and the MB_RegiSensor01CycComType02 function block is started to receive the measured values. The initialization phase is completed and InOperation becomes TRUE. Each error that occurred in the initialization phase immediately results in a transition to the error status and Error = TRUE. In cyclic operation, it is differentiated between errors and warnings. The sensor is in state SENSOR_OFF during the initialization.

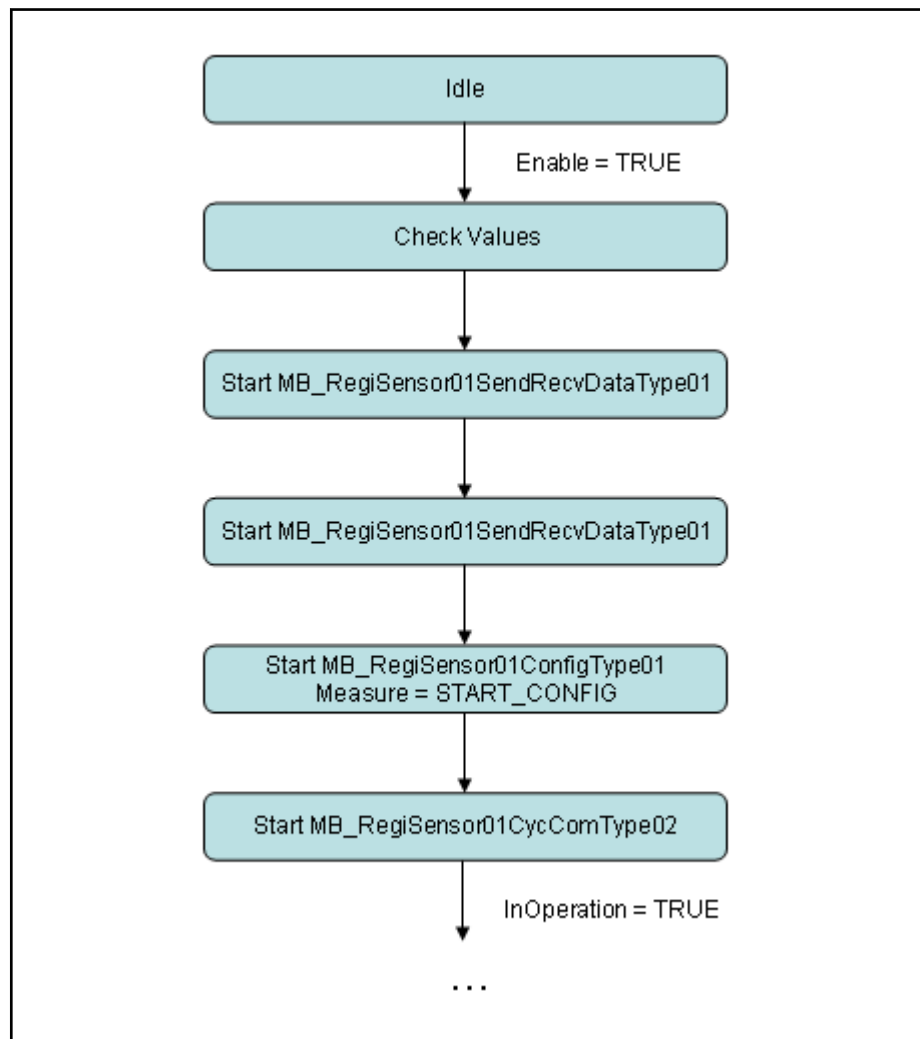


Fig. 3-10: Initialization phase sequence



During the initialization, it is checked if a measurement is still running on the mark stream sensor. If this is the case, the measurement is automatically stopped.

Cyclic operation

The following states can be reached in cyclic operation:

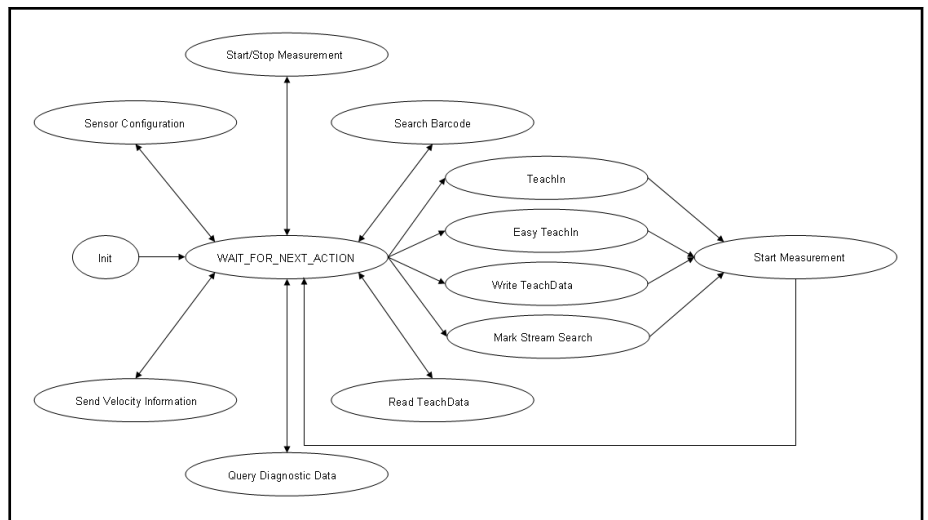


Fig. 3-11: State transition in cyclic operation

The function block is in status "WAIT_FOR_NEXT_ACTION" in cyclic operation (i.e. InOperation = TRUE). In this state, all actions defined in the "MB_REGI_SENSOR01_COMM_ACTIONS" enumeration type can be executed. The corresponding action has to be selected in SelectAction. Subsequently, the action has to be started with a rising edge at StartAction. While an action is running, the corresponding status of the function block can be read via status. ActionDone indicates a successfully completed action. No new action can be started if an action is already running. Three events can cause the action to be closed. ActionDone indicates a successfully completed action. If a severe communication error occurred, the function block is in error status (Error = TRUE). If an action was not completed successfully (e.g., not all marks have been found during TeachIn), the function block reports a warning (ActionWarning = TRUE) and returns to status "WAIT_FOR_NEXT_ACTION". The warning is automatically reset upon starting a new action.

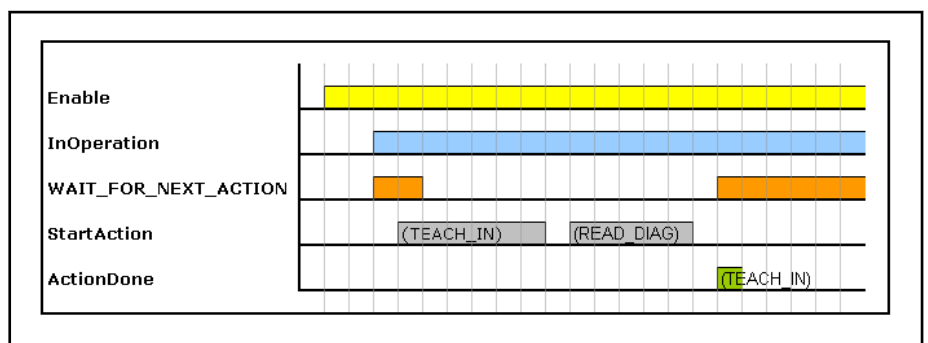


Fig. 3-12: Time diagram in cyclic operation

The time diagram shows that InOperation is set after a rising edge at Enable and after a couple of cycles. It is changed to status "WAIT_FOR_NEXT_ACTION". A rising edge at StartAction with SelectAction := TEACH_IN results in leaving the status "WAIT_FOR_NEXT_ACTION". Teaching is executed. However, a new rising edge at StartAction with SelectAction := READ_DIAG is ignored. At this point, the function block is not in status "WAIT_FOR_NEXT_ACTION", as the teaching action is already being executed. After teaching has been completed successfully, ActionDone = TRUE is set for one cycle (as the rising edge at StartAction has already been removed). The function block returns to the "WAIT_FOR_NEXT_ACTION" state.

RMB_TechRegiExtended

An executed action is completed successfully if the corresponding acknowledgement ActionDone = TRUE is reported. It is not possible to execute several operations simultaneously. If the rising edge has already been removed from the input, ActionDone is pending for exactly 1 cycle (see the following time diagram).

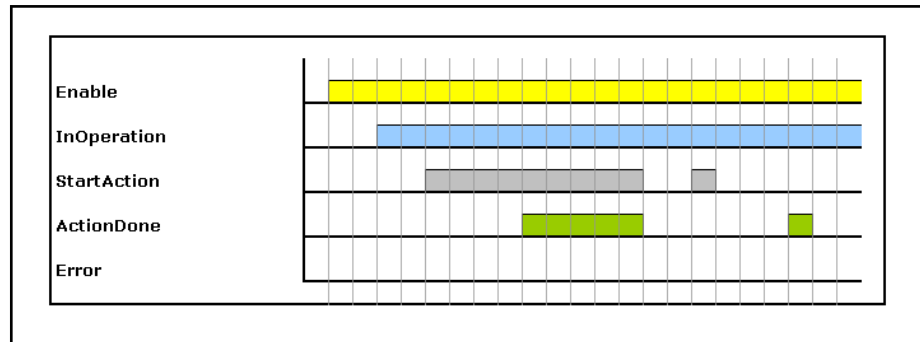


Fig. 3-13: Time diagram of ActionDone behavior

As several actions take some time (e.g. TeachIn), the function block has the output "Status". If Status = 0 and InOperation = TRUE, the function block is in status "WAIT_FOR_NEXT_ACTION". The next action can be executed. The status can assume the values of the "MB_REGI_SENSOR01_COMM_STATUS" enumeration type. If the function block is not InOperation = TRUE, the status is "SENSOR_OFF", except for if the status is in error status. If this is the case, the status is "SENSOR_ERROR".

After starting or stopping the measurement or after starting or stopping the sending of velocity information, the current status is displayed at the MeasureInOp and SendSpeedInOp outputs. Thus, it is continuously displayed that the measurement and the transmission of the velocity information is running.

Before executing the following actions, it is checked if a measuring is running:

- SET_CONFIG
- SEARCH_BARCODE
- SEARCH_MARK_STREAM
- TEACH_IN
- EASY_TEACH_IN

If this is the case, the measurement is stopped and the action is subsequently executed.

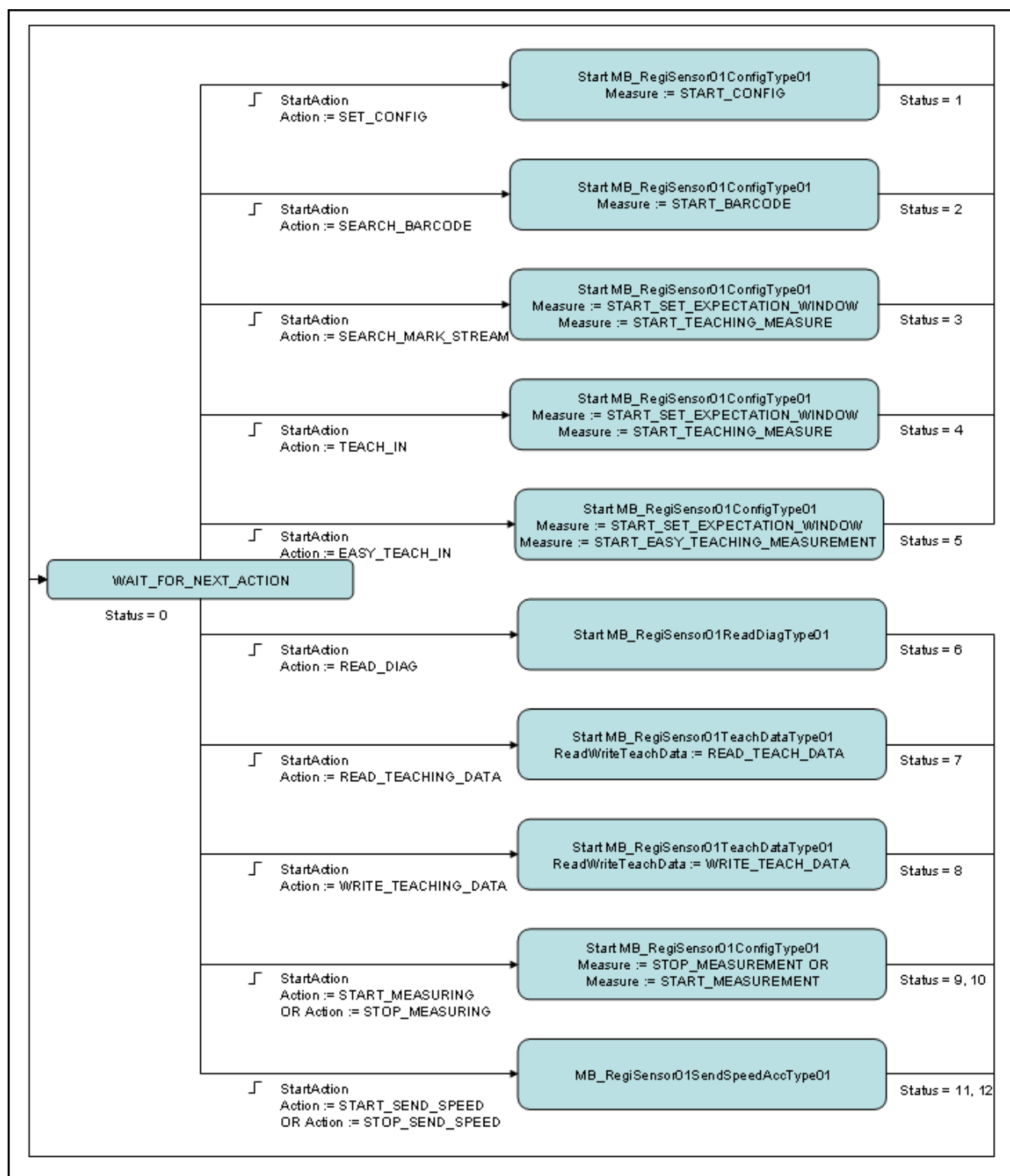


Fig. 3-14: Cyclic operation sequence

The figure shows the sketched sequence of the cyclic operation. It is shown, that it can be jumped to the respective actions starting from the "WAIT_FOR_NEXT_ACTION" status. It is returned to the initial state after the action has been completed. It is also possible to jump to the error status (not shown). Basic inputs for these actions have been marked. Several function

RMB_TechRegiExtended

blocks can be called in the states. For example, in case of a rising edge at StartAction (SelectAction := TEACH_IN), the MB_RegiSensor01Config-Type01 function block is initially called to set the expectation window and to subsequently call it again for teaching and starting the measurement.

SEARCH_MARK_STREAM

This function searches for the mark stream without bar code search and without expectation window specification.

The following preconditions have to be fulfilled for this function:

- Only register marks can be found on the 360° rotation that is sampled by the mark stream sensor. This is to say that neither register crosses nor other markings should be on the sampling track of the sensor.
- The register mark stream should take up a maximum of 2/3 of the printing cylinder format. Thus, at least 1/3 of the printing cylinder format has to be an area in which no printing takes place. Example: If 12 register marks are contained on the format to be measured with a distance of 20 mm (mark stream of size 220 mm), the printing cylinder format has to be at least 330 mm.

If one of the specified preconditions is not met, the function cannot be executed or is closed with an error.



The function searches for the mark stream with a maximum of 3 teach actions. Only a valid configuration needs to have been transferred. The expectation window does not need to be specified.

Data/data structures

As not all inputs are required for each action, the following table shows the data applied during the individual actions:

SelectAction	Data	Data changed by the action
SET_CONFIG	Format MarkStructure Bar code MarkStream LengthRegisterMode SideRegisterMode CommunicationConfig Tolerances InternalParameter	ActionDone Status
SEARCH_BARCODE	Barcode.NumberOfBlocks Barcode.FreeSpaceBeforeBarcode Barcode.BarcodeBlocks InternalParameter.Gain	ActionDone Status BarcodeSearchBeginOfExpWindow
SEARCH_MARK_STREAM	-	ActionDone Status
TEACH_IN	Barcode.BeginExpectationWindow Barcode.SizeOfExpectationWindow	ActionDone Status RegiSensorData

RMB_TechRegiExtended

SelectAction	Data	Data changed by the action
EASY_TEACH_IN	Barcode.BeginExpectationWindow Barcode.SizeOfExpectationWindow InternalParameter.Gain	ActionDone Status RegiSensorData
READ_DIAG	-	ActionDone Status DiagData
READ_TEACHING_DATA	-	ActionDone Status TeachData
WRITE_TEACHING_DATA	TeachData	ActionDone Status
START_MEASURING	-	ActionDone MeasureInOp Status RegiSensorData
STOP_MEASURING	-	ActionDone MeasureInOp Status RegiSensorData
START_SEND_SPEED	TimeSendIntervall CycleTime Master	ActionDone SendSpeedInOp Status
STOP_SEND_SPEED	-	ActionDone SendSpeedInOp Status

Tab. 3-9: Data dependencies

Exemplary sequence

RMB_TechRegiExtended

The exemplary sequence from configuration to measurement is described for fbRegiSensor01Comm instance of the MB_RegiSensor01CommType01 function block in the following figure:

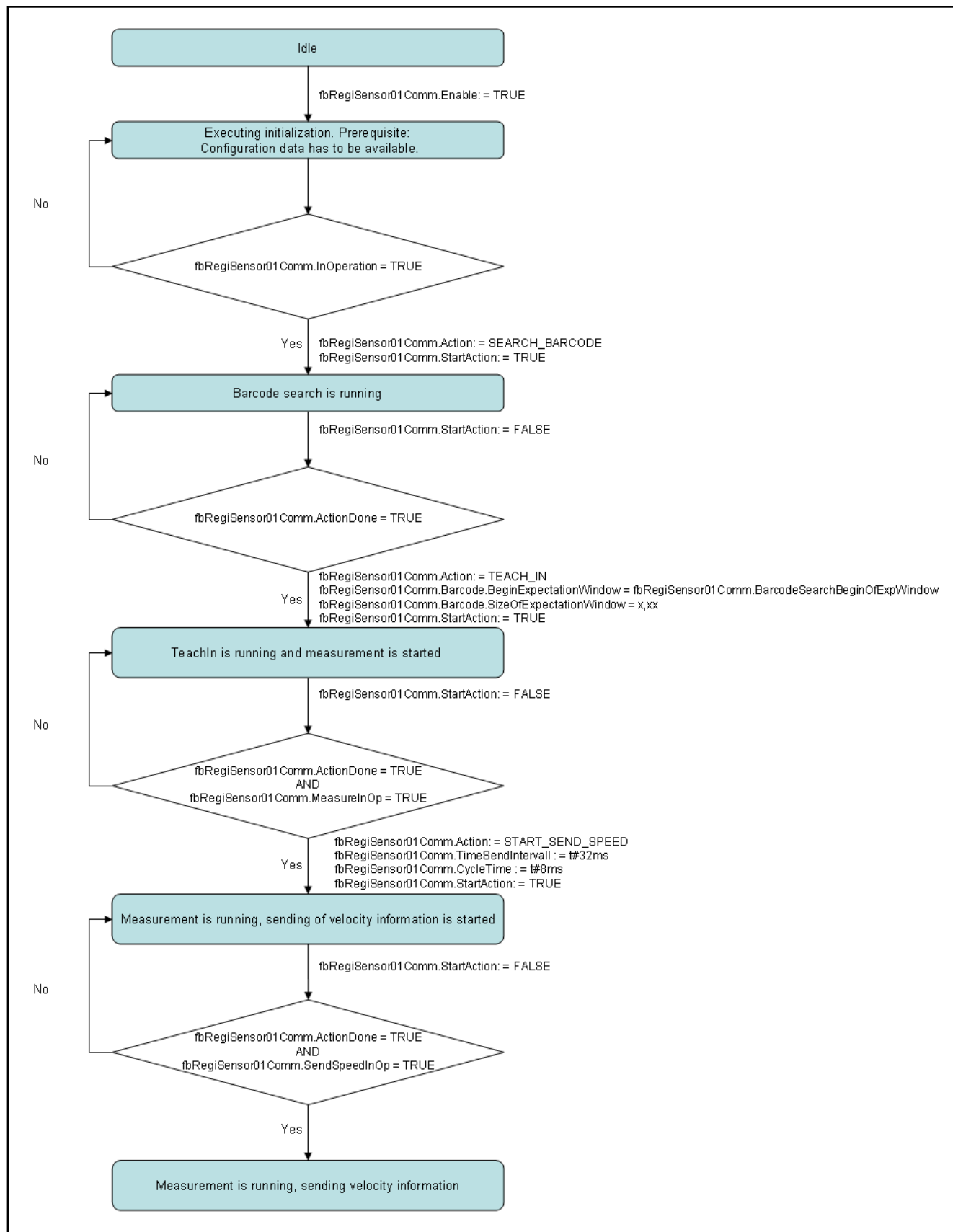


Fig. 3-15: Exemplary sequence from configuration to measurement

Initially, a rising edge is set at enable of the fbRegiSensor01Comm instance. The function block MB_RegiSensor01SendRecvDataType01 establishes the communication connection to the sensor. Subsequently, the configuration is transferred using the MB_RegiSensor01ConfigType01 function block. Pre-condition: valid data have to be used. The MB_RegiSensor01CycComType02 measuring block is started and InOperation becomes "TRUE". Subsequently, the function block is in cyclic mode. The bar code search starts. After the search has been completed, the value has to be assigned as expectation window start and a valid value has to be available for the expectation window size. TeachIn is started. The expectation window is set, the teaching is started to subsequently start the measurement. To achieve a higher accuracy during the acceleration phase, transmitting of the velocity information is enabled using the MB_RegiSensor01SendSpeedAccType01 function block. The measurement is started and the measured data can be found in the RegiSensorData structure.

Characteristics of warning and error

The function block has a second diagnostic interface "MB_REGI_WARNING" apart from the standardized error handling. Warnings can be displayed by means of that interface. The second diagnostic structure is needed due to the connection of several function blocks and the external Rexroth mark stream sensor. Thus, it is differentiated between severe error (Error = TRUE and error status) and error that are reported by an internal function block but which should not reset the entire MB_RegiSensor01CommType01 function block. For example, not all marks can be found during a TeachIn. This is reported as warning as it is no severe error and the user can still execute another action.

A warning is defined as follows:

- Warnings are only generated during cyclic operation.
- A warning is set after an action has not been executed successfully.
- No confirmation by ActionDone is output.
- The function block returns to the "WAIT_FOR_NEXT_ACTION" state.
- The warning is automatically reset once a new action is executed.
- In WarningAdditional1, Additional 1 of the "ERROR_STRUCT" error structure and in WarningAdditional2, Additional 2 of the "ERROR_STRUCT" error structure of the internal function block are pending.

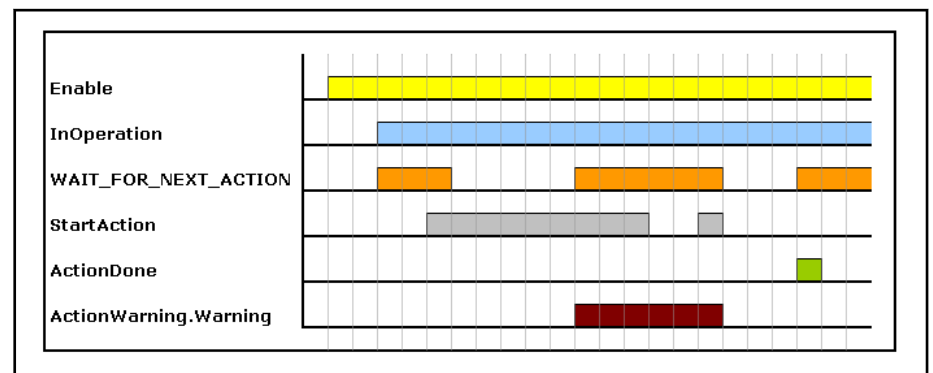


Fig. 3-16: Time response of the warning

The diagram shows the time response when a warning occurs. An action is executed in cyclic operation (InOperation=TRUE). The action is not completed successfully. That is the reason why a warning is reported. The function block returns to the "WAIT_FOR_NEXT_ACTION" state when the warning is reported. The warning is automatically reset after a start of a new action. This action can be completed successfully and is acknowledged with ActionDone.

RMB_TechRegiExtended

Warning and Error during initialization

Each error generated by an internal function block during initialization results in an error of the MB_RegiSensor01CommType01 function block. I.e., a connection has to be re-established to the sensor, a valid configuration has to be transferred and the measuring block has to be started.

Warning and error in cyclic operation

Only connection errors result in an error during cyclic operation as connection errors are severe errors. Severe errors are all errors of the MB_RegiSensor01SendRecvDataType01 and MB_RegiSensor01CycComType02 function blocks.

The following error numbers of other internally used function blocks result in a transition to the error status:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000000	Timeout; no response from the MB_RegiSensor01SendRecvDataType01 function block within 10 seconds
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000005	Timeout while receiving a package.
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000006	Timeout; no response from the MB_RegiSensor01SendRecvDataType01 function block within 10 seconds

Tab. 3-10: Internal function block errors, resulting in error status

All other internal function block errors are interpreted as warnings. If a warning occurs, Additional1 and Additional2 of "ERROR_STRUCT" of the internal function block is copied to WarningAdditional1 and WarningAdditional2 and the warning bit is set. Internally, the function block in which the error occurs is reset. When executing an action again, the warning is automatically deleted and WarningAdditional1 and WarningAdditional2 are reset to 0.

Error handling

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
OTHER_ERROR (16#7FFE)	16#00001D13	16#00000001	Mark stream search failed
OTHER_ERROR (16#7FFE)	16#00001D13	16#00000002	Measurement has already been started
OTHER_ERROR (16#7FFE)	16#00001D13	16#00000003	Measurement has already been stopped
OTHER_ERROR (16#7FFE)	16#00001D13	16#00000004	Send velocity information has already been started
OTHER_ERROR (16#7FFE)	16#00001D13	16#00000005	Send velocity information has already been stopped
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000003C	Zero pointer format
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000003D	Zero pointer MarkStructure
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000003E	Zero pointer bar code
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000003F	Zero pointer MarkStream

ErrorID	Additional 1	Additional 2	Description
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000040	Zero pointer LengthRegisterMode
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000041	Zero pointer SideRegisterMode
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000042	Zero pointer CommunicationConfig
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000043	Zero pointer Tolerances
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000044	Zero pointer InternalParameter

Tab. 3-11: Error codes of the MB_RegiSensor01CommType01 function block
The function block continues to report error messages of the integrated function blocks:

- MB_RegiSensor01ConfigType01 (chapter "Error handling" on page 55)
- MB_RegiSensor01CycComType02 (chapter "Error handling" on page 67)
- MB_RegiSensor01ReadDiagType01 (chapter "Troubleshooting" on page 112)
- MB_RegiSensor01SendRecvDataType01 (chapter "Error handling" on page 59)
- MB_RegiSensor01SendSpeedAccType01 (chapter "Error handling" on page 116)
- MB_RegiSensor01TeachDataType01 (chapter "Error handling" on page 119)

The corresponding error messages are only generated when the respective function block is enabled. The application of the respective values has to be taken into consideration. Error messages can be generated in case of an active InOperation. Furthermore, these errors, as previously described, can be reported as warnings.

3.3.3 MB_RegiSensor01ConfigType01

Brief description

The MB_RegiSensor01ConfigType01 function block is required for the configuration of the mark stream sensor. The function block prepares the configuration data and transmits them to the MB_RegiSensor01SendRecvDataType01 function block that is responsible for sending the package and receiving the response package. The expectation window can be set apart from the configuration, the teach-in and the measurement can be started or the measurement can be stopped. It executes the initial configuration of the sensor and configures the sensor during the measurement.

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V11	RMB_TechRegiExtended

Tab. 3-12: Reference table of the MB_RegiSensor01ConfigType01 function block

RMB_TechRegiExtended

Interface description

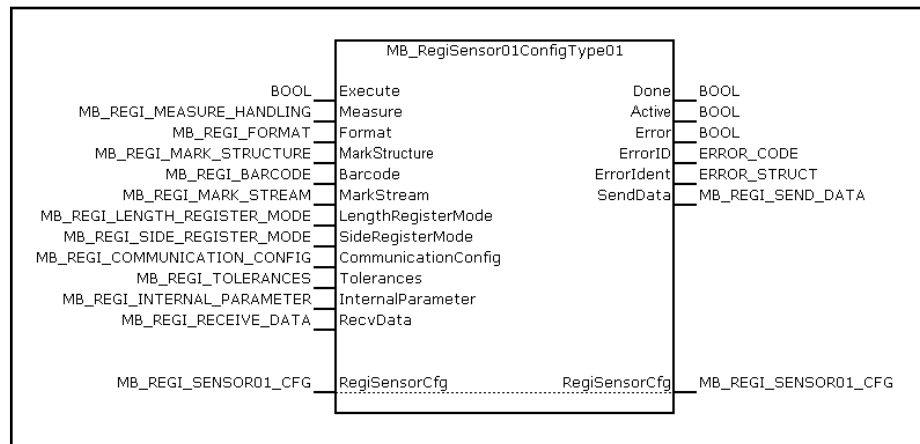


Fig. 3-17: MB_RegiSensor01ConfigType01 function block

I/O type	Name	Data type	Explanation
VAR_IN_OUT	RegiSensorCfg	MB_REGI_SENSOR01_CFG	Contains the current configuration data of the sensor
VAR_INPUT	Execute	BOOL	Processing of the function block enabled (edge-controlled)
	Measure	MB_REGI_MEASURE_HANDLING	Transmits the configuration, sets the expectation window, starts the teach-in and measurement or stops the measurement.
	Format	MB_REGI_FORMAT	Defines the format characteristics
	MarkStructure	MB_REGI_MARK_STRUCTURE	Defines the mark structure
	Bar code	MB_REGI_BARCODE	Defines the bar code
	MarkStream	MB_REGI_MARK_STREAM	Defines the mark stream
	LengthRegister-Mode	MB_REGI_LENGTH_REGISTER_MODE	Defines the web measuring principle for the length register
	SideRegister-Mode	MB_REGI_SIDE_REGISTER_MODE	Defines the measuring principle used for the side register
	Communication-Config	MB_REGI_COMMUNICATION_CONFIG	Defines the communication characteristics of the UDP connection
	Tolerances	MB_REGI_TOLERANCES	Defines the tolerances allowed for teaching and measurement
	InternalParameter	MB_REGI_INTERNAL_PARAMETER	Internal sensor parameters that should only be changed if specifically mentioned
	RecvData	MB_REGI_RECEIVE_DATA	Receives data from the MB_RegiSensor01SendRecvDataType01 function block

I/O type	Name	Data type	Explanation
VAR_OUTPUT	Done	BOOL	Data outputs are valid
	Active	BOOL	Function block is working and currently in an interim state
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Error group
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics
	SendData	MB_REGI_SEND_DATA	Data to be sent via the MB_RegiSensor01SendRecvData function block

Tab. 3-13: I/O interface of the MB_RegiSensor01ConfigType01 function block

Data structures

The following data structures are used by the MB_RegiSensor01ConfigType01 function block:

MB_REGI_SENSOR01_CFG		
CurrentConfig	USINT	Current configuration
Format	MB_REGI_FORMAT	Describes the printing cylinder format
MarkStructure	MB_REGI_MARK_STRUCTURE	Describes the mark structure
Bar code	MB_REGI_BARCODE	Describes the bar code
MarkStream	MB_REGI_MARK_STREAM	Describes the mark stream
LengthRegisterMode	MB_REGI_LENGTH_REGISTER_MODE	Length register mode
SideRegisterMode	MB_REGI_SIDE_REGISTER_MODE	Side register mode
CommunicationData	MB_REGI_COMMUNICATION_CONFIG	Communication ports of the UDP connection
Tolerances	MB_REGI_TOLERANCES	Teaching tolerances and measurement tolerances
InternalParameter	MB_REGI_INTERNAL_PARAMETER	Internal sensor parameters
BarcodeSearchBeginOffsetExpWindow	REAL	Returns the beginning of the expectation window of the bar code search

Tab. 3-14: MB_REGI_SENSOR01_CFG data structure

MB_REGI_FORMAT		
FormatLength	REAL	Printing cylinder format in [mm]
Increments	UDINT	Number of increments per revolution

Tab. 3-15: MB_REGI_FORMAT data structure

MB_REGI_MARK_STRUCTURE		
MarkType	UDINT	Mark type
SingleMark	ARRAY [0...3] OF REAL	Describes the printed mark

Tab. 3-16: MB_REGI_MARK_STRUCTURE data structure

RMB_TechRegiExtended

MB_REGI_BARCODE		
NumberOfBlocks	UDINT	Number of bar code blocks (0 means that no bar code exists)
BeginExpectationWindow	REAL	Beginning of the expectation window (distance from the beginning of the expectation window to the zero pulse) [mm]
SizeOfExpectationWindow	REAL	Size of the expectation window [mm]
FreeSpaceBeforeBarcode	REAL	Printing-free area in front of the bar code [mm]
BarcodeBlocks	ARRAY [0...19] OF REAL	Bar code description in [mm]

Tab. 3-17: MB_REGI_BARCODE data structure

MB_REGI_MARK_STREAM		
NumberOfMarks	UINT	Number of marks
MarkStream	ARRAY [1...12] OF REAL	Distance of the marks [mm]
MarkDirection	BOOL	First or second mark edge for the length register

Tab. 3-18: MB_REGI_MARK_STREAM data structure

MB_REGI_LENGTH_REGISTER_MODE		
ModeSelect	UDINT	Selected length register mode
LengthProperties	ARRAY [0...4] OF REAL	Characteristics for the length register mode [mm]

Tab. 3-19: MB_REGI_LENGTH_REGISTER_MODE data structure

MB_REGI_SIDE_REGISTER_MODE		
ModeSelect	UDINT	Selected side register mode
MarkSelect	ARRAY [0...1] OF UINT	Marks for the side register

Tab. 3-20: MB_REGI_SIDE_REGISTER_MODE data structure

MB_REGI_COMMUNICATION_CONFIG		
IpSensor	STRING	IP address of the sensor
PortConfiguration	UINT	Port for acyclic communication (configuration, diagnostics, velocity data)
IpPLC	STRING	IP address of the PLC
PortMeasurement	UINT	Port for the measurement results

Tab. 3-21: MB_REGI_COMMUNICATION_CONFIG data structure

MB_REGI_TOLERANCES		
TeachZeroPulseToMark	REAL	Teaching tolerance of the first mark to the zero pulse [mm]
TeachMarkToMark	REAL	Teaching tolerance between the individual marks [mm]
MeasureMarkTolerance	REAL	Measurement tolerance compared to the next measured value [mm]
DeviationOfMarkWidth	REAL	Deviation of the mark width to the last measured value [%]
DeviationOfMarkHight	REAL	Deviation of the mark height to the last measured value [%]
DeviationOfLastPosition	REAL	Absolute deviation of the measurement tolerance from the last measured value [mm]
ShiftSyncPulse	REAL	The zero pulse is exactly shifted in front of the expectation window by ShiftSync-Pulse [mm]

Tab. 3-22: MB_REGI_TOLERANCES data structure

MB_REGI_INTERNAL_PARAMETER		
ColorWeighting	REAL	Color weighting of the sensor
IntensityWeighting	REAL	Intensity weighting of the sensor
Gain	UINT	Sensor gain

Tab. 3-23: MB_REGI_INTERNAL_PARAMETER data structure

MB_REGI_SEND_DATA		
NoOfBytes	UDINT	Number of bytes to be sent
ValueAddr	POINTER TO DWORD	Pointer to data
NewData	UINT	Indicates new data for sending
SensorPort	UINT	Sensor port to which the data should be sent

Tab. 3-24: MB_REGI_SEND_DATA data structure

MB_REGI_RECEIVE_DATA		
NoOfBytes	UDINT	Number of received bytes
ValueAddr	POINTER TO DWORD	Pointer to data
NewData	UINT	Indicates newly received data

Tab. 3-25: MB_REGI_RECEIVE_DATA data structure

Enumeration type

The enumeration type MB_REGI_MEASURE_HANDLING is used for configuration, measurement start, teach-in start as well as for setting the expectation window and can assume the following values:

Element	Value	Description
START_CONFIG	0	Starts the configuration
START_TEACHING	1	Starts the teaching
START_MEASUREMENT	2	Starts the measurement
STOP_MEASUREMENT	3	Stops the measurement
START_CONFIG_TEACHING	4	Starts the configuration and the teaching

RMB_TechRegiExtended

Element	Value	Description
START_TEACHING_MEASUREMENT	5	Starts the teaching and the measurement
START_CONFIG_TEACHING_MEASUREMENT	6	Starts the configuration, the teaching and the measurement
START_SET_EXPECTATION_WINDOW	7	Sets the expectation window
START_EASY_TEACHING	8	Starts an easy teach-in. The gain as well as the beginning and the size of the expectation window have to be transferred.
START_EASY_TEACHING_MEASUREMENT	9	Starts the each teach-in and then the measurement (cf. 8)
START_BARCODE	10	Starts the bar code search

Tab. 3-26: Enumeration type MB_REGI_MEASURE_HANDLING

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
MB_REGI_SENSOR01_CFG					
CurrentConfig	USINT	0	255	0	Rising edge at "Execute"
MB_REGI_FORMAT					
FormatLength	REAL	> 0	n. def	0	Rising edge at "Execute"
Increments	UDINT	> 0	n. def	0	Rising edge at "Execute"
MB_REGI_MARK_STRUCTURE					
MarkType	UDINT	1	7	1 (only the marks 1, 3 and 5 are implemented)	Rising edge at "Execute"
SingleMark	ARRAY [0...3] OF REAL	> 0 (for required entries)	n. def	0	Rising edge at "Execute"
MB_REGI_BARCODE					
NumberOfBlocks	UDINT	0 (1 and 2 blocks are invalid entries)	10	0	Rising edge at "Execute"
BeginExpectationWindow	REAL	0	n. def	0	Rising edge at "Execute"
SizeOfExpectationWindow	REAL	0	n. def	1	Rising edge at "Execute"
FreeSpaceBeforeBarcode	REAL	0	n. def	0	Rising edge at "Execute"
BarcodeBlocks	ARRAY [0...19] OF REAL	> 0 (for required entries)	n. def		Rising edge at "Execute"
MB_REGI_MARK_STREAM					
NumberOfMarks	UDINT	1	12	1	Rising edge at "Execute"
MarkStream	Array [1...12] OF REAL	> 0 (for required entries)	n. def	0	Rising edge at "Execute"

RMB_TechRegiExtended

Name	Type	Min. value	Max. value	Default value	Effective
MarkDirection	BOOL			FALSE	Rising edge at "Execute"
MB_REGI_LENGTH_REGISTER_MODE					
ModeSelect	UDINT	1	1	1	Rising edge at "Execute"
LengthProperties	ARRAY [0...4] OF REAL	n. def	n. def	0	Rising edge at "Execute"
MB_REGI_SIDE_REGISTER_MODE					
ModeSelect	UDINT	0	1	1	Rising edge at "Execute"
SelectMarks	ARRAY [0...1] OF UINT	n. def	n. def	0	Rising edge at "Execute"
MB_REGI_COMMUNICATION_CONFIG					
IpSensor	STRING	LEN > 0	n. def	""	Rising edge at "Execute"
PortConfiguration	UINT	1	n. def	0	Rising edge at "Execute"
IpPLC	STRING	LEN > 0	n. def	""	Rising edge at "Execute"
PortMeasurement	UINT	1	n. def	0	Rising edge at "Execute"
MB_REGI_TOLERANCES					
TeachZeroPulseToMark	REAL	1	n. def	5	Rising edge at "Execute"
TeachMarkToMark	REAL	1	n. def	2	Rising edge at "Execute"
MeasureMarkTolerance	REAL	1	n. def	10	Rising edge at "Execute"
DeviationOfMarkWidth	REAL	1	100	20	Rising edge at "Execute"
DeviationOfMarkHeight	REAL	1	100	20	Rising edge at "Execute"
DeviationOfLastPosition	REAL	1	n. def	10	Rising edge at "Execute"
ShiftSyncPulse	REAL	1	n. def	5	Rising edge at "Execute"
MB_REGI_INTERNAL_PARAMETER					
ColorWeighting	REAL	0	< 1	0.9999	Rising edge at "Execute"
IntensityWeighting	REAL	0	< 1	0.15	Rising edge at "Execute"
Gain	UINT	0	7	0	Rising edge at "Execute"
MB_REGI_RECEIVE_DATA					
NoOfBytes	UDINT	0	n. def	0	Change at "NewData"
ValueAddr	POINTER TO BYTE	n. def	n. def	0	Change at "NewData"
NewData	UINT	0	65535	0	Continuous

Tab. 3-27: Overview on min./max. values and initial values of the inputs

RMB_TechRegiExtended



Valid entries are currently only functionalities implemented in the sensor.

Function description

In case of the acyclic communication, the sensor is the client which means the control sends data to the sensor and the sensor responds. The acyclic communication is used to configure the sensor.

The RegiSensorCfg structure contains the current configuration of the sensor. If configuration changes should be made, the changed configuration data can be directly applied at the function block. They are only applied at a rising edge at "Execute". The individual structures are described in the following:

RegiSensorCfg: It includes the current configuration data of the sensor and the current configuration ID transmitted to the cyclic measured data. Additionally, the determined beginning of the expectation window is specified after a bar code search.

Measure: *At a rising edge at "Execute" it is distinguished between 11 processes executed according to the function selected:*

- **START_CONFIG:**

The new configuration is transmitted to the sensor. The expectation window is calculated via mark distances, tolerances and the beginning of the expectation window by the sensor.



Therefore, exact entries are required. It is recommended to determine the beginning and the size of the expectation window via a graphical user interface (HMI) and to transfer these values with the START_SET_EXPECTATION_WINDOW option or to use the bar code search.

- **START_TEACHING:**

Starts the sensor teaching. Therefore, the configuration data sent last to the sensor and the expectation window are applied and the marks are determined with their threshold values in the expectation window. In this mode, the sensor additionally determines the ideal gain.

- **START_MEASUREMENT:**

Starts a measurement without transmitting a configuration or starting a teach-in.

- **STOP_MEASUREMENT:**

Completes a running measurement.

- **START_CONFIG_TEACHING:**

First, it transfers the configuration and subsequently starts a teach-in. If a number higher than 0 was specified in SizeOfExpectationWindow, this value is used for the calculation of the expectation window. If the value is 0, the expectation window is computed via mark distances, the tolerances and the beginning of the expectation window by the sensor.

- **START_TEACHING_MEASUREMENT:**

Starts the sensor teaching and subsequently the measurement. The configuration data transmitted last to the sensor and the expectation window are used for teaching.

- **START_CONFIG_TEACHING_MEASUREMENT:**

First, it transfers the configuration and subsequently starts a teach-in and then the measurement. If a number higher than 0 was specified in `SizeOfExpectationWindow`, this value is used for the calculation of the expectation window. If the value is 0, the expectation window is computed via mark distances, the tolerances and the beginning of the expectation window by the sensor.

- **START_SET_EXPECTATION_WINDOW:**

Sets only the expectation window that was for example transferred from a graphical interface. Therefore, a value higher than 0 has to be specified in `SizeOfExpectationWindow` as well as the beginning of the expectation window (`BeginExpectationWindow`).



If a new configuration is started after transferring the expectation window, the transferred expectation window is overwritten. Thus, it is recommended to transfer the configuration first and then the expectation window.

- **START_EASY_TEACHING:**

A teach-in is started without determining the gain. The gain located under "InternalParameters" in "Gain" is applied. This gain was ideally read out with the function block `MB_RegiSensor01ReadDiagType01` after a first normal teach-in. Furthermore, the size and the beginning of the expectation window have to be specified.

- **START_EASY_TEACHING_MEASUREMENT:**

As `START_EASY_TEACHING`, but the measurement is immediately started after teaching.

- **START_BARCODE:**

If this mode is executed, the bar code search is started. Therefore, 3 to 10 bar codes have to be defined. After the bar code search has been completed successfully, the position is right behind the bar code that is the beginning of the expectation window in `RegiSensorCfg.BarcodeSearchBeginOfExpWindow`. This value can be used to define the expectation window.

The following figure shows an exemplary sequence from the configuration of the mark stream sensor to the measurement.

RMB_TechRegiExtended

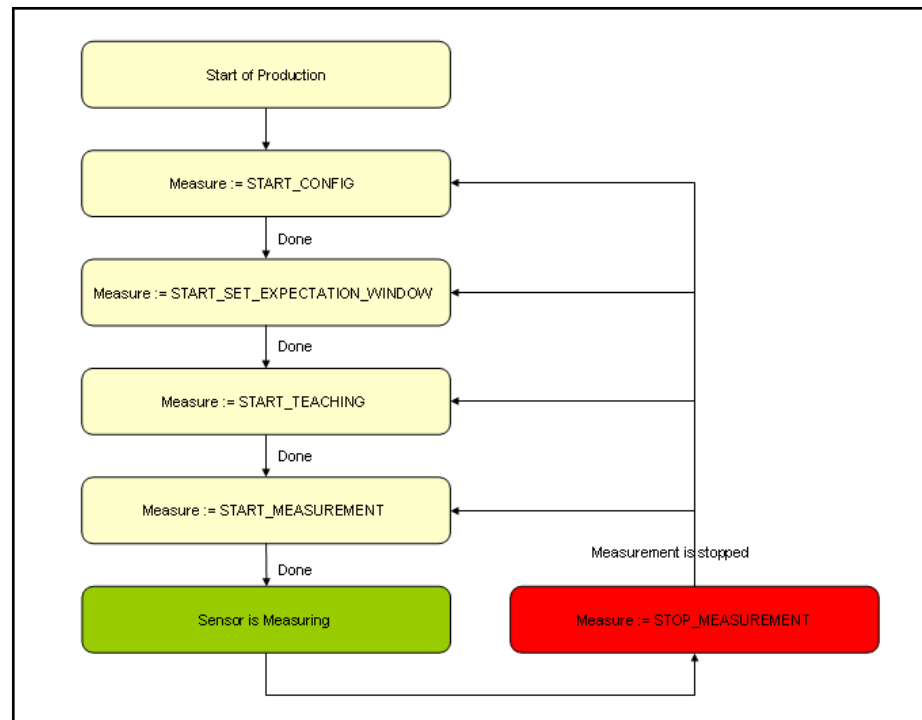


Fig. 3-18: Sequence from the configuration to the measurement

The sensor has to be configured first. Subsequently, the expectation window should be preferably set via a visualization or the thus determined values. Subsequently, teaching and measurement can be started. After the measurement has been started, it can be stopped any time and continued at the sensor state desired. The "Measure" functionalities can be called in any combination.

MarkStructure: Contains the predefined mark structure and the dimension of the marks.

The following mark types are supported:

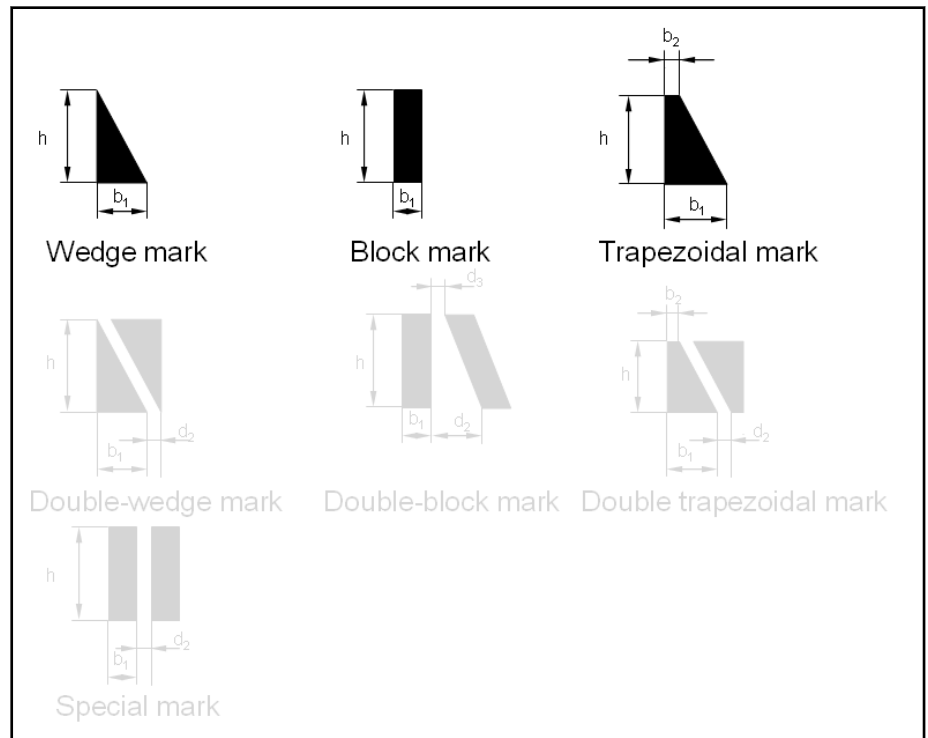


Fig. 3-19: Supported mark types

The "MarkType" parameter can be set as follows and the parameters have to be subsequently specified in "SingleMark" according to the mark type:

	MarkType	SingleMark			
Wedge mark	1	Width [mm]	Height [mm]		
Double-wedge marks	2	Width [mm]	Height [mm]	Distance [mm]	
Block marks	3	Width [mm]	Height [mm]		
Double-block marks	4	Width [mm]	Height [mm]	Distance [mm]	Distance2[mm]
Trapezoidal mark	5	Width [mm]	Height [mm]	Width2 [mm]	
Double-trapezoidal marks	6	Width [mm]	Height [mm]	Distance [mm]	Width2 [mm]
Special mark	7	Width [mm]	Height [mm]	Distance [mm]	

Tab. 3-28: Mark type parameters

The following shows a configuration example of a triangular mark:

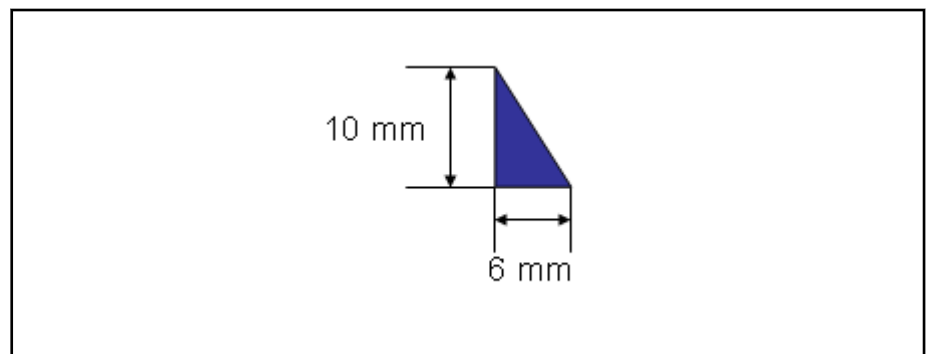


Fig. 3-20: Mark structure example

RMB_TechRegiExtended

Code example:

```
MarkStruct: MB_REGI_MARK_STRUCTURE := (MarkType:=1, SingleMark:= 6, 10, 0, 0);
```

This is a wedge mark (mark type 1). This is defined by its height and width.



Only the mark types 1, 3 and 5 are allowed up to now in the implementation.

Bar code: Contains the number of bar code blocks and the description of the expectation window. Up to ten bar code blocks are possible. If NumberOfBlocks = 0, there is no bar code available. The user has to set the beginning and the size of the expectation window. This can be carried out either by determining this data via the oscilloscope representation or by calculating the size of the expectation window by the sensor (from the number of marks and their distance as well as the tolerances and the beginning of the expectation window). Therefore, Barcode.SizeOfExpectationWindow has to be set to 0.

If a bar code is available, the sensor can determine the beginning of the expectation window by returning the position of the bar code end. Therefore, START_BARCODE has to be started. If the function block returns "Done", the position of the bar code end is written to RegiSensorCfg.BarcodeSearchBeginOfExpWindow. The user has to specify the size of the expectation window and can then set the expectation window with START_SET_EXPECTATION_WINDOW. The normal procedure with "teach-in" and "start measurement" can now be continued.

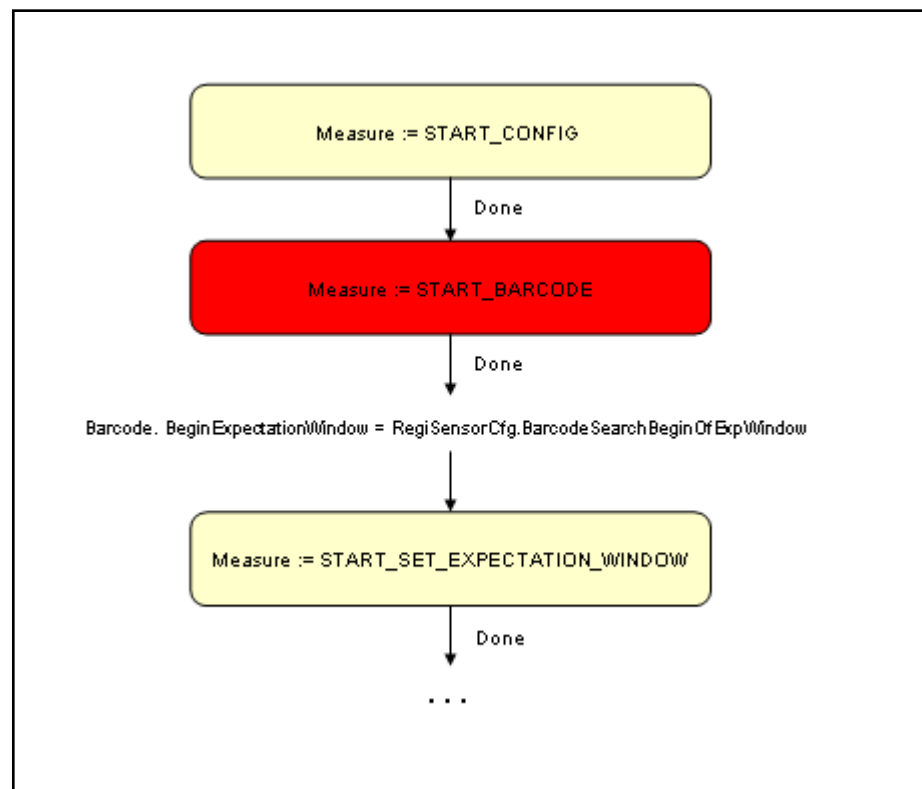


Fig. 3-21: Sequence of the bar code search

If a bar code is available, the sensor searches for the beginning of the expectation window. "FreeSpaceBeforeBarcode" indicates the printing-free area in front of the bar code and "BarcodeBlocks" defines the individual bar code blocks. The dimensions of the bar code still have to be specified as follows:

Bar code example:

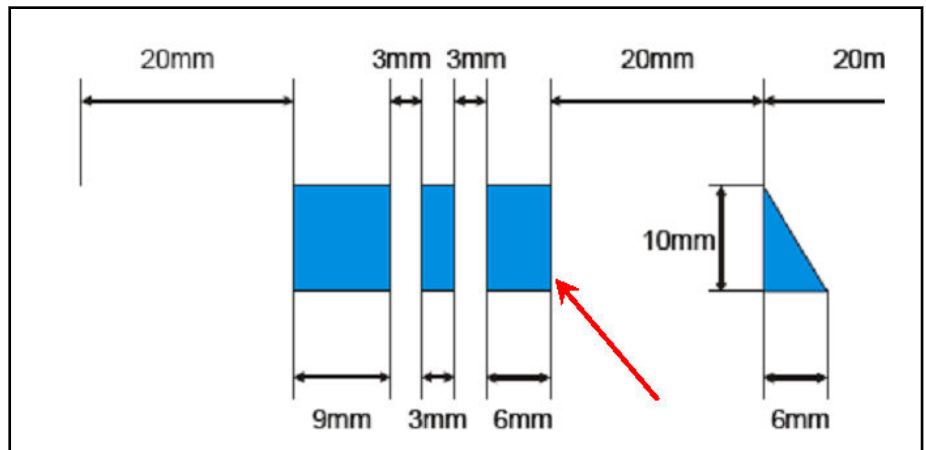


Fig. 3-22: Bar code example

The dimensions are given in the array "BarcodeBlocks" in [mm]:

BarcodeBlocks[0]: Width of bar code block 1 in [mm]

BarcodeBlocks[1]: Distance from bar code block 1 to block 2 in [mm]

BarcodeBlocks[2]: Width of bar code block 2 in [mm]

BarcodeBlocks[3]: Distance from bar code block 2 to block 3 in [mm]

.....

The last distance specified is the distance between the last bar code block and the first mark (in this example, the distance between the bar code block and the registration mark is 20 mm). The position is indicated with the red arrow that is returned by the sensor in RegiSensorCfg.BarcodeSearchBeginOfExpWindow after a successful bar code search. That is the direct position behind the bar code.

MarkStream: Contains the number of marks and the respective distance between the marks. If five marks are available, there are four mark distances and the first defined distance is the distance between mark 1 and mark 2.

Mark stream example:

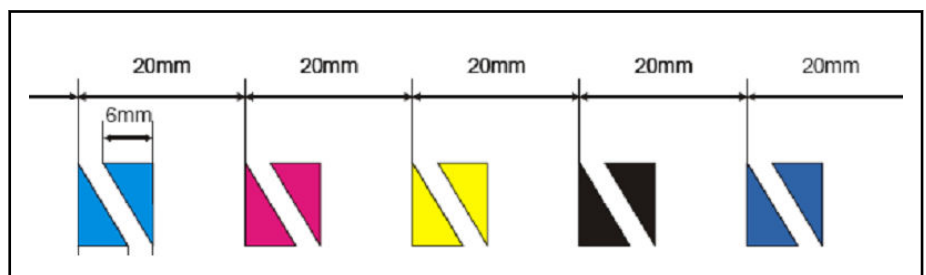


Fig. 3-23: Mark stream example

Example of the shown mark stream:

```
MarkStream: MB_REGI_MARK_STREAM := (NumberOfMarks:=5,
MarkStream:=20,20,20,20);
```

The mark stream consists of five marks. These have a distance of 20 mm each.

Furthermore, it can be specified - when defining the mark stream - whether the vertical mark edge is at first or second place.

RMB_TechRegiExtended

If MarkDirection = FALSE, the length register value is returned with regard to respective first mark edge. That is the distance between the zero pulse and the first mark edge of the mark.

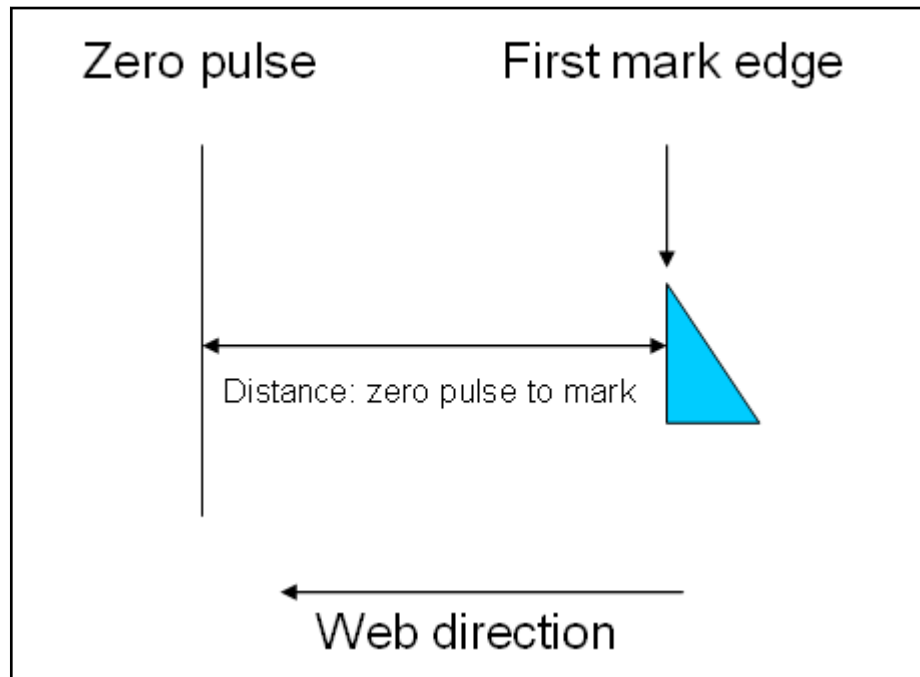


Fig. 3-24: First mark edge

If MarkDirection = TRUE, the length register value is returned with regard to respective second mark edge.

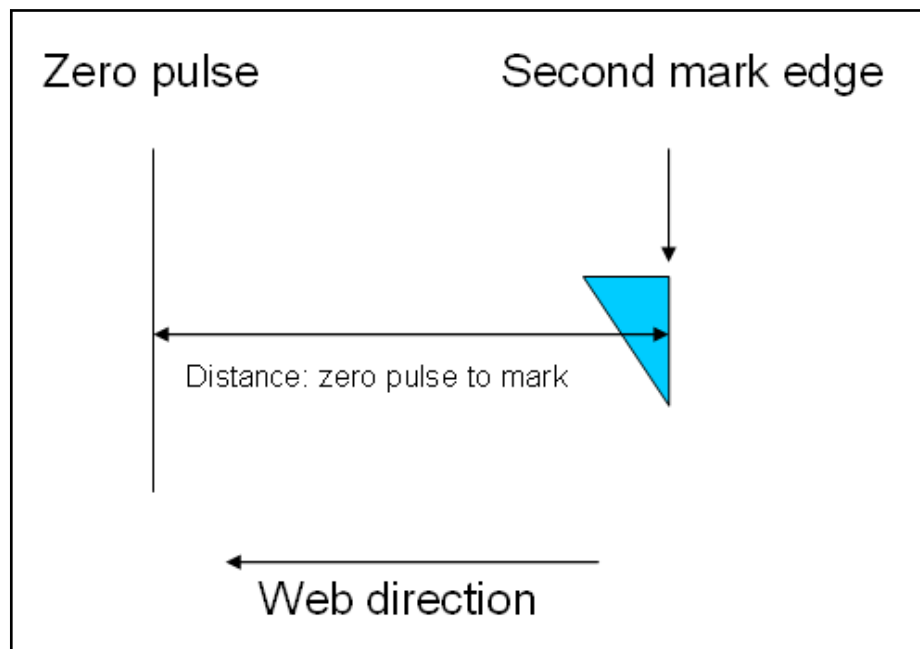


Fig. 3-25: Second mark edge



It is recommended to **always** use the vertical edge for the length register control. If the inclined mark edge is used, an adjustment in the side register always results in an offset in the length register.

LengthRegisterMode: Indicates the respective web measuring principle for the length register. Furthermore, all required dimensions are saved for the respective web measuring principle. Currently, only Mode 1 of the return of all mark distances is implemented in the mark stream sensor.

Example:

```
LengthRegisterMode: MB_REGI_LENGTH_REGISTER_MODE := (ModeSelect:=1);
```

ModeSelect := 1 means that all mark distances of all marks are returned.

SideRegisterMode: Indicates which mark widths of which marks should be measured for the side register. If ModeSelect := 0, no mark widths are returned. If ModeSelect := 1, all mark widths are returned.

Example 1:

```
SideRegisterMode: MB_REGI_SIDE_REGISTER_MODE := (ModeSelect:=0);
```

No mark widths are returned.

Example 2:

```
SideRegisterMode: MB_REGI_SIDE_REGISTER_MODE := (ModeSelect:=1);
```

All mark widths are returned.

Currently, there is no function available that can specify two marks whose mark widths should be output.

CommunicationConfig: Contains the communication settings for the acyclic and the cyclic communication. The port on the PLC for the measured values and the port for the configuration data and diagnostic data as well as the IP address of the sensor and the PLC are specified.

Example:

```
CommunicationCfg: MB_REGI_COMMUNICATION_CONFIG := (IpSensor:='192.168.2.5', PortConfiguration:=6001, IpPLC:='192.168.2.3', PortMeasurement:=6000);
```

A configuration of the connection data is shown in the example above. The sensor has the IP address 192.168.2.5 and the acyclic communication is executed via the UDP port 6001 of the control. The IP address of the PLC is 192.168.2.3. Thus, the sensor knows that the cyclic measurement results should be sent to this IP address 192.168.2.3 and the UDP port 6000 of the control.

Tolerances:



For the tolerances, the respective default values should be used and only changed in an exceptional case.

Tolerances indicate tolerances for teaching, measuring and the robustness of the measurement:

- **TeachZeroPulseToMark:**
Indicates the maximum tolerance the BeginExpectationWindow may deviate from. The sensor sets its expectation window exactly on one position:
BeginExpectationWindow - TeachZeroPulseToMark.
- **TeachMarkToMark:**
Indicates a tolerance for teaching. That is the number of mm that may be attached as tolerance to the expectation window. This value should

RMB_TechRegiExtended

be equal to "MeasureMarkTolerance" to avoid that a mark is lost when starting the measurement.

- **MeasureMarkTolerance:**

During a measurement, an individual mark expectation window is used for each mark. That means that it is searched for the mark exactly in this expectation window due to its last position. If the position of the mark kept changing after the last revolution, the mark is not found.

The following three parameters are used for the robustness of the measurement. That means that a mark was detected with a certain width, height and position during the last measurement. If the mark found deviates from the absolute position during the next measurement by the respective percentage, this measurement is ignored and it is searched again with the measured data last valid. This should ensure that a mark can be found during next measurement in case of an disturbance occurring once (e.g. due to a crease).

- **DeviationOfMarkWidth:**

Maximum deviation of the mark width from the last measurement allowed in percent. If this value is set to 100 %, the monitoring of the mark width is switched off.

- **DeviationOfMarkHeight:**

Maximum deviation of the mark height from the last measurement allowed in percent. If this value is set to 100 %, the monitoring of the mark height is switched off.

- **DeviationOfLastPosition:**

Maximum absolute deviation of the mark position from the last measurement allowed.

- **ShiftSyncPulse:**

Indicates the distance between the shifted zero pulse and the beginning of the expectation window. To ensure a high measurement accuracy during acceleration phase, the zero pulse is shifted up to the beginning of the expectation window with regard to the hardware. This value indicates the tolerance between the shifted zero pulse and the beginning of the expectation window. If it is to be expected that the mark stream does not keep moving forward, a default value of 5 mm is recommended.

InternalParameter:

This data structure includes internal parameters of the sensor that should only be changed if instructed by Bosch Rexroth. Color and intensity can be scaled here. The default values are values from experience. The structure also contains the gain transmitted during easy-teach-in.

- **ColorWeighting:**

Indicates the color weighting in the analog signal.

- **IntensityWeighting:**

Indicates the intensity weighting in the analog signal.

- **Gain:**

Indicates the current gain of the sensor written during easy teach-in. This gain is generally read out via the diagnostic function block MB_RegiSensor01ReadDiagType01.

Sending and receiving data packages

The function block indicates with the incrementation from "NewData" to "SendData" that it wants to send new data. It is sent via the MB_RegiSensor01SendRecvDataType01 function block. If it receives new data, it sets the "NewData" value equal to the "RecvData" value. Thus, the function block

RMB_TechRegiExtended

knows that it is the respected response package. A response package is expected with each transmitted data package.

Timeout:

If a new data package is sent, an internal timer runs in the function block. If it does not get a response within 10 s, it aborts with an error due to timeout.

A possible error case is that the MB_RegiSensor01SendRecvDataType01 function block is not activated or incorrectly connected.

Error handling

The function block uses the error table F_RELATED_TABLE, 16#0170. It can generate the following error messages in Additional1/Additional2.

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000000	Timeout; no response from the MB_RegiSensor01SendRecvDataType01 function block within 10 seconds
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000001	Timeout while teaching. Teaching could not be completed in the time specified.
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000000	Invalid plate cylinder format
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000001	Invalid mark type
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000002	Invalid mark geometry
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000003	Invalid number of bar code blocks
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000004	Invalid number of marks in the mark stream
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000005	Invalid mark distance
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000006	Invalid length register mode
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000007	Invalid length register parameter
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000008	Invalid side register mode
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000009	Invalid sensor IP
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000000A	Invalid control IP
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000000B	Invalid UDP port for the measurement
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000000C	Invalid configuration port
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000000D	Invalid number of increments
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000000E	Invalid measurement tolerance
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000000F	Invalid teaching tolerance of a mark
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000010	Invalid teaching tolerance, beginning of expectation window
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000011	Invalid beginning of expectation window
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000012	Invalid size of expectation window
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000013	Invalid tolerance for mark width
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000014	Invalid tolerance for mark height
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000015	Invalid tolerance for mark position

RMB_TechRegiExtended

ErrorID	Additional 1	Additional 2	Description
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000016	Invalid "ShiftSyncPulse" tolerance
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000017	Invalid color weighting
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000018	Invalid intensity weighting
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000019	Invalid "Measurement" mode
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000001A	Invalid gain
COMMUNICATION_ERROR (16#0002)	16#0C001D10	16#00000002	Timeout during bar code search
OTHER_ERROR (16#7FFE)	16#00001D12	16#xxxxxxx	Error handling for the mark stream sensor (see table "Mark stream sensor") is not integrated in "F_RELATED_TABLE"

Tab. 3-29: Error codes of the MB_RegiSensor01ConfigType01 function block

Additionally, the function block can output errors of the mark stream sensor.

3.3.4 MB_RegiSensor01SendRecvDataType01

Brief description

The MB_RegiSensor01SendRecvDataType01 function block communicates with the sensor. Therefore, the function blocks transfer their data to be sent and get the respective data packages in return for the following: sensor configuration, sending velocity information and acceleration information. The diagnostic function block sends its data and gets the respective data packages in return during an acceleration phase. The function block prevents a collision of function blocks accessing the same UDP port simultaneously. A prioritization is implemented as sending of the acceleration information is time-sensitive. Therefore, data packages of the MB_RegiSensor01SendSpeedAccType01 function block are sent immediately, as soon as sending of the data is possible. Configuration data of the MB_RegiSensor01ConfigType01 function block, diagnostic data of the MB_RegiSensor01ReadDiagType01 function block or teach data of the MB_RegiSensor01TeachDataType01 function block have a lower priority than the MB_RegiSensor01SendSpeedAccType01 function block.

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V11	RMB_TechRegiExtended

Tab. 3-30: Reference table of the MB_RegiSensor01SendRecvDataType01 function block

Interface description

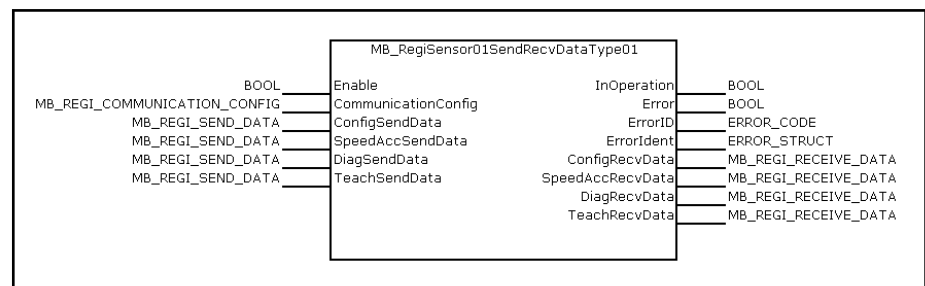


Fig. 3-26: MB_RegiSensor01SendRecvDataType01 function block

RMB_TechRegiExtended

I/O type	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Processing of function block enabled.
	CommunicationConfig	MB_REGI_COMMUNICATION_CFG	Contains the UDP port of the control to which the control connects to and to which the sensor sends the measured data.
	ConfigSendData	MB_REGI_SEND_DATA	Includes data that the MB_RegiSensor01ConfigType01 function block wants to send.
	SpeedAccSendData	MB_REGI_SEND_DATA	Includes data that the MB_RegiSensor01SendSpeedAccType01 function block wants to send.
	DiagSendData	MB_REGI_SEND_DATA	Includes data that the MB_RegiSensor01ReadDiagType01 function block wants to send.
	TeachSendData	MB_REGI_SEND_DATA	Includes data that the MB_RegiSensor01TeachData-Type01 function block wants to send.
VAR_OUTPUT	InOperation	BOOL	Function block is working.
	Error	BOOL	Processing completed with errors.
	ErrorID	ERROR_CODE	Error group
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics
	ConfigRecvData	MB_REGI_RECEIVE_DATA	Includes data that was received for the MB_RegiSensor01ConfigType01 function block.
	SpeedAccRecvData	MB_REGI_RECEIVE_DATA	Includes data that was received for the MB_RegiSensor01SendSpeedAccType01 function block.
	DiagRecvData	MB_REGI_RECEIVE_DATA	Includes data that was received for the MB_RegiSensor01ReadDiagType01 function block.
	TeachRecvData	MB_REGI_RECEIVE_DATA	Includes data that was received for the MB_RegiSensor01TeachData-Type01 function block.

Tab. 3-31: I/O interface of the MB_RegiSensor01SendRecvDataType01 function block

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
MB_REGI_COMMUNICATION_CONFIG					
IpSensor	STRING	LEN > 0	n. def	""	Rising edge at "Enable"
PortConfiguration	UINT	1	n. def	0	Rising edge at "Enable"
IpPLC	STRING	LEN > 0	n. def	""	Rising edge at "Enable"
PortMeasurement	UINT	1	n. def	0	Rising edge at "Enable"
MB_REGI_SEND_DATA					
NoOfBytes	UDINT	0	n. def	0	Change at "NewData"
ValueAddr	POINTER TO BYTE	n. def	n. def	0	Change at "NewData"

RMB_TechRegiExtended

Name	Type	Min. value	Max. value	Default value	Effective
NewData	UINT	0	65535	0	Continuous
SensorPort	UINT	1	65535	0	Change at "NewData"

Tab. 3-32: Overview on min./max. values and initial values of the inputs

Data structures

The data structures [tab. 3-24 "MB_REGI_SEND_DATA data structure" on page 43](#) and [tab. 3-21 "MB_REGI_COMMUNICATION_CONFIG data structure" on page 42](#) are described in the [chapter 3.3.3 "MB_RegiSensor01ConfigType01" on page 39](#) function block.

Function description

If a communication should be established with a sensor, this function block has to be started at a rising edge at "Enable" before or at the same time with the attached function blocks MB_RegiSensor01SendSpeedAccType01, MB_RegiSensor01ConfigType01, MB_RegiSensor01TeachDataType01 and MB_RegiSensor01ReadDiagType01. At a rising edge at "Enable" and after checking the inputs, the respective UDP port "PortConfiguration" is bound to the control. Thus, the respective data can be received on this port. The function block waits for data. That means that it checks during each call whether the "NewData" input changes for the structures "ConfigSendData", "AccelerationSendData", "DiagSendData" and "TeachSendData". If this is the case, the respective data (NoOfBytes) is sent. The function block waits for the respective data if a response package is expected from the sensor.

Exception: No response package is expected when transmitting acceleration and velocity data.

A timeout handling also takes place. If no response package has been received after 2 seconds, the data package is sent again. An error is generated after third try and the transmission is aborted. If a package is received, the number of bytes and the address (ValueAddr) at which the data is stored are specified. NewData at the output is set to the same value NewData received at the input when sending the data. Thus, it can be exactly allocated to which sent data package the response belongs to.

Data buffer: No **data buffer** is internally implemented since each created function block (if a handshake takes place) sends the next package only after it received the respective response.

Exception: If there is a new package with velocity and acceleration information and if it cannot be sent until a new package is created, the new package is sent and the old one is discarded.

Prioritization: The transferred configuration data is not time-critical. Only the acceleration and velocity values has to be transferred as fast as possible. Thus, a prioritization is implemented. The data package of the MB_RegiSensor01SendSpeedAccType01 function block has absolute priority and is immediately sent as soon as the "send" function block is empty. If it is waited for a response package of the configuration or diagnostics within this period, it is first stopped, the time-critical package is sent in order to subsequently receive the configuration/diagnostic package. The prioritization is shown in the following figure.

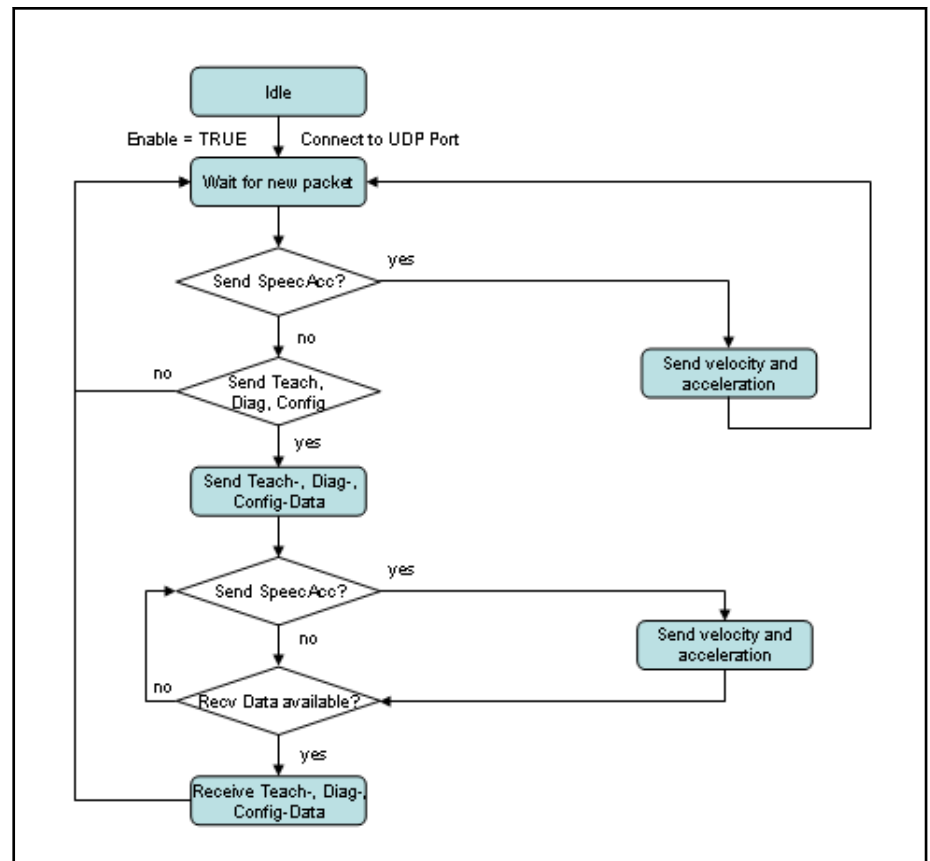


Fig. 3-27: Internal sequence of the MB_RegiSensor01SendRecvDataType01 function block

Error handling

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000003	Timeout while connecting to the UDP port
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000001B	Invalid UDP port, port configuration
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000004	Timeout while sending
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000005	Timeout while receiving
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000001C	Invalid sensor IP

Tab. 3-33: Error codes of the MB_RegiSensor01SendRecvDataType01 function block

Additionally, the function block can output errors of the mark stream sensor and of the library RIL_SocketComm.lib.

3.3.5 MB_RegiSensor01CycComType01

Brief description

The MB_RegiSensor01CycComType01 function block executes the cyclic communication with the sensor via the Ethernet interface. It receives measured sensor values, evaluates them and returns the mark distance or the dif-

RMB_TechRegiExtended

ference of the mark widths of two registration marks. One measured value is returned for the length register and one for the side register. The communication with the sensor is executed internally. If measured values of several registration marks are required, the MB_RegiSensor01CycComType02 function block is to be used.

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V11	RMB_TechRegiExtended

Tab. 3-34: Reference table of the MB_RegiSensor01CycComType01 function block

Interface description

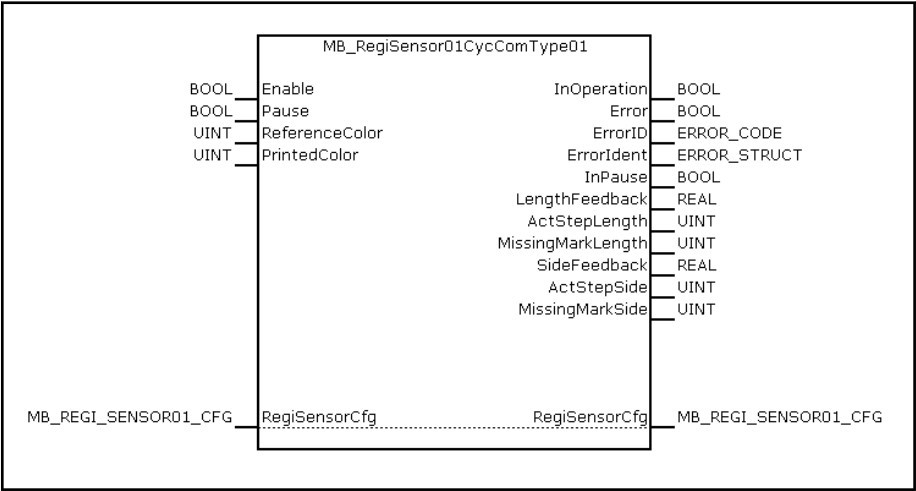


Fig. 3-28: The MB_RegiSensor01CycComType01 function block

I/O type	Name	Data type	Description
VAR_IN_OUT	RegiSensorCfg	MB_REGI_SENSOR01_CFG	Contains the current configuration data of the sensor.
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled).
	Pause	BOOL	Cancels the function block processing. The value at "LengthFeedback" and "SideFeedback" remains constant.
	ReferenceColor	UINT	Indicates the reference color with which the currently printed mark is compared to. ReferenceColor = 1 means that the currently printed mark is compared to the first mark of the mark stream.
	PrintedColor	UINT	Indicates the position of the mark printed in the mark stream by the controlling printing unit.

I/O type	Name	Data type	Description
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Error group
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics
	InPause	BOOL	Indicates that the "Pause" input is active and that the values in "LengthFeedback" and "SideFeedback" do not change.
	LengthFeed-back	REAL	The mark distance calculated for the length register.
	ActStepLength	UINT	Contains the sequence number of the last measurement package if a new valid measured value is present for the length register.
	MissingMarkLength	UINT	Is incremented if an invalid measured value is present for the length register.
	SideFeedback	REAL	The actual value for the side register.
	ActStepSide	UINT	Contains the sequence number of the last measurement package if a new valid measured value is present for the side register.
	MissingMarkSide	UINT	Is incremented if an invalid measured value is present for the side register.

Tab. 3-35: I/O interface of the MB_RegiSensor01CycComType01 function block

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
Pause	BOOL			FALSE	Continuous
ReferenceColor	UINT	0	16	0	Continuous
PrintedColor	UINT	0	16	0	Continuous

Tab. 3-36: Overview on min./max. values and initial values of the inputs

Data structures

The data structure [tab. 3-14 "MB_REGI_SENSOR01_CFG data structure" on page 41](#) used was described in the [chapter 3.3.3 "MB_RegiSensor01ConfigType01" on page 39](#) function block.

Function description

The function block may only be activated after the MB_RegiSensor01ConfigType01 function block was called to configure at a rising edge at "Enable" since the MB_RegiSensor01CycComType01 function block accesses the UDP port of the RegiSensorCfg structure and then connects. If no configuration has yet been executed, a faulty UDP port is present, since this port has not been written to the structure. In this case, the respective "Error" output of the function block is set.

The PLC works as UDP server in the cyclic communication. That means the control receives the packages of the sensor with its measured values at the UDP port via the acyclic communication. The sensor sends the measured data cyclically after each cylinder revolution (that means after each measurement evaluation). This measured data is evaluated by the function block. When setting "Enable", a bind() is made on UDP port defined in "RegiSen-

RMB_TechRegiExtended

sorCfg". Thus, it can be listened on the UDP port and measured values can be received and evaluated.

ReferenceColor und PrintedColor:

The inputs "ReferenceColor" and "PrintedColor" indicate which marks are measured to each other.

If ReferenceColor = 1 and PrintedColor = 4 for example, the mark distance between the marks 1 and 4 of the mark stream is returned. This allows switching from reference color to previous color during operation by selecting ReferenceColor = 3. "PrintedColor" is a mark that is printed from the controlling printing unit.

If a new actual value is calculated, it is output. "ActStepLength" and "ActStepSide" contain the sequence number of the last measurement package. Thus, the register controller knows that a new measured value is present.

If there is an invalid measured value that does not allow the evaluation of the marks "ReferenceColor" and "PrintedColor", "MissingMarkLength/Side" is incremented and "ActStepLength/Side" remains unchanged. If there is a new valid value, "MissingMarkLength/Side" is reset again.

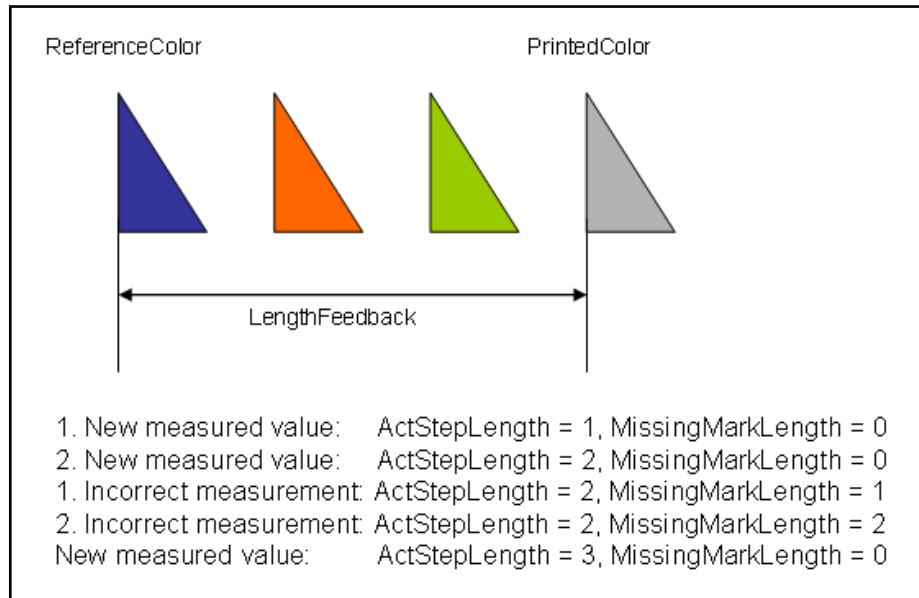


Fig. 3-29: Behavior of the outputs in case of incorrect measurements

The figure above shows the behavior of the outputs for the length register measurement as example. The side register behavior is equivalent. Registration mark 1 was selected as reference color, registration mark 4 as printed mark and thus, "LengthFeedback" is the distance between the marks 4 and 1. In case of a new measured value, "LengthFeedback" is updated and "ActStepLength" is incremented. "ActStepLength" and "LengthFeedback" are frozen at the first incorrect measurement and "MissingMarkLength" is incremented. In case of a new valid measurement, "MissingMarkLength" is set to 0, "ActStepLength" is incremented and the current measured value "LengthFeedback" is output.

Error handling

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000003	Timeout while connecting
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000001D	"ReferenceColor" is not within the defined limits
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000001E	"PrintedColor" is not within the defined limits
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000001F	"PrintedColor" and "ReferenceColor" are equal
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000020	No valid configuration found
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000021	"PrintedColor" or "ReferenceColor" are not within the range of the marks defined
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000000B	Invalid UDP port for the measurement

Tab. 3-37: Error codes of the MB_RegiSensor01CycComType01 function block

Additionally, the function block can output errors of the mark stream sensor and of the library RIL_SocketComm.lib.

3.3.6 MB_RegiSensor01CycComType02

Brief description

The MB_RegiSensor01CycComType02 function block is required to communicate with the mark stream sensor. It executes only the cyclic communication. The sensor is configured by the MB_RegiSensor01ConfigType01 function block. Only after the sensor has been configured directly, it is started with the measurement and the function block can start the measurement evaluation. All measured values are specified for the length register and the side register at the output. This data can be evaluated by the user according to the configuration. This MB_RegiSensor01CycComType02 function block is especially suitable for applications in which several printed marks are measured to each other. This is for example the case if only one mark stream sensor is located at the end of the machine.

An exemplary application is shown in the following figure:

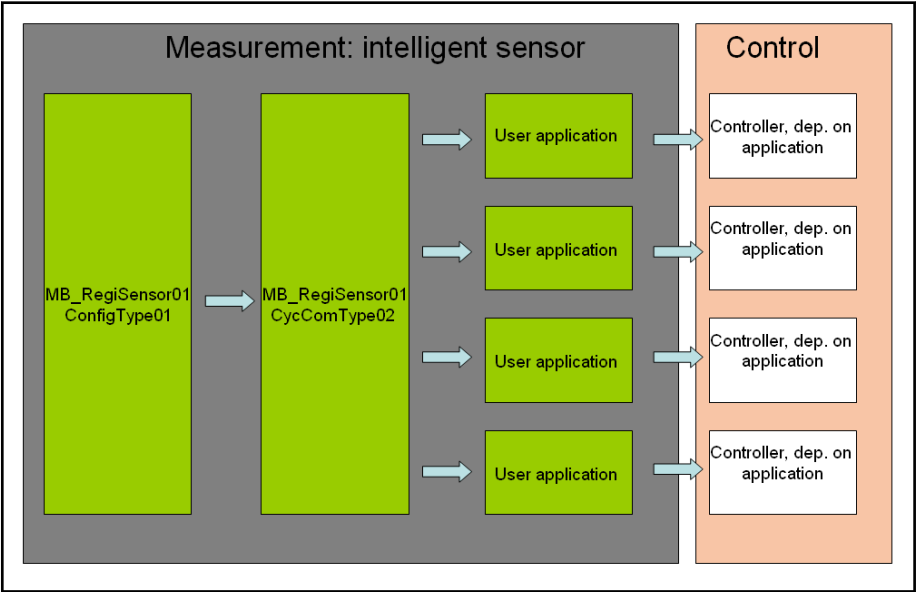


Fig. 3-30: Exemplary application of the MB_RegiSensor01CycComType02 function block

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V11	RMB_TechRegiExtended

Tab. 3-38: Reference table of the MB_RegiSensor01CycComType02 function block

Interface description

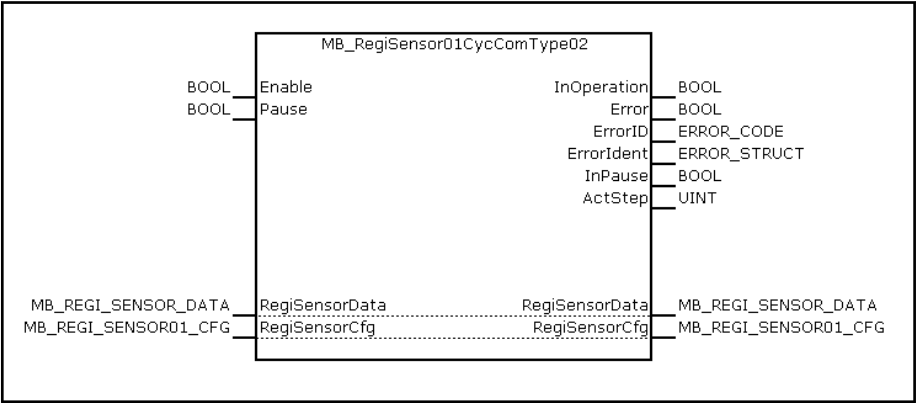


Fig. 3-31: The MB_RegiSensor01CycComType02 function block

I/O type	Name	Data type	Description
VAR_IN_OUT	RegiSensorCfg	MB_REGI_SENSOR01_CFG	Contains the current configuration data of the sensor.
	RegiSensorData	MB_REGI_SENSOR01_DATA	Contains the respective measured values of the sensor.
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled).
	Pause	BOOL	Cancels the function block processing. No new measured values are evaluated. The last measured data is frozen.

I/O type	Name	Data type	Description
VAR_OUTPUT	InOperation	BOOL	Function block is working.
	Error	BOOL	Processing completed with errors.
	ErrorID	ERROR_CODE	Error group
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics
	InPause	BOOL	Indicates that the "Pause" input is active and that the value in "RegiSensorData" does not change.
	ActStep	UINT	Indicates that a new data package with measured values was received and evaluated.

Tab. 3-39: I/O interface of the MB_RegiSensor01CycComType02 function block

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
Pause	BOOL			FALSE	Continuous

Tab. 3-40: Overview on min./max. values and initial values of the inputs

Data structures

MB_REGI_SENSOR01_DATA		
ActStep	UINT	Current step
LengthFeedback	ARRAY [0..16] OF REAL	It contains the mark distance from the zero pulse to the marks 1..16. In [0], the mark distance stands for web/web2 if a mark on the upper side was measured with one on the lower side.
SideFeedback	ARRAY [1..16] OF REAL	Contains the respective mark widths.
MarkState	ARRAY[0..16] OF MB_REGI_MARK_STATE	Status of the registration marks.
SensorState	MB_REGI_SENSOR_STATE	Outputs warnings and diagnostics of the sensor during measurement.

Tab. 3-41: MB_REGI_SENSOR01_DATA data structure

The data structure [tab. 3-14 "MB_REGI_SENSOR01_CFG data structure" on page 41](#) used was described in the [chapter 3.3.3 "MB_RegiSensor01Config-Type01" on page 39](#) function block.

Enumeration type

The enumeration type MB_REGI_MARK_STATE can assume the following values:

Element	Value	Description
FOUND	0	Mark found
ROTATED	1	Mark rotated
LARGE	2	Mark too large
SMALL	3	Mark too small

RMB_TechRegiExtended

Element	Value	Description
MISSING	4	Mark missing
NOT_CONFIGURED	5	Mark not configured

Tab. 3-42: Enumeration type MB_REGI_MARK_STATE

The enumeration type MB_REGI_SENSOR_STATE can assume the following values:

Element	Value	Description
START	0	Starting state
SEARCHING	1	Search marks
INOP	2	Marks found (operating mode)
STOP	3	Web velocity below 0,4m/s – no evaluation
LAMP	4	Lamp defective
SENSOR	5	Sensor dirty
TEACHING	6	Sensor in teach phase

Tab. 3-43: Enumeration type MB_REGI_SENSOR_STATE

Function description

The function block may only be activated after the MB_RegiSensor01Config-Type01 function block was called to configure at a rising edge at "Enable" since the MB_RegiSensor01CycComType02 function block accesses the UDP port of the RegiSensorCfg structure and then connects. If no configuration has yet been executed, a faulty UDP port is present, since this port has not been written to the structure. In this case, the respective "Error" output of the function block is set.

The PLC works as UDP server in the cyclic communication. That means the control receives the measured value packages of the sensor at the UDP port via the acyclic communication. The sensor sends the measured data cyclically after each cylinder revolution (that means after each measurement evaluation). This measured data is evaluated by the function block. When setting "Enable", a bind() is made on UDP port defined in "RegiSensorCfg". Thus, it can be listened on the UDP port and measured values can be received and evaluated.

Each sensor thus requires 2 ports on the control: One for configuration data and diagnostic data and one for the measured value acquisition. The communication with the sensor to receive measured values is internally takes place in the function block.

The **RegiSensorData** contains all measured values.

LengthFeedback[i]:

The "LengthFeedback" array contains all measured mark distances with regard to the zero pulse as well as a measured value of the web/web2 procedure that are entered into the array as follows:

LengthFeedback[0]: Mark distance between the defined mark below and the mark above.

LengthFeedback[1]: Mark distance between the zero pulse and mark 1.

LengthFeedback[2]: Mark distance between the zero pulse and mark 2.

...

LengthFeedback[15]: Mark distance between the zero pulse and mark 15.

LengthFeedback[16]: Mark distance between the zero pulse and mark 16.
If a measured value is not available since it has not been configured or if less marks are available, the return value is 0.0 and defined correspondingly as "Missing" or "Not_Configured" in the "MarkState".

SideFeedback[i]: The "SideFeedback" array contains all measured mark widths that are entered in the array as follows:
SideFeedback[1]: Mark width of mark 1
SideFeedback[2]: Mark width of mark 2
...
SideFeedback[16]: Mark width of mark 16
If a measured value is not available since it has not been configured or if less marks are available, the return value is 0.0 and defined correspondingly as "Missing" or "Not_Configured" in the "MarkState".

SensorState: It is not set.

MarkState[i]: Indicates the validity of the values in the arrays "LengthFeedback" and "Side-Feedback". If the measured value of mark 1 is valid for example, then, MarkState[1] = FOUND. Thus, the values "LengthFeedback[1]" and "Side-Feedback[1]" can be evaluated. Otherwise, MarkState[1] = Missing or Not_Configured. That depends on the fact whether the mark was omitted or not configured.

Error handling

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000003	Timeout while connecting
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000020	No valid configuration found
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000000B	Invalid UDP port

Tab. 3-44: Error codes of the MB_RegiSensor01CycComType02 function block

Additionally, the function block can output errors of the mark stream sensor and of the library RIL_SocketComm.lib.

3.3.7 MB_RegiSensor01ConfigType02

Brief description

The MB_RegiSensor01CommType02 function block contains the individually implemented function blocks:

- MB_RegiSensor01ConfigType02
- MB_RegiSensor01CycComType04
- MB_RegiSensor01ReadDiagType01
- MB_RegiSensor01SendRecvDataType02
- MB_RegiSensor01SendSpeedAccType01 and
- MB_RegiSensor01TeachDataType01

The function blocks are internally connected in the function block and provide a user-friendly interface to execute the individual functions. The function block facilitates the full control of the mark stream sensor.



The following function block to set the IP address is not integrated as it is only required once per machine:

- MB_RegiSensor01SetIPAddressType01



The function block MB_RegiSensor01CommType02 has to be called in a Motion-synchronous task to ensure a correction calculation of the acceleration. If this is not possible due to the runtime, the "mMotionCyclic()" function of the function block from the Motion-synchronous tasks has to be called and the MB_RegiSensor01CommType02 function block can be called in a cyclic task. The call of the MotionTask has to be provided with the following structure: {CyclicTask}.fbRegiSensor01CommType02.{fbInstanz}.mMotionCyclic(). The elements in the curly brackets have to be replaced by the program name in the cyclic task and the instance name of the function block. Thus, the method call from a program running in the Motion task could be as follows: PlcProg.fbRegiSensor01CommType02.fbRegiSensor01CommType02.mMotionCyclic(). If this call is not executed, a correct calculation of the acceleration cannot be ensured.



The following extensions have been implemented in the MB_RegiSensor01CommType02 function block:

- The MB_REGI_INTERNAL_PARAMETER_TYPE02 data structure has been modified
- The color weighting now also defined the intensity weighting
- MarkAnalogLevel specifies the analog level from which a mark is detected as valid mark
- BarcodeSearchLevel specified the probability from which a bar code is detected as valid bar code
- The BarcodeFoundProbability output specifies the probability with which a bar code has been determined during the last bar code scan
- The gain, the color weighting, the MarkAnalogLevel and the BarcodeSearchLevel can now be set individually using the MB_REGI_SENSOR01_COMM_ACTIONS action
- Double marks are valid
- It is verified that a sensor firmware 12VRS is used; otherwise an error is returned



The MB_RegiSensor01CommType02 function block has to be used in combination with the sensor firmware 12 VRS.

Using this function block, additionally required settings can be selected for the sensor firmware 12 which cannot be set using the MB_RegiSensor01CommType01 function block.

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V14	RMB_TechRegiExtended

Tab. 3-45: Reference table of the MB_RegiSensor01ConfigType02 function block

Interface description

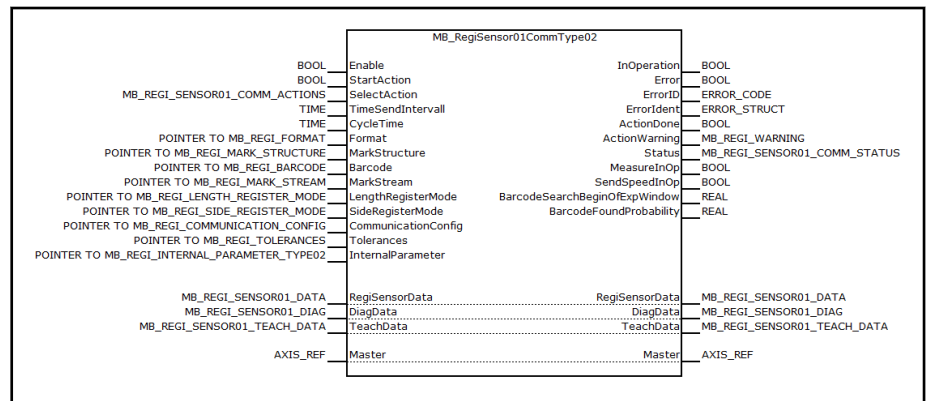


Fig. 3-32: MB_RegiSensor01CommType02 function block

I/O type	Name	Data type	Explanation
VAR_IN_OUT CONST	Master	AXIS_REF	Addressing the master axis
VAR_IN_OUT	RegiSensor-Data	MB_REGI_SENSOR01_DATA	Contains the respective measured values of the sensor
	DiagData	MB_REGI_SENSOR01_DIAG	Diagnostic data of the sensor
	TeachData	MB_REGI_SENSOR01_TEACH_DATA	Sensor data to be read or written
VAR_INPUT	Enable	BOOL	Processing of function block enabled (level-controlled)
	StartAction	BOOL	Starts the selected action
	SelectAction	MB_REGI_SENSOR01_COMM_ACTIONS	Selecting the executable actions
	TimeSendInterval	TIME	Data of the velocity and acceleration information is calculated and sent in this interval
	CycleTime	TIME	Cycle time at which the function block is called.
	Format	POINTER TO MB_REGI_FORMAT	Defines the format characteristics
	MarkStructure	POINTER TO MB_REGI_MARK_STRUCTURE	Defines the mark structure
	Bar code	POINTER TO MB_REGI_BARCODE	Defines the bar code
	MarkStream	POINTER TO MB_REGI_MARK_STREAM	Defines the mark stream
	LengthRegisterMode	POINTER TO MB_REGI_LENGTH_REGISTER_MODE	Defines the web measuring principle for the length register
	SideRegisterMode	POINTER TO MB_REGI_SIDE_REGISTER_MODE	Defines the web measuring method for the length register
	CommunicationConfig	POINTER TO MB_REGI_COMMUNICATION_CONFIG	Defines the communication characteristics of the UDP connection
	Tolerances	POINTER TO MB_REGI_TOLERANCES	Defines the tolerances allowed for teaching and measurement
	InternalParameter	POINTER TO MB_REGI_INTERNAL_PARAMETER_TYPE02	Internal sensor parameters that should only be changed if specifically mentioned.

RMB_TechRegiExtended

I/O type	Name	Data type	Explanation
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in the event of an error
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics
	ActionDone	BOOL	Action was completed successfully
	ActionWarning	MB_REGI_WARNING	Contains warnings of failed actions or warnings of the sensor system that do not result in a function block error
	Status	MB_REGI_SENSOR01_COMM_STATUS	Returns the current MB_RegiSensor01Comm-Type02 function block status
	MeasurementOp	BOOL	Measurement running
	SendSpeedOp	BOOL	Transmitting velocity and acceleration values.
	Barcode-SearchBeginOfExpWindow	REAL	This value contains the position of the bar code end
	Barcode-FoundProbability	REAL	Probability at which the bar code was found

Tab. 3-46: I/O interface of the MB_RegiSensor01CommType02 function block

Data structures

The following data structures have been described in the previous function blocks of the RMB_TechRegiExtended library:

- [tab. 3-59 "MB_REGI_FORMAT data structure" on page 87](#)
- [tab. 3-60 "MB_REGI_MARK_STRUCTURE data structure" on page 87](#)
- [tab. 3-61 "MB_REGI_BARCODE data structure" on page 88](#)
- [tab. 3-62 "MB_REGI_MARK_STREAM data structure" on page 88](#)
- [tab. 3-63 "MB_REGI_LENGTH_REGISTER_MODE data structure" on page 88](#)
- [tab. 3-64 "MB_REGI_SIDE_REGISTER_MODE data structure" on page 88](#)
- [tab. 3-65 "MB_REGI_COMMUNICATION_CONFIG data structure" on page 88](#)
- [tab. 3-66 "MB_REGI_TOLERANCES data structure" on page 89](#)
- [tab. 3-81 "MB_REGI_SENSOR01_DATA data structure" on page 107](#)
- [tab. 3-89 "MB_REGI_SENSOR01_DIAG data structure" on page 111](#)
- [tab. 3-98 "MB_REGI_SENSOR01_TEACH_DATA data structure" on page 118](#)

MB_REGI_INTERNAL_PARAMETER_TYPE02		
ColorWeighting	REAL	Color weighting of the sensor. Defines the intensity weighting with 1-ColorWeighting.
Gain	UINT	Sensor gain.
MarkAnalogLevel	UINT	Analog level from which a mark is detected as valid mark.
BarcodeSearchLevel	REAL	Probability from which a bar code is detected as valid bar code.

Tab. 3-47: MB_REGI_INTERNAL_PARAMETER_TYPE02 data structure

MB_REGI_WARNING		
Warning	BOOL	Indicated that a warning has been reported
WarningAdditional1	DWORD	Shows Additional 1.
WarningAdditional2	DWORD	Shows Additional 2.

Tab. 3-48: MB_REGI_WARNING data structure

Enumeration type

The enumeration type MB_REGI_SENSOR01_COMM_STATUS can assume the following values:

Element	Value	Description
WAIT_FOR_NEXT_ACTION	0	Function block waits for the subsequent action
CONFIGURATION_IN_WORK	1	Configuration is transmitted
BARCODE_SEARCH_IN_WORK	2	Bar code search is running
MARKSTREAM_SEARCH_IN_WORK	3	Mark stream search is running
TEACH_IN_IN_WORK	4	TeachIn is running
EASY_TEACH_IN_IN_WORK	5	EasyTeachIn in running
READ_DIAG_IN_WORK	6	ReadDiagData is running
READ_TEACH_DATA_IN_WORK	7	ReadTeachData is running
WRITE_TEACH_DATA_IN_WORK	8	WriteTeachData is running
MEASUREMENT_STARTING	9	Measurement is started
MEASUREMENT_STOPPING	10	Measurement is stopped
SEND_SPEED_STARTING	11	SendSpeed is started
SEND_SPEED_STOPPING	12	SendSpeed is stopped
SENSOR_OFF	13	The function block is not in state InOperation.
SENSOR_ERROR	14	Sensor in status Error
SET_GAIN_IN_WORK	15	The gain is set
SET_MARK_ANALOG_LEVEL_IN_WORK	16	The mark Analoglevel is set
SET_BARCODE_SEARCH_LEVEL_IN_WORK	17	The probability of the bar code threshold is set
SET_COLOR_WEIGHTING_IN_WORK	18	The color weighting is set

Tab. 3-49: Enumeration type MB_REGI_SENSOR01_COMM_STATUS

RMB_TechRegiExtended

The enumeration type MB_REGI_SENSOR01_COMM_ACTIONS can assume the following values:

Element	Value	Description
SET_CONFIG	1	Start configuration transmission
SEARCH_BARCODE	2	Start bar code search
SEARCH_MARK_STREAM	3	Start mark stream search
TEACH_IN	4	Start TeachIn
EASY_TEACH_IN	5	Start Easy TeachIn
READ_DIAG	6	Read diagnostic data
READ_TEACHING_DATA	7	Read Teach data
WRITE_TEACHING_DATA	8	Write Teach data
START_MEASURING	9	Start measurement
STOP_MEASURING	10	Stop measurement
START_SEND_SPEED	11	Start sending velocity information
STOP_SEND_SPEED	12	Stop sending velocity information
SET_GAIN	13	Transmit gain
SET_MARK_ANALOG_LEVEL	14	Set mark Analoglevel
SET_BARCODE_SEARCH_LEVEL	15	Set probability of bar code threshold
SET_COLOR_WEIGHTING	16	Set color weighting

Tab. 3-50: Enumeration type MB_REGI_SENSOR01_COMM_ACTIONS

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
StartAction	BOOL			FALSE	Continuous
SelectAction	MB_REGI_SENSOR01_COMM_ACTIONS	1	16	1 (SET_CONFIG)	In case of rising edge at StartAction and status = WAIT_FOR_NEXT_ACTION
TimeSendInterval	TIME	$\geq 2 \cdot \text{CycleTime}$	n. def	t#32ms	Action START_SEND_SPEED is started
CycleTime	TIME	> 0	n. def	t#8ms	Action START_SEND_SPEED is started
Format	POINTER TO MB_REGI_FORMAT	> 0	n. def	0	Rising edge at "Enable"
MarkStructure	POINTER TO MB_REGI_MARK_STRUCTURE	> 0	n. def	0	Rising edge at "Enable"
Bar code	POINTER TO MB_REGI_BARCODE	> 0	n. def	0	Rising edge at "Enable"
MarkStream	POINTER TO MB_REGI_MARK_STREAM	> 0	n. def	0	Rising edge at "Enable"

RMB_TechRegiExtended

Name	Type	Min. value	Max. value	Default value	Effective
LengthRegister-Mode	POINTER TO MB_REGI_LENGTH_REGISTER_MODE	> 0	n. def	0	Rising edge at "Enable"
SideRegister-Mode	POINTER TO MB_REGI_SIDE_REGISTER_MODE	> 0	n. def	0	Rising edge at "Enable"
Communication-Config	POINTER TO MB_REGI_COMMUNICATION_CONFIG	> 0	n. def	0	Rising edge at "Enable"
Tolerances	POINTER TO MB_REGI_TOLERANCES	> 0	n. def	0	Rising edge at "Enable"
InternalParameter	POINTER TO MB_REGI_INTERNAL_PARAMETER_TYPE02	> 0	n. def	0	Rising edge at "Enable"

Tab. 3-51: Overview on min./max. values and initial values of the inputs

In case of a rising edge at enable, it is checked if all pointers > 0 and thus, if the addresses containing the data structures are created. The data of the created pointers are checked when calling the individual function blocks. The data structures have the following default values and are applied or checked during the following actions:

Name	Type	Min. value	Max. value	Default value	Effective
MB_REGI_FORMAT					
FormatLength	REAL	> 0	n. def	0	Rising edge at "Enable"
Increments	UDINT	> 0	n. def	0	Rising edge at "Enable"
MB_REGI_MARK_STRUCTURE					
MarkType	UDINT	1	7	1 (the mark types 1,2,3,5,6,7 are implemented)	Action SET_CONFIG is started
SingleMark	ARRAY [0...3] OF REAL	> 0 (for required entries)	n. def	0	Action SET_CONFIG is started
MB_REGI_BARCODE					
NumberOfBlocks	UDINT	0 (1 and 2 blocks are invalid entries)	10	0	Action SET_CONFIG is started
BeginExpectationWindow	REAL	0	n. def	0	Action SET_CONFIG is started
SizeOfExpectationWindow	REAL	0	n. def	1	Action SET_CONFIG is started
FreeSpaceBeforeBarcode	REAL	0	n. def	0	Action SET_CONFIG is started
BarcodeBlocks	ARRAY [0...19] OF REAL	> 0 (for required entries)	n. def	0	Action SET_CONFIG is started

RMB_TechRegiExtended

Name	Type	Min. value	Max. value	Default value	Effective
MB_REGI_MARK_STREAM					
NumberOfMarks	UDINT	1	12	1	Action SET_CONFIG is started
MarkStream	Array [1...12] OF REAL	> 0 (for required entries)	n. def	0	Action SET_CONFIG is started
MarkDirection	BOOL			FALSE	Action SET_CONFIG is started
MB_REGI_LENGTH_REGISTER_MODE					
ModeSelect	UDINT	1	1	1	Action SET_CONFIG is started
LengthProperties	ARRAY [0...4] OF REAL	n. def	n. def	0	Action SET_CONFIG is started
MB_REGI_SIDE_REGISTER_MODE					
ModeSelect	UDINT	0	1	1	Action SET_CONFIG is started
SelectMarks	ARRAY [0...1] OF UINT	n. def	n. def	0	Action SET_CONFIG is started
MB_REGI_COMMUNICATION_CONFIG					
IpSensor	STRING	LEN > 0	n. def	""	Rising edge at "Enable"
PortConfiguration	UINT	1	n. def	0	Rising edge at "Enable"
IpPLC	STRING	LEN > 0	n. def	""	Rising edge at "Enable"
PortMeasurement	UINT	1	n. def	0	Rising edge at "Enable"
MB_REGI_TOLERANCES					
TeachZeroPulseToMark	REAL	1	n. def	5	Action SET_CONFIG is started
TeachMarkToMark	REAL	1	n. def	2	Action SET_CONFIG is started
MeasureMarkTolerance	REAL	1	n. def	10	Action SET_CONFIG is started
DeviationOfMarkWidth	REAL	1	100	20	Action SET_CONFIG is started
DeviationOfMarkHeight	REAL	1	100	100	Action SET_CONFIG is started
DeviationOfLastPosition	REAL	1	n. def	10	Action SET_CONFIG is started
ShiftSyncPulse	REAL	1	n. def	5	Action SET_CONFIG is started
MB_REGI_INTERNAL_PARAMETER_TYPE02					

Name	Type	Min. value	Max. value	Default value	Effective
ColorWeighting	REAL	0	< 1	0.5	Action SET_CONFIG or SET_COLOR_WEIGHTING is started
Gain	UINT	0	7	0	Action SET_CONFIG or SET_Gain is started
MarkAnalogLevel	UINT	0	n. def	100	Action SET_CONFIG or SET_MARK_ANALOG_LEVEL is started
BarcodeSearchLevel	REAL	0	< 100	50.0	Action SET_CONFIG or SET_BARCODE_SEARCH_LEVEL is started

Tab. 3-52: Overview of min./max. and default values of the referenced structures

The VAR_IN_OUTs RegiSensorData and DiagData do not have any min./max. and default values, as only internal function blocks and not the user is allowed to write on them. The corresponding data are stored in TeachData when reading the tech data. The created teach data, e.g. originating from a previously saved production, are written on the sensor when writing the teach data. This data structure cannot be interpreted.

Function description



For the function description, it is also recommended to consider the individual descriptions of the function blocks:

- [chapter 3.3.8 "MB_RegiSensor01ConfigType02" on page 85](#)
- [chapter 3.3.10 "MB_RegiSensor01CycComType04" on page 106](#)
- [chapter 3.3.11 "MB_RegiSensor01ReadDiagType01" on page 110](#)
- [chapter 3.3.9 "MB_RegiSensor01SendRecvDataType02" on page 102](#)
- [chapter 3.3.12 "MB_RegiSensor01SendSpeedAccType01" on page 112](#)
- [chapter 3.3.13 "MB_RegiSensor01TeachDataType01" on page 117](#)

Initialization

In case of a rising edge at Enable, it is checked for created pointers. The pointer contents of the respective data structure are directly transferred at the respective function blocks. Only after this is done, it is checked when the corresponding function block is called. Subsequently, the MB_RegiSensor01SendRecvDataTyp function block is started, executing the communication with the mark stream sensor. It is checked if the mark stream sensor can be contacted under the specified IP address using the "Ping" command. Subsequently, the function block configuration is transmitted using the MB_RegiSensor01ConfigType02 function block and the MB_RegiSensor01CycComType04 function block is started to receive the measured values. The initialization phase is completed and InOperation becomes TRUE. Each error that occurred in the initialization phase immediately results in a transition to the error status and Error = TRUE. In cyclic operation, it is differentiated between

RMB_TechRegiExtended

errors and warnings. The sensor is in state SENSOR_OFF during the initialization.

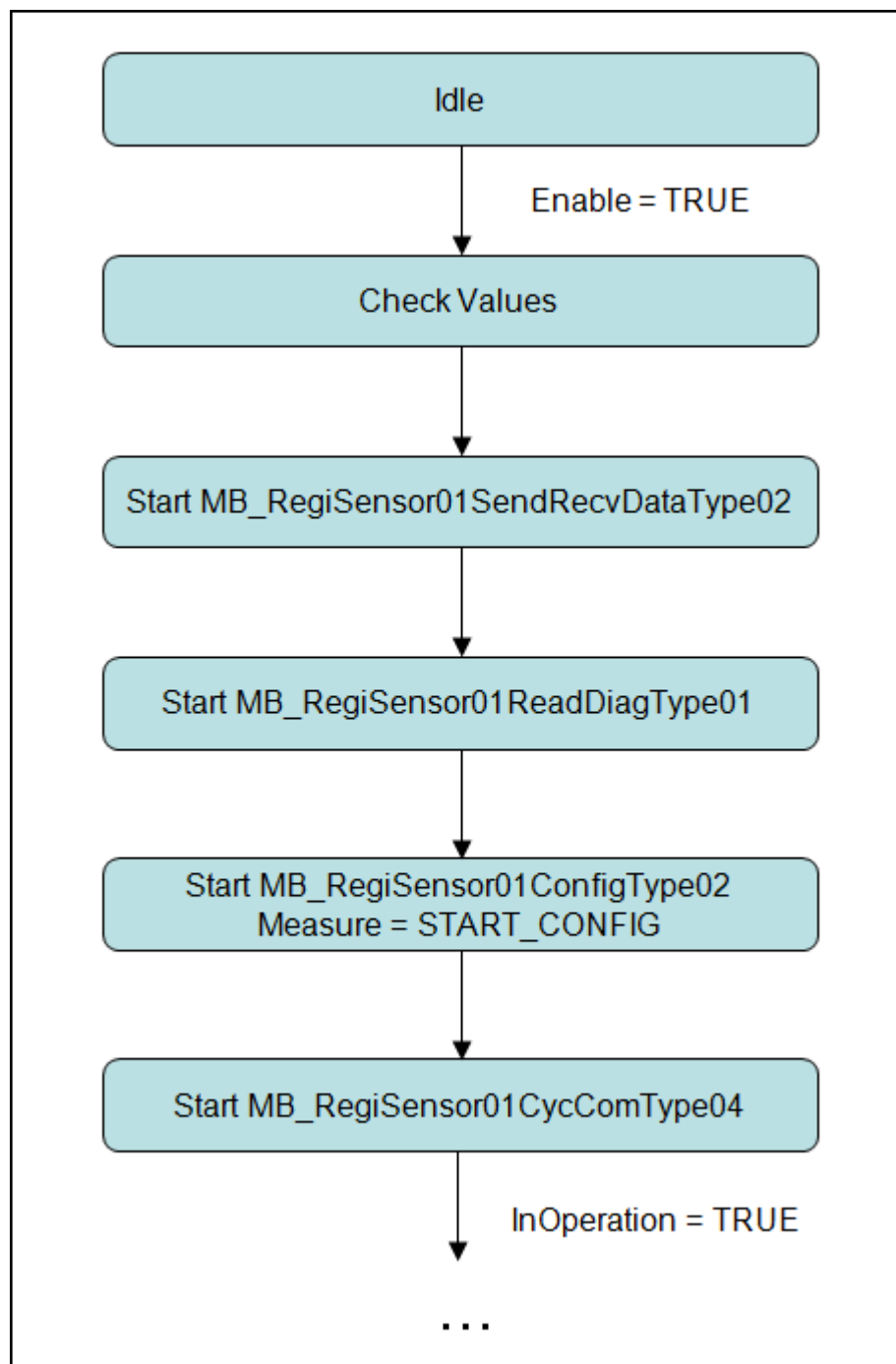


Fig. 3-33: Initialization phase sequence



During the initialization, it is checked if a measurement is still running on the mark stream sensor. If this is the case, the measurement is automatically stopped.

Cyclic operation

The following states can be reached in cyclic operation:

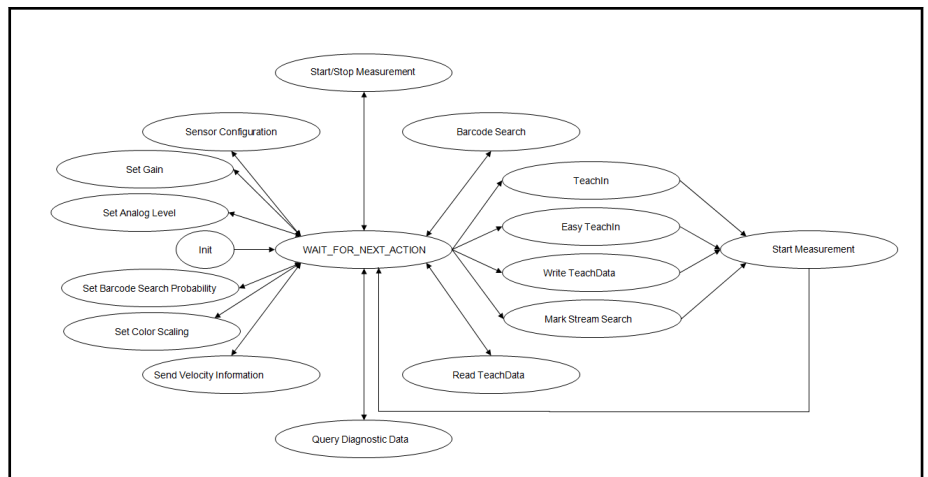


Fig. 3-34: State transition in cyclic operation

The function block is in status "WAIT_FOR_NEXT_ACTION" in cyclic operation (i.e. InOperation = TRUE). In this state, all actions defined in the "MB_REGI_SENSOR01_COMM_ACTIONS" enumeration type can be executed. The corresponding action has to be selected in SelectAction. Subsequently, the action has to be started with a rising edge at StartAction. While an action is running, the corresponding status of the function block can be read via status. ActionDone indicates a successfully completed action. No new action can be started if an action is already running. Three events can cause the action to be closed. ActionDone indicates a successfully completed action. If a severe communication error occurred, the function block is in error status (Error = TRUE). If an action was not completed successfully (e.g., not all marks have been found during TeachIn), the function block reports a warning (ActionWarning = TRUE) and returns to status "WAIT_FOR_NEXT_ACTION". The warning is automatically reset upon starting a new action.

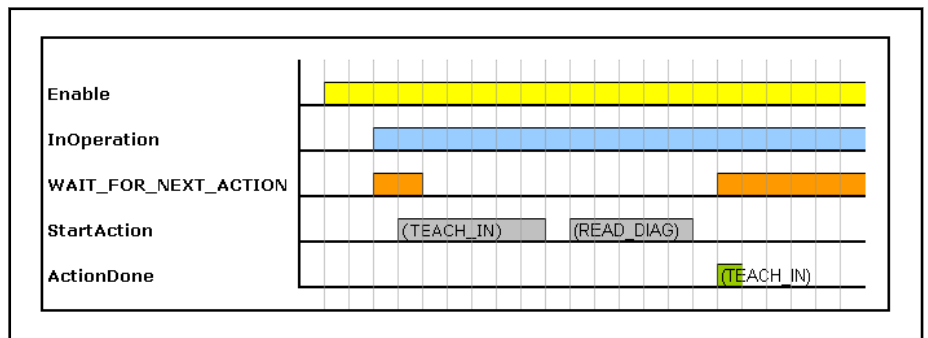


Fig. 3-35: Time diagram in cyclic operation

The time diagram shows that InOperation is set after a rising edge at Enable and after a couple of cycles. It is changed to status "WAIT_FOR_NEXT_ACTION". A rising edge at StartAction with SelectAction := TEACH_IN results in leaving the status "WAIT_FOR_NEXT_ACTION". Teaching is executed. However, a new rising edge at StartAction with SelectAction := READ_DIAG is ignored. At this point, the function block is not in status "WAIT_FOR_NEXT_ACTION", as the teaching action is already being executed. ActionDone = TRUE is set for one cycle (as the rising edge at StartAction has already been removed) after teaching has been completed successfully. The function block returns to status "WAIT_FOR_NEXT_ACTION".

An executed action is completed successfully if the corresponding acknowledgement ActionDone = TRUE is reported. It is not possible to execute sev-

RMB_TechRegiExtended

eral operations simultaneously. If the rising edge has already been removed from the input, ActionDone is pending for exactly 1 cycle (see the following time diagram).

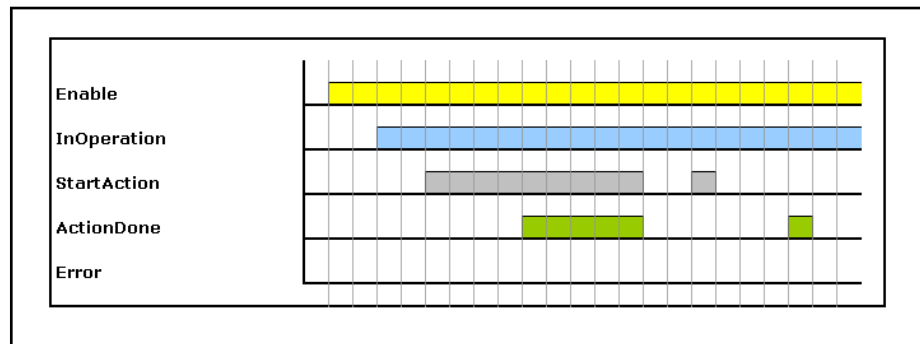


Fig. 3-36: Time diagram of ActionDone behavior

As several actions take some time (e.g. TeachIn), the function block has the output "Status". If Status = 0 and InOperation = TRUE, the function block is in status "WAIT_FOR_NEXT_ACTION". The next action can be executed. The status can assume the values of the "MB_REGI_SENSOR01_COMM_STATUS" enumeration type. If the function block is not InOperation = TRUE, the status is "SENSOR_OFF", except for if the status is in error status. If this is the case, the status is "SENSOR_ERROR".

After starting or stopping the measurement or after starting or stopping the sending of velocity information, the current status is displayed at the MeasureInOp and SendSpeedInOp outputs. Thus, it is continuously displayed that the measurement and the transmission of the velocity information is running.

Prior to executing the "SET_CONFIG", "SEARCH_BARCODE", "SEARCH_MARK_STREAM", "TEACH_IN" and "EASY_TEACH_IN" actions, it is checked if a measurement is running. If this is the case, the measurement is stopped and the action is subsequently executed.

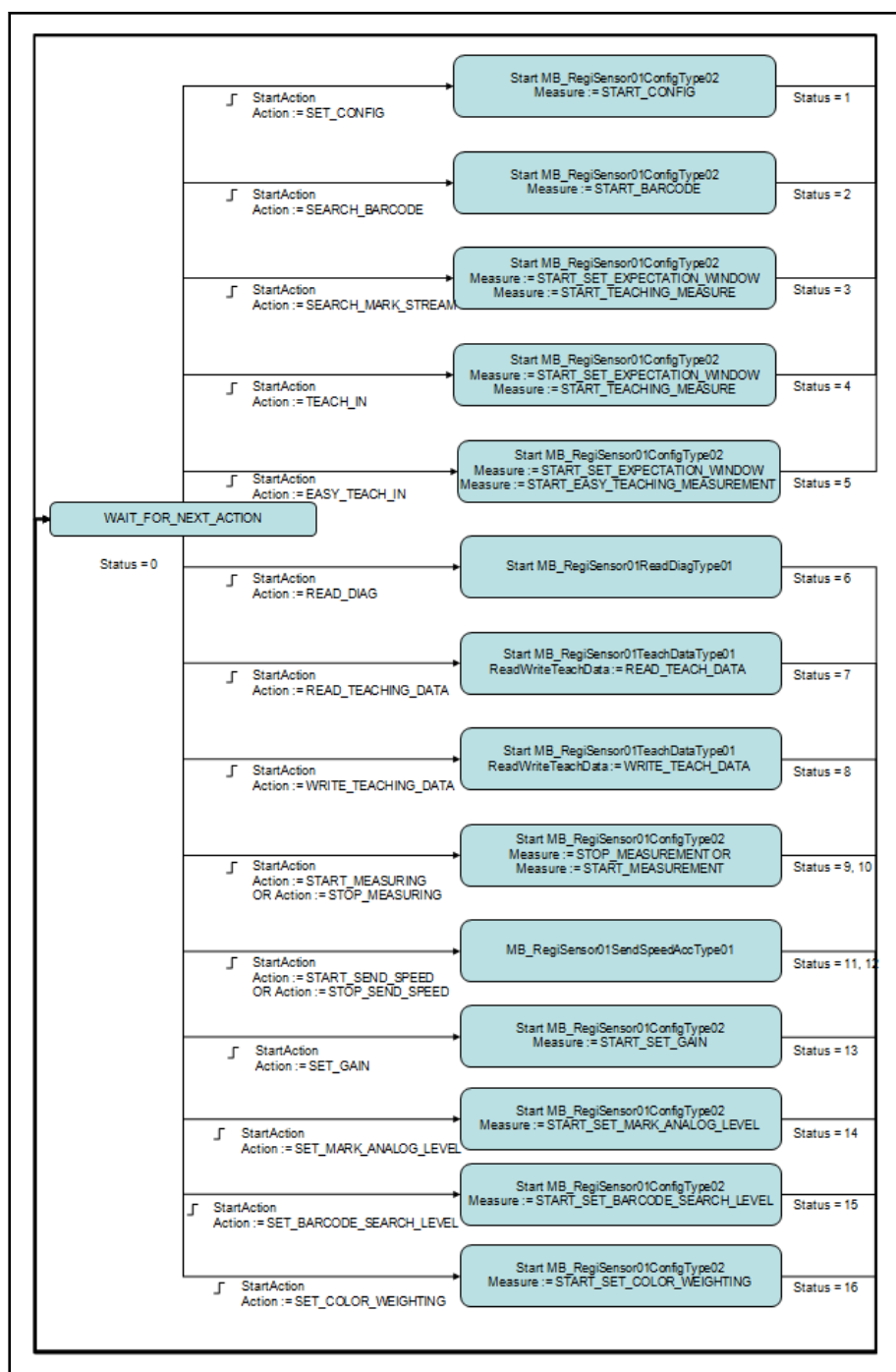


Fig. 3-37: Cyclic operation sequence

The figure shows the sketched sequence of the cyclic operation. It is shown, that it can be jumped to the respective actions starting from the "WAIT_FOR_NEXT_ACTION" status. It is returned to the initial state after the action has been completed. It is also possible to jump to the error status (not shown). Basic inputs for these actions have been marked. Several function blocks can be called in the states. For example, in case of a rising edge at StartAction (SelectAction := TEACH_IN), the MB_RegiSensor01Config-Type02 function block is initially called to set the expectation window and to subsequently call it again for teaching and starting the measurement.

RMB_TechRegiExtended

SEARCH_MARK_STREAM

This function searches for the mark stream without bar code search and without expectation window specification.

The following preconditions have to be fulfilled for this function:

- Only register marks can be found on the 360° rotation that is sampled by the mark stream sensor. This is to say that neither register crosses nor other markings should be on the sampling track of the sensor.
- The register mark stream should take up a maximum of 2/3 of the printing cylinder format. Thus, at least 1/3 of the printing cylinder format has to be an area in which no printing takes place. Example: If 12 register marks are contained on the format to be measured with a distance of 20 mm (mark stream of size 220 mm), the printing cylinder format has to be at least 330 mm.

If one of the specified preconditions is not met, the function cannot be executed or is closed with an error.



The function searches for the mark stream with a maximum of 3 teach actions. Only a valid configuration needs to have been transferred. The expectation window does not need to be specified.

Data/data structures

As not all inputs are required for each action, the following table shows the data applied during the individual actions:

SelectAction	Data	Data changed by the action
SET_CONFIG	Format MarkStructure Bar code MarkStream LengthRegisterMode SideRegisterMode CommunicationConfig Tolerances InternalParameter	ActionDone Status
SEARCH_BARCODE	Barcode.NumberOfBlocks Barcode.FreeSpaceBeforeBarcode Barcode.BarcodeBlocks InternalParameter.Gain	ActionDone Status BarcodeSearchBeginOfExpWindow BarcodeFoundProbability
SEARCH_MARK_STREAM	-	ActionDone Status
TEACH_IN	Barcode.BeginExpectationWindow Barcode.SizeOfExpectationWindow	ActionDone Status RegiSensorData

RMB_TechRegiExtended

SelectAction	Data	Data changed by the action
EASY_TEACH_IN	Barcode.BeginExpecta- tionWindow Barcode.SizeOfExpecta- tionWindow InternalParameter.Gain	ActionDone Status RegiSensorData
READ_DIAG	-	ActionDone Status DiagData
READ_TEACHING_DATA	-	ActionDone Status TeachData
WRITE_TEACHING_DA- TA	TeachData	ActionDone Status
START_MEASURING	-	ActionDone MeasureInOp Status RegiSensorData
STOP_MEASURING	-	ActionDone MeasureInOp Status RegiSensorData
START_SEND_SPEED	TimeSendIntervall CycleTime Master	ActionDone SendSpeedInOp Status
STOP_SEND_SPEED	-	ActionDone SendSpeedInOp Status
SET_GAIN	InternalParameter.Gain	ActionDone Status
SET_MARK_ANALOG_ LEVEL	InternalParameter.MarkA- nalogLevel	ActionDone Status
SET_BAR- CODE_SEARCH_LEVEL	InternalParameter.Barco- deSearchLevel	ActionDone Status
SET_COLOR_WEIGHT- ING	InternalParameter.Color- Weighting	ActionDone Status

Tab. 3-53: Data dependencies

Exemplary sequence

RMB_TechRegiExtended

The exemplary sequence from configuration to measurement is described for fbRegiSensor01Comm instance of the MB_RegiSensor01CommType02: function block in the following figure:

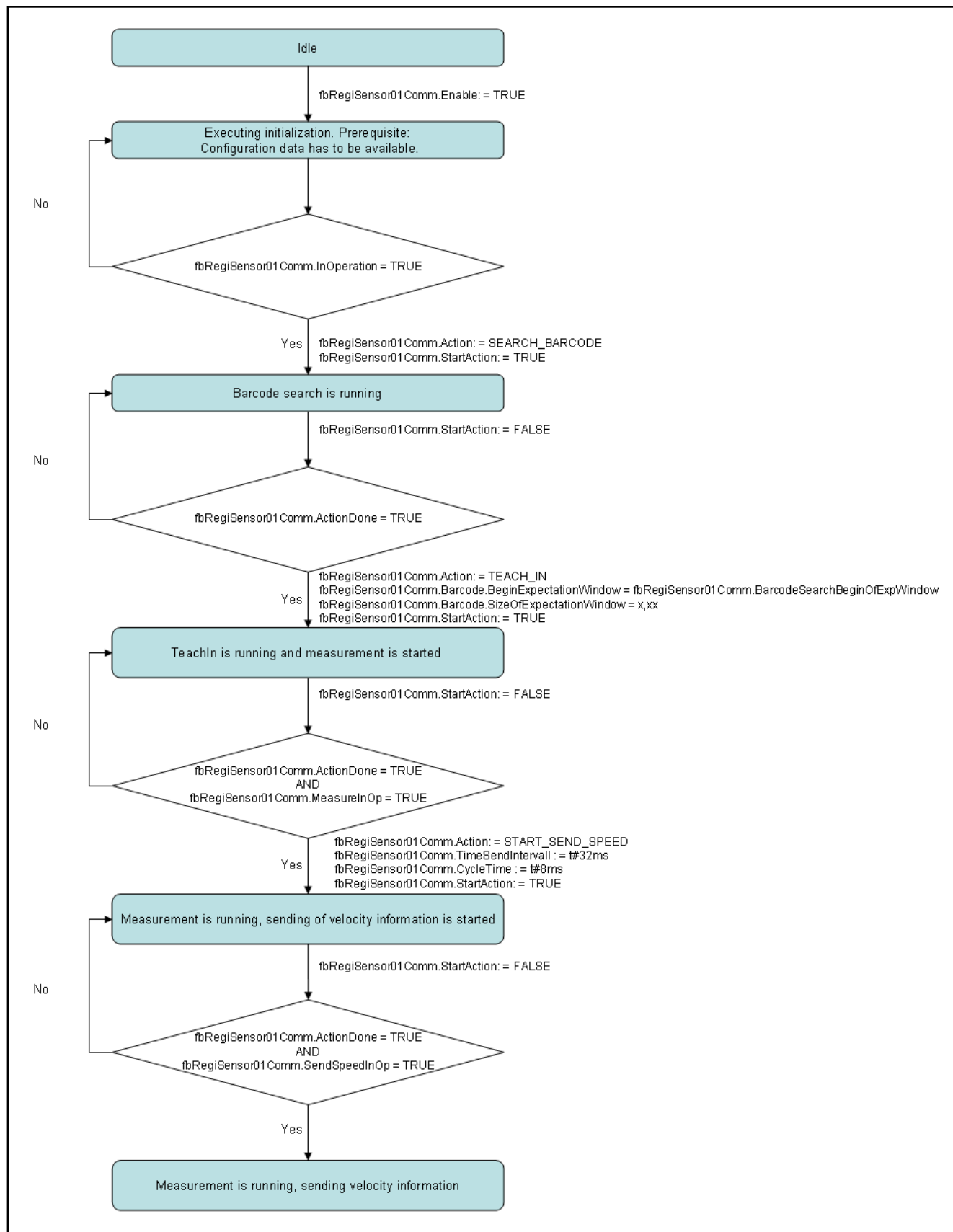


Fig. 3-38: Exemplary sequence from configuration to measurement

Initially, a rising edge is set at enable of the fbRegiSensor01Comm instance. The function block MB_RegiSensor01SendRecvDataType02 establishes the communication connection to the sensor. Subsequently, the configuration is transferred using the MB_RegiSensor01ConfigType02 function block. Pre-condition: valid data have to be used. The MB_RegiSensor01CycComType04 measuring block is started and InOperation becomes "TRUE". Subsequently, the function block is in cyclic mode. The bar code search starts. After the search has been completed, the value has to be assigned as expectation window start and a valid value has to be available for the expectation window size. TeachIn is started. The expectation window is set, the teaching is started to subsequently start the measurement. To achieve a higher accuracy during the acceleration phase, transmitting of the velocity information is enabled using the MB_RegiSensor01SendSpeedAccType01 function block. The measurement is started and the measured data can be found in the RegiSensorData structure.

Characteristics of warning and error

The function block has a second diagnostic interface "MB_REGI_WARNING" apart from the standardized error handling. Warnings can be displayed by means of that interface. The second diagnostic structure is needed due to the connection of several function blocks and the external Rexroth mark stream sensor. Thus, it is differentiated between severe error (Error = TRUE and error status) and error that are reported by an internal function block but which should not reset the entire MB_RegiSensor01CommType02 function block. For example, not all marks can be found during a TeachIn. This is reported as warning as it is no severe error and the user can still execute another action.

A warning is defined as follows:

- Warnings are only generated during cyclic operation.
- A warning is set after an action has not been executed successfully.
- No confirmation by ActionDone is output.
- The function block returns to the "WAIT_FOR_NEXT_ACTION" state.
- The warning is automatically reset once a new action is executed.
- In WarningAdditional1, Additional 1 of the "ERROR_STRUCT" error structure and in WarningAdditional2, Additional 2 of the "ERROR_STRUCT" error structure of the internal function block are pending.

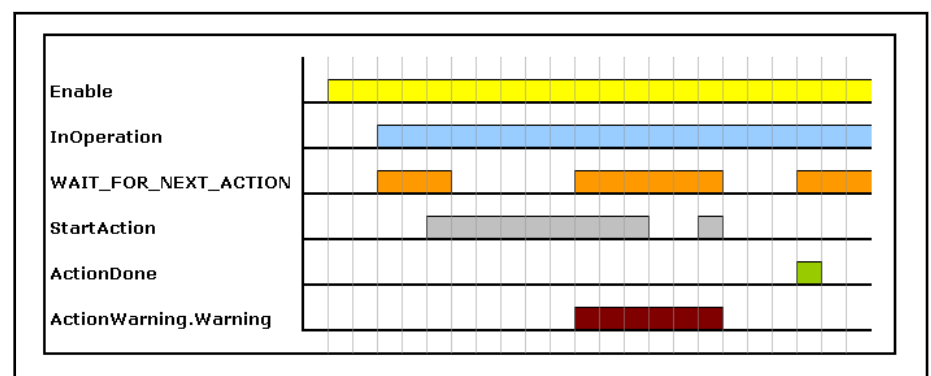


Fig. 3-39: Time response of the warning

The diagram shows the time response when a warning occurs. An action is executed in cyclic operation (InOperation=TRUE). The action is not completed successfully. That is the reason why a warning is reported. The function block returns to the "WAIT_FOR_NEXT_ACTION" state when the warning is reported. The warning is automatically reset after a start of a new action. This action can be completed successfully and is acknowledged with ActionDone.

RMB_TechRegiExtended

Warning and Error during initialization

Each error generated by an internal function block during initialization results in an error of the MB_RegiSensor01CommType02 function block. I.e., a connection has to be re-established to the sensor, a valid configuration has to be transferred and the measuring block has to be started.

Warning and error in cyclic operation

Only connection errors result in an error during cyclic operation as connection errors are severe errors. Severe errors are all errors of the MB_RegiSensor01SendRecvDataType02 and MB_RegiSensor01CycComType04. function blocks.

The following error numbers of other internally used function blocks result in a transition to the error status:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000000	Timeout; no response from the MB_RegiSensor01SendRecvDataType02 function block within 10 seconds
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000005	Timeout while receiving a package.
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000006	Timeout; no response from the MB_RegiSensor01SendRecvDataType02 function block within 10 seconds.

Tab. 3-54: Internal function block errors, resulting in error status

All other internal function block errors are interpreted as warnings. If a warning occurs, Additional1 and Additional2 of "ERROR_STRUCT" of the internal function block is copied to WarningAdditional1 and WarningAdditional2 and the warning bit is set. Internally, the function block in which the error occurs is reset. When executing an action again, the warning is automatically deleted and WarningAdditional1 and WarningAdditional2 are reset to 0.

Error handling

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
OTHER_ERROR (16#7FFE)	16#00001D13	16#00000001	Mark stream search failed
OTHER_ERROR (16#7FFE)	16#00001D13	16#00000002	Measurement has already been started
OTHER_ERROR (16#7FFE)	16#00001D13	16#00000003	Measurement has already been stopped
OTHER_ERROR (16#7FFE)	16#00001D13	16#00000004	Send velocity information has already been started
OTHER_ERROR (16#7FFE)	16#00001D13	16#00000005	Send velocity information has already been stopped
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000003C	Zero pointer format
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000003D	Zero pointer MarkStructure
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000003E	Zero pointer bar code
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000003F	Zero pointer MarkStream

ErrorID	Additional 1	Additional 2	Description
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000040	Zero pointer LengthRegisterMode
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000041	Zero pointer SideRegisterMode
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000042	Zero pointer CommunicationConfig
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000043	Zero pointer Tolerances
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000044	Zero pointer InternalParameter

Tab. 3-55: Error codes of the MB_RegiSensor01CommType02 function block
The function block continues to report error messages of the integrated function blocks:

- [chapter 3.3.8 "MB_RegiSensor01ConfigType02" on page 85](#)
- [chapter 3.3.10 "MB_RegiSensor01CycComType04" on page 106](#)
- [chapter 3.3.11 "MB_RegiSensor01ReadDiagType01" on page 110](#)
- [chapter 3.3.9 "MB_RegiSensor01SendRecvDataType02" on page 102](#)
- [chapter 3.3.12 "MB_RegiSensor01SendSpeedAccType01" on page 112](#)
- [chapter 3.3.13 "MB_RegiSensor01TeachDataType01" on page 117](#)

The corresponding error messages are only generated when the respective function block is enabled. The application of the respective values has to be taken into consideration. Error messages can be generated in case of an active InOperation. Furthermore, these errors, as previously described, can be reported as warnings.

3.3.8 MB_RegiSensor01ConfigType02

Brief description

The MB_RegiSensor01ConfigType02 function block is required for the configuration of the mark stream sensor from sensor firmware 12. The function block prepares the configuration data and transmits them to the MB_RegiSensor01SendRecvDataType02 function block that is responsible for sending the package and receiving the response package. The expectation window can be set apart from the configuration, the teach-in and the measurement can be started or the measurement can be stopped. It executes the initial configuration of the sensor and configures the sensor during the measurement.

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V14	RMB_TechRegiExtended

Tab. 3-56: Reference table of the MB_RegiSensor01ConfigType02 function block

RMB_TechRegiExtended

Interface description

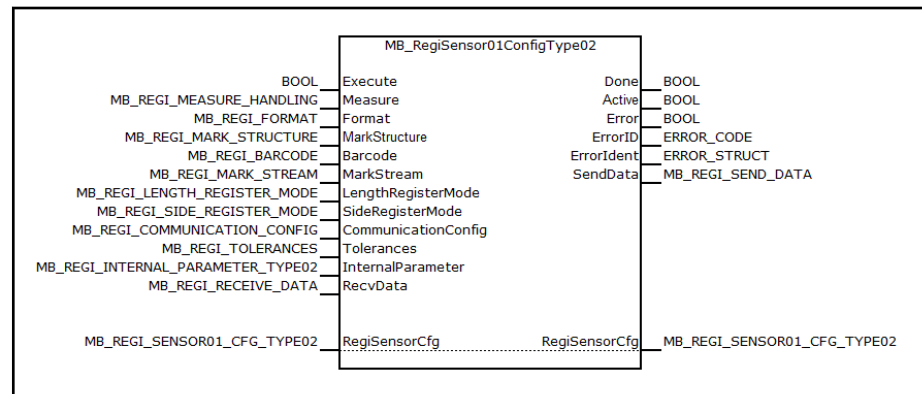


Fig. 3-40: MB_RegiSensor01ConfigType02 function block

I/O type	Name	Data type	Explanation
VAR_IN_OUT	RegiSensorCfg	MB_REGI_SENSOR01_CFG_TYPE02	Contains the current configuration data of the sensor
VAR_INPUT	Execute	BOOL	Processing of the function block enabled (edge-controlled)
	Measure	MB_REGI_MEASURE_HANDLING	Transmits the configuration, sets the expectation window, starts the teach-in and measurement or stops the measurement.
	Format	MB_REGI_FORMAT	Defines the format characteristics
	MarkStructure	MB_REGI_MARK_STRUCTURE	Defines the mark structure
	Bar code	MB_REGI_BARCODE	Defines the bar code
	MarkStream	MB_REGI_MARK_STREAM	Defines the mark stream
	LengthRegister-Mode	MB_REGI_LENGTH_REGISTER_MODE	Defines the web measuring principle for the length register
	SideRegister-Mode	MB_REGI_SIDE_REGISTER_MODE	Defines the measuring principle used for the side register
	Communication-Config	MB_REGI_COMMUNICATION_CONFIG	Defines the communication characteristics of the UDP connection
	Tolerances	MB_REGI_TOLERANCES	Defines the tolerances allowed for teaching and measurement
	InternalParameter	MB_REGI_INTERNAL_PARAMETER_TYPE02	Internal sensor parameters that should only be changed if specifically mentioned.
	RecvData	MB_REGI_RECEIVE_DATA	It receives data from the MB_RegiSensor01SendRecvData02 function block

I/O type	Name	Data type	Explanation
VAR_OUTPUT	Done	BOOL	Data outputs are valid
	Active	BOOL	Function block is working and currently in an interim state
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Error group
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics
	SendData	MB_REGI_SEND_DATA	Data to be sent via the MB_RegiSensor01SendRecvData function block

Tab. 3-57: I/O interface of the MB_RegiSensor01ConfigType02 function block

Data structures

The following data structures are used by the MB_RegiSensor01ConfigType02 function block:

MB_REGI_SENSOR01_CFG_TYPE02		
CurrentConfig	USINT	Current configuration
Format	MB_REGI_FORMAT	Describes the printing cylinder format
MarkStructure	MB_REGI_MARK_STRUCTURE	Describes the mark structure
Bar code	MB_REGI_BARCODE	Describes the bar code
MarkStream	MB_REGI_MARK_STREAM	Describes the mark stream
LengthRegisterMode	MB_REGI_LENGTH_REGISTER_MODE	Length register mode
SideRegisterMode	MB_REGI_SIDE_REGISTER_MODE	Side register mode
CommunicationData	MB_REGI_COMMUNICATION_CONFIG	Communication ports of the UDP connection
Tolerances	MB_REGI_TOLERANCES	Teaching tolerances and measurement tolerances
InternalParameter	MB_REGI_INTERNAL_PARAMETER_TYPE02	Internal sensor parameters
BarcodeSearchBeginOfExpWindow	REAL	Returns the beginning of the expectation window of the bar code search
BarcodeFoundProbability	REAL	Returns the probability of the bar code from the bar code search

Tab. 3-58: MB_REGI_SENSOR01_CFG_TYPE02 data structure

MB_REGI_FORMAT		
FormatLength	REAL	Printing cylinder format in [mm]
Increments	UDINT	Number of increments per revolution

Tab. 3-59: MB_REGI_FORMAT data structure

MB_REGI_MARK_STRUCTURE		
MarkType	UDINT	Mark type
SingleMark	ARRAY [0...3] OF REAL	Describes the printed mark

Tab. 3-60: MB_REGI_MARK_STRUCTURE data structure

RMB_TechRegiExtended

MB_REGI_BARCODE		
NumberOfBlocks	UDINT	Number of bar code blocks (0 means that no bar code exists)
BeginExpectationWindow	REAL	Beginning of the expectation window (distance from the beginning of the expectation window to the zero pulse) [mm]
SizeOfExpectationWindow	REAL	Size of the expectation window [mm]
FreeSpaceBeforeBarcode	REAL	Printing-free area in front of the bar code [mm]
BarcodeBlocks	ARRAY [0...19] OF REAL	Bar code description in [mm]

Tab. 3-61: MB_REGI_BARCODE data structure

MB_REGI_MARK_STREAM		
NumberOfMarks	UINT	Number of marks
MarkStream	ARRAY [1...12] OF REAL	Distance of the marks [mm]
MarkDirection	BOOL	First or second mark edge for the length register

Tab. 3-62: MB_REGI_MARK_STREAM data structure

MB_REGI_LENGTH_REGISTER_MODE		
ModeSelect	UDINT	Selected length register mode
LengthProperties	ARRAY [0...4] OF REAL	Characteristics for the length register mode [mm]

Tab. 3-63: MB_REGI_LENGTH_REGISTER_MODE data structure

MB_REGI_SIDE_REGISTER_MODE		
ModeSelect	UDINT	Selected side register mode
MarkSelect	ARRAY [0...1] OF UINT	Marks for the side register

Tab. 3-64: MB_REGI_SIDE_REGISTER_MODE data structure

MB_REGI_COMMUNICATION_CONFIG		
IpSensor	STRING	IP address of the sensor
PortConfiguration	UINT	Port for acyclic communication (configuration, diagnostics, velocity data)
IpPLC	STRING	IP address of the PLC
PortMeasurement	UINT	Port for the measurement results

Tab. 3-65: MB_REGI_COMMUNICATION_CONFIG data structure

MB_REGI_TOLERANCES		
TeachZeroPulseToMark	REAL	Teaching tolerance of the first mark to the zero pulse [mm]
TeachMarkToMark	REAL	Teaching tolerance between the individual marks [mm]
MeasureMarkTolerance	REAL	Measurement tolerance compared to the next measured value [mm]
DeviationOfMarkWidth	REAL	Deviation of the mark width to the last measured value [%]
DeviationOfMarkHight	REAL	Deviation of the mark height to the last measured value [%]
DeviationOfLastPosition	REAL	Absolute deviation of the measurement tolerance from the last measured value [mm]
ShiftSyncPulse	REAL	The zero pulse is exactly shifted in front of the expectation window by ShiftSync-Pulse [mm]

Tab. 3-66: MB_REGI_TOLERANCES data structure

MB_REGI_INTERNAL_PARAMETER_TYPE02		
ColorWeighting	REAL	Color weighting of the sensor. Subsequently, the intensity weighting is automatically calculated from (1-ColorWeighting)
Gain	UINT	Sensor gain
MarkAnalogLevel	UINT	The Analoglevel of the mark has to be above this threshold during the TeachIn in order to be detected as valid mark.
BarcodeSearchLevel	REAL	Bar code probability threshold. If the bar code probability is above this threshold, the bar code was found successfully. Otherwise, the bar code scan fails.

Tab. 3-67: MB_REGI_INTERNAL_PARAMETER_TYPE02 data structure

MB_REGI_SEND_DATA		
NoOfBytes	UDINT	Number of bytes to be sent
ValueAddr	POINTER TO DWORD	Pointer to data
NewData	UINT	Indicates new data for sending
SensorPort	UINT	Sensor port to which the data should be sent

Tab. 3-68: MB_REGI_SEND_DATA data structure

MB_REGI_RECEIVE_DATA		
NoOfBytes	UDINT	Number of received bytes
ValueAddr	POINTER TO DWORD	Pointer to data
NewData	UINT	Indicates newly received data

Tab. 3-69: MB_REGI_RECEIVE_DATA data structure

Enumeration type

The enumeration type MB_REGI_MEASURE_HANDLING is used for configuration, measurement start, teach-in start as well as for setting the expectation window and can assume the following values:

Element	Value	Description
START_CONFIG	0	Starts the configuration
START_TEACHING	1	Starts the teaching

RMB_TechRegiExtended

Element	Value	Description
START_MEASUREMENT	2	Starts the measurement
STOP_MEASUREMENT	3	Stops the measurement
START_CONFIG_TEACHING	4	Starts the configuration and the teaching
START_TEACHING_MEASUREMENT	5	Starts the teaching and the measurement
START_CONFIG_TEACHING_MEASUREMENT	6	Starts the configuration, the teaching and the measurement
START_SET_EXPECTATION_WINDOW	7	Sets the expectation window
START_EASY_TEACHING	8	Starts an easy teach-in. The gain as well as the beginning and the size of the expectation window have to be transferred.
START_EASY_TEACHING_MEASUREMENT	9	Starts the each teach-in and then the measurement (cf. 8)
START_BARCODE	10	Starts the bar code search
START_SET_GAIN	11	Starts setting of gain
START_SET_MARK_ANALOG_LEVEL	12	Starts setting of the mark analog level
START_SET_BARCODE_SEARCH_LEVEL	13	Starts setting of the bar code search level
START_SET_COLOR_WEIGHTING	14	Starts setting of color weighting
START_SET_DATE	15	Starts setting of the date

Tab. 3-70: Enumeration type MB_REGI_MEASURE_HANDLING

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
MB_REGI_SENSOR01_CFG_TYPE02					
CurrentConfig	USINT	0	255	0	Rising edge at "Execute"
MB_REGI_FORMAT					
FormatLength	REAL	> 0	n. def	0	Rising edge at "Execute"
Increments	UDINT	> 0	n. def	0	Rising edge at "Execute"
MB_REGI_MARK_STRUCTURE					
MarkType	UDINT	1	7	1 (the marks 1,2,3,5,6,7 are implemented)	Rising edge at "Execute"
SingleMark	ARRAY [0...3] OF REAL	> 0 (for required entries)	n. def	0	Rising edge at "Execute"
MB_REGI_BARCODE					
NumberOfBlocks	UDINT	0 (1 and 2 blocks are invalid entries)	10	0	Rising edge at "Execute"
BeginExpectationWindow	REAL	0	n. def	0	Rising edge at "Execute"
SizeOfExpectationWindow	REAL	0	n. def	1	Rising edge at "Execute"

RMB_TechRegiExtended

Name	Type	Min. value	Max. value	Default value	Effective
FreeSpaceBeforeBarcode	REAL	0	n. def	0	Rising edge at "Execute"
BarcodeBlocks	ARRAY [0...19] OF REAL	> 0 (for required entries)	n. def		Rising edge at "Execute"
MB_REGI_MARK_STREAM					
NumberOfMarks	UDINT	1	12	1	Rising edge at "Execute"
MarkStream	Array [1...12] OF REAL	> 0 (for required entries)	n. def	0	Rising edge at "Execute"
MarkDirection	BOOL			FALSE	Rising edge at "Execute"
MB_REGI_LENGTH_REGISTER_MODE					
ModeSelect	UDINT	1	1	1	Rising edge at "Execute"
LengthProperties	ARRAY [0...4] OF REAL	n. def	n. def	0	Rising edge at "Execute"
MB_REGI_SIDE_REGISTER_MODE					
ModeSelect	UDINT	0	1	1	Rising edge at "Execute"
SelectMarks	ARRAY [0...1] OF UINT	n. def	n. def	0	Rising edge at "Execute"
MB_REGI_COMMUNICATION_CONFIG					
IpSensor	STRING	LEN > 0	n. def	""	Rising edge at "Execute"
PortConfiguration	UINT	1	n. def	0	Rising edge at "Execute"
IpPLC	STRING	LEN > 0	n. def	""	Rising edge at "Execute"
PortMeasurement	UINT	1	n. def	0	Rising edge at "Execute"
MB_REGI_TOLERANCES					
TeachZeroPulseToMark	REAL	1	n. def	5	Rising edge at "Execute"
TeachMarkToMark	REAL	1	n. def	2	Rising edge at "Execute"
MeasureMarkTolerance	REAL	1	n. def	10	Rising edge at "Execute"
DeviationOfMarkWidth	REAL	1	100	20	Rising edge at "Execute"
DeviationOfMarkHeight	REAL	1	100	20	Rising edge at "Execute"
DeviationOfLastPosition	REAL	1	n. def	10	Rising edge at "Execute"
ShiftSyncPulse	REAL	1	n. def	5	Rising edge at "Execute"
MB_REGI_INTERNAL_PARAMETER_TYPE02					
ColorWeighting	REAL	0	< 1	0.5	Rising edge at "Execute"

RMB_TechRegiExtended

Name	Type	Min. value	Max. value	Default value	Effective
Gain	UINT	0	7	0	Rising edge at "Execute"
MarkAnalogLevel	UINT	0	n. def	100	Rising edge at "Execute"
BarcodeSearchLevel	REAL	0	<100	50.0	Rising edge at "Execute"
MB_REGI_RECEIVE_DATA					
NoOfBytes	UDINT	0	n. def	0	Change at "NewData"
ValueAddr	POINTER TO BYTE	n. def	n. def	0	Change at "NewData"
NewData	UINT	0	65535	0	Continuous

Tab. 3-71: Overview on min./max. values and initial values of the inputs



Valid entries are currently only functionalities implemented in the sensor.

Function description

In case of the acyclic communication, the sensor is the client which means the control sends data to the sensor and the sensor responds. The acyclic communication is used to configure the sensor.

The RegiSensorCfg structure contains the current configuration of the sensor. If configuration changes should be made, the changed configuration data can be directly applied at the function block. They are only applied at a rising edge at "Execute". The individual structures are described in the following:

RegiSensorCfg: It includes the current configuration data of the sensor and the current configuration ID transmitted to the cyclic measured data. Additionally, the determined beginning of the expectation window is specified after a bar code search.

Measure: *At a rising edge at "Execute" it is distinguished between 11 processes executed according to the function selected:*

- **START_CONFIG:**

The new configuration is transmitted to the sensor. The expectation window is calculated via mark distances, tolerances and the beginning of the expectation window by the sensor.



Therefore, exact entries are required. It is recommended to determine the beginning and the size of the expectation window via a graphical user interface (HMI) and to transfer these values with the START_SET_EXPECTATION_WINDOW option or to use the bar code search.

- **START_TEACHING:**

Starts the sensor teaching. Therefore, the configuration data sent last to the sensor and the expectation window are applied and the marks are determined with their threshold values in the expectation window. In this mode, the sensor additionally determines the ideal gain.

- **START_MEASUREMENT:**

Starts a measurement without transmitting a configuration or starting a teach-in.

- **STOP_MEASUREMENT:**

Completes a running measurement.

- **START_CONFIG_TEACHING:**

First, it transfers the configuration and subsequently starts a teach-in. If a number higher than 0 was specified in `SizeOfExpectationWindow`, this value is used for the calculation of the expectation window. If the value is 0, the expectation window is computed via mark distances, the tolerances and the beginning of the expectation window by the sensor.

- **START_TEACHING_MEASUREMENT:**

Starts the sensor teaching and subsequently the measurement. The configuration data transmitted last to the sensor and the expectation window are used for teaching.

- **START_CONFIG_TEACHING_MEASUREMENT:**

First, it transfers the configuration and subsequently starts a teach-in and then the measurement. If a number higher than 0 was specified in `SizeOfExpectationWindow`, this value is used for the calculation of the expectation window. If the value is 0, the expectation window is computed via mark distances, the tolerances and the beginning of the expectation window by the sensor.

- **START_SET_EXPECTATION_WINDOW:**

Sets only the expectation window that was for example transferred from a graphical interface. Therefore, a value higher than 0 has to be specified in `SizeOfExpectationWindow` as well as the beginning of the expectation window (`BeginExpectationWindow`).



If a new configuration is started after transferring the expectation window, the transferred expectation window is overwritten. Thus, it is recommended to transfer the configuration first and then the expectation window.

- **START_EASY_TEACHING:**

A teach-in is started without determining the gain. The gain located under "InternalParameters" in "Gain" is applied. This gain was ideally read out with the function block `MB_RegiSensor01ReadDiagType01` after a first normal teach-in. Furthermore, the size and the beginning of the expectation window have to be specified.

- **START_EASY_TEACHING_MEASUREMENT:**

As `START_EASY_TEACHING`, but the measurement is immediately started after teaching.

- **START_BARCODE:**

If this mode is executed, the bar code search is started. Therefore, 3 to 10 bar codes have to be defined. After the bar code search has been completed successfully, the position is right behind the bar code that is the beginning of the expectation window in `RegiSensorCfg.BarcodeSearchBeginOfExpWindow`. This value can be used to define the expectation window.

- **START_SET_GAIN**

RMB_TechRegiExtended

If this mode is executed, the gain can be set. The gain can thus be transmitted individually without having to transmit an entire configuration.

- **START_SET_MARK_ANALOG_LEVEL**

If this mode is executed, the mark analog level can be set. The parameter can thus be transmitted individually without having to transmit an entire configuration.

- **START_SET_BARCODE_SEARCH_LEVEL**

If this mode is executed, the bar code search level can be set. The parameter can thus be transmitted individually without having to transmit an entire configuration.

- **START_SET_COLOR_WEIGHTING**

If this mode is executed, the color weighting can be set. The parameter can thus be transmitted individually without having to transmit an entire configuration.

- **START_SET_DATE**

If this mode is executed, the date can be set. The parameter can thus be transmitted individually without having to transmit an entire configuration. The date is a basic parameter of the sensor and has to have been transmitted to the sensor prior to connecting the ActiveXControls.

The following figure shows an exemplary sequence from the configuration of the mark stream sensor to the measurement.

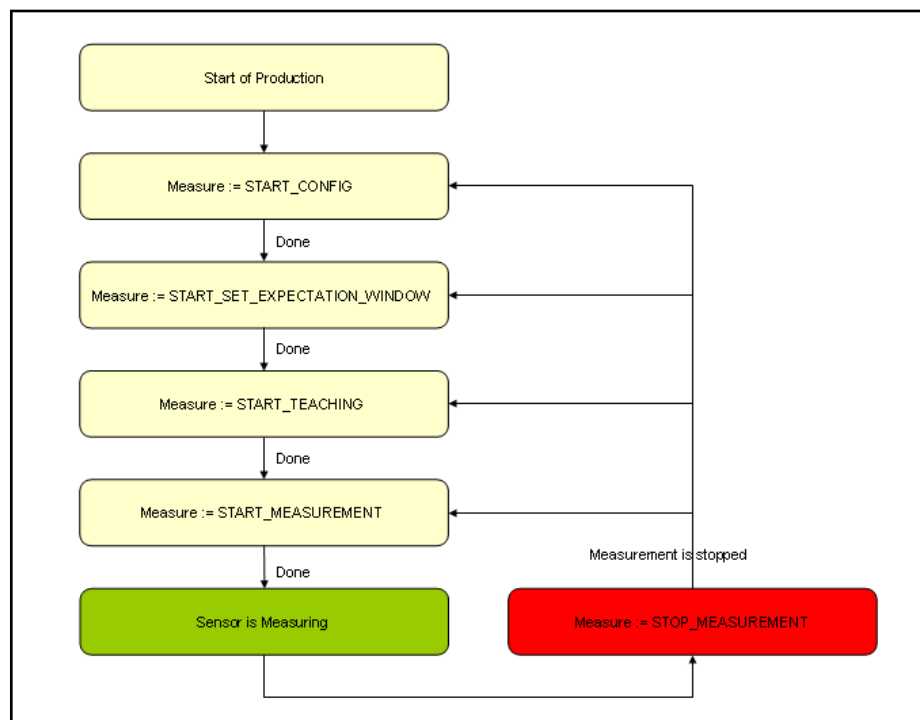


Fig. 3-41: Sequence from the configuration to the measurement

The sensor has to be configured first. Subsequently, the expectation window should be preferably set via a visualization or the thus determined values. Subsequently, teaching and measurement can be started. After the measurement has been started, it can be stopped any time and continued at the sensor state desired. The "Measure" functionalities can be called in any combination.

MarkStructure: Contains the predefined mark structure and the dimension of the marks.

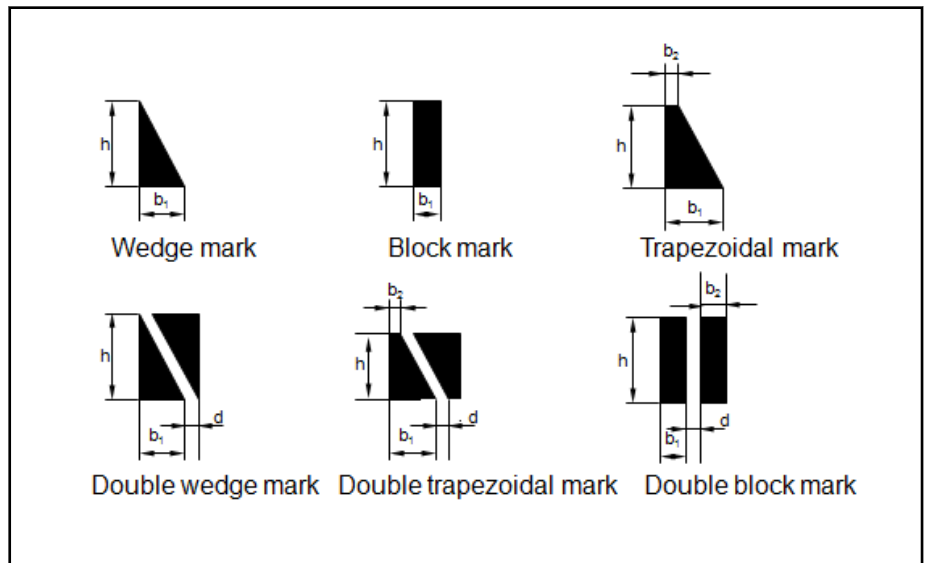


Fig. 3-42: Supported mark types

The "MarkType" parameter can be set as follows and the parameters have to be subsequently specified in "SingleMark" according to the mark type:

	MarkType	SingleMark			
Wedge mark	1	Width b_1 [mm]	Height h [mm]		
Double-wedge marks	2	Width b_1 [mm]	Height h [mm]	Distance d [mm]	
Block mark	3	Width b_1 [mm]	Height h [mm]		
Trapezoidal mark	5	Width b_1 [mm]	Height h [mm]	Width b_2 [mm]	
Double-trapezoidal mark	6	Width b_1 [mm]	Height h [mm]	Distance d [mm]	Width b_2 [mm]
Double-block mark	7	Width b_1 [mm]	Height h [mm]	Distance d [mm]	Width b_2 [mm]

Tab. 3-72: Mark type parameters

The following shows a configuration example of a triangular mark:

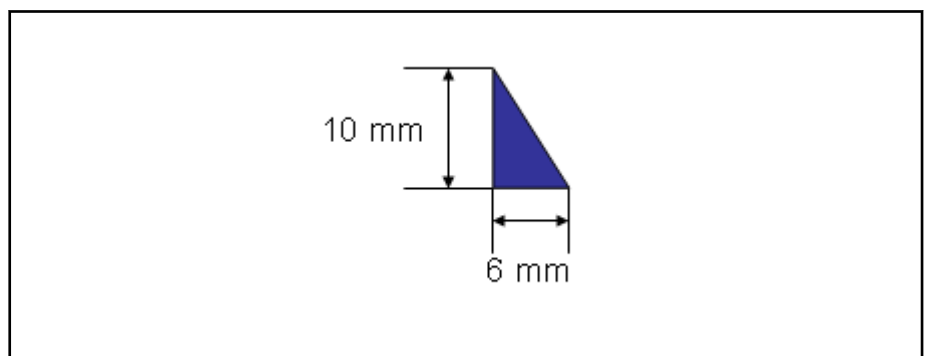


Fig. 3-43: Mark structure example

Code example:

```
MarkStruct: MB_REGI_MARK_STRUCTURE := (MarkType:=1, SingleMark:= 6, 10, 0, 0);
```

This is a wedge mark (mark type 1). This is defined by its height and width.

RMB_TechRegiExtended

Bar code: Contains the number of bar code blocks and the description of the expectation window. Up to ten bar code blocks are possible. If NumberOfBlocks = 0, there is no bar code available. The user has to set the beginning and the size of the expectation window. This can be carried out either by determining this data via the oscilloscope representation or by calculating the size of the expectation window by the sensor (from the number of marks and their distance as well as the tolerances and the beginning of the expectation window). Therefore, Barcode.SizeOfExpectationWindow has to be set to 0.

If a bar code is available, the sensor can determine the beginning of the expectation window by returning the position of the bar code end. Therefore, START_BARCODE has to be started. If the function block returns "Done", the position of the bar code end is written to RegiSensorCfg.BarcodeSearchBeginOfExpWindow. The user has to specify the size of the expectation window and can then set the expectation window with START_SET_EXPECTATION_WINDOW. The normal procedure with "teach-in" and "start measurement" can now be continued.

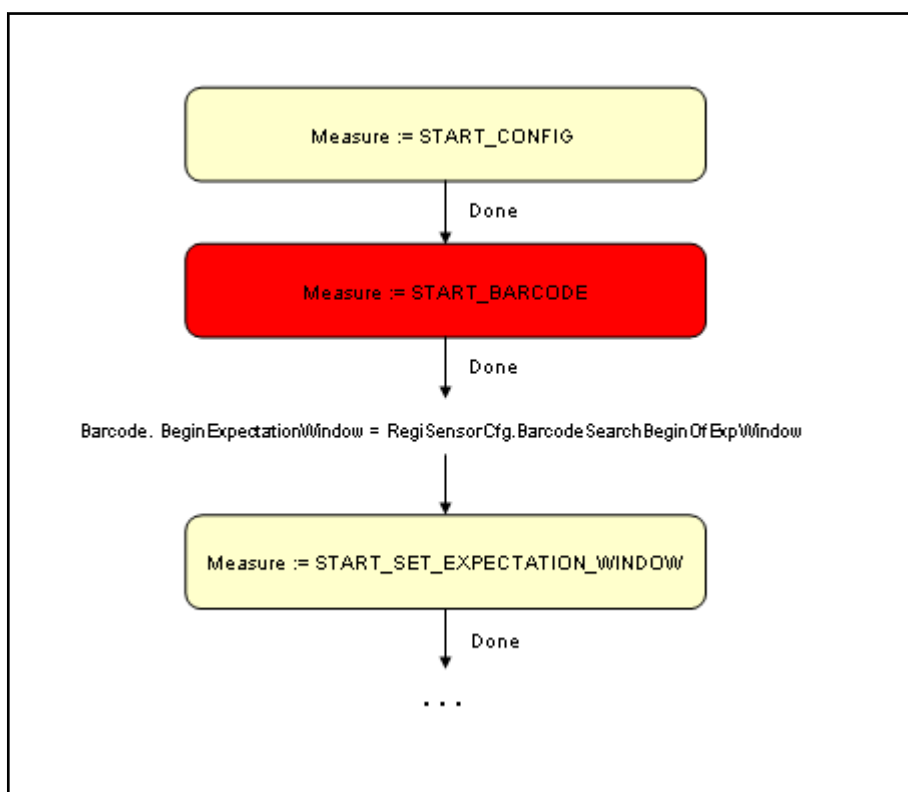


Fig. 3-44: Sequence of the bar code search

If a bar code is available, the sensor searches for the beginning of the expectation window. "FreeSpaceBeforeBarcode" indicates the printing-free area in front of the bar code and "BarcodeBlocks" defines the individual bar code blocks. The dimensions of the bar code still have to be specified as follows:

Bar code example:

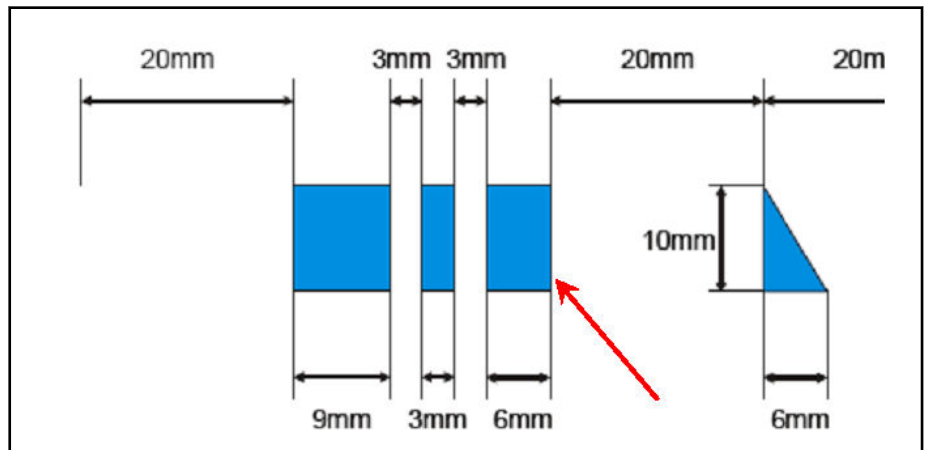


Fig. 3-45: Bar code example

The dimensions are given in the array "BarcodeBlocks" in [mm]:

BarcodeBlocks[0]: Width of bar code block 1 in [mm]

BarcodeBlocks[1]: Distance from bar code block 1 to block 2 in [mm]

BarcodeBlocks[2]: Width of bar code block 2 in [mm]

BarcodeBlocks[3]: Distance from bar code block 2 to block 3 in [mm]

.....

The last distance specified is the distance between the last bar code block and the first mark (in this example, the distance between the bar code block and the registration mark is 20 mm). The position is indicated with the red arrow that is returned by the sensor in RegiSensorCfg.BarcodeSearchBeginOfExpWindow after a successful bar code search. That is the direct position behind the bar code.

MarkStream: Contains the number of marks and the respective distance between the marks. If five marks are available, there are four mark distances and the first defined distance is the distance between mark 1 and mark 2.

Mark stream example:

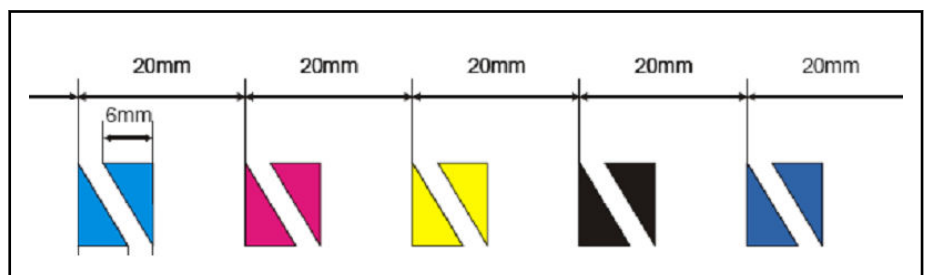


Fig. 3-46: Mark stream example

Example of the shown mark stream:

```
MarkStream: MB_REGI_MARK_STREAM := (NumberOfMarks:=5,
MarkStream:=20,20,20,20);
```

The mark stream consists of five marks. These have a distance of 20 mm each.

Furthermore, it can be specified - when defining the mark stream - whether the vertical mark edge is at first or second place.

RMB_TechRegiExtended

If MarkDirection = FALSE, the length register value is returned with regard to respective first mark edge. That is the distance between the zero pulse and the first mark edge of the mark.

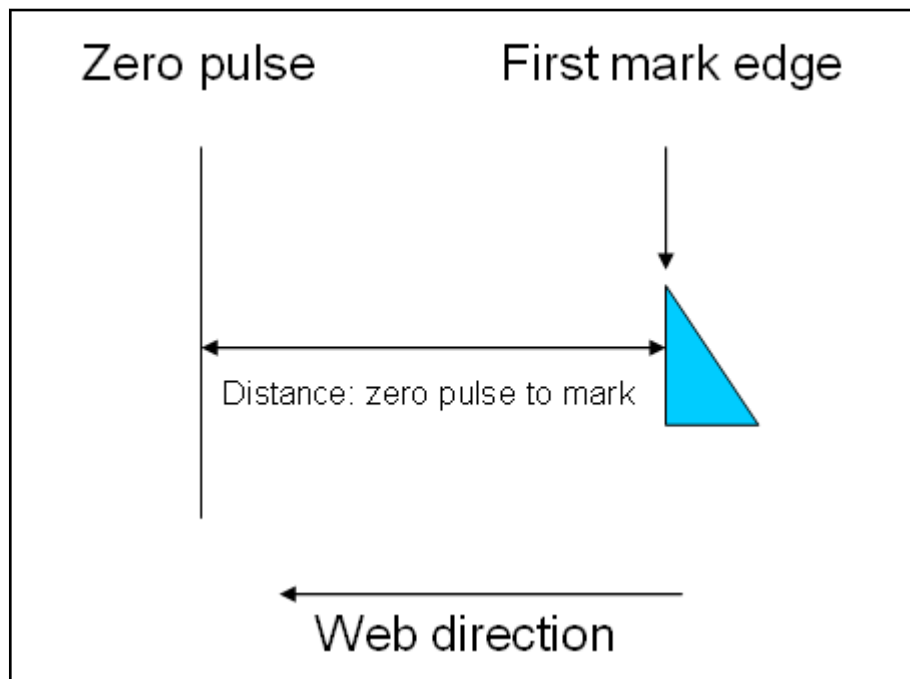


Fig. 3-47: First mark edge

If MarkDirection = TRUE, the length register value is returned with regard to respective second mark edge.

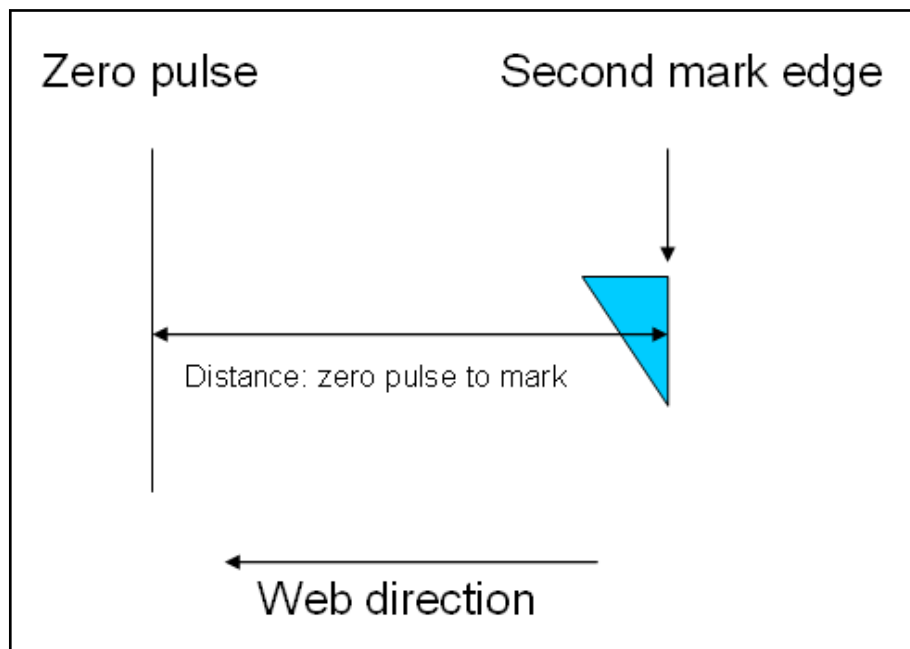


Fig. 3-48: Second mark edge



It is recommended to **always** use the vertical edge for the length register control. If the inclined mark edge is used, an adjustment in the side register always results in an offset in the length register.

LengthRegisterMode: Indicates the respective web measuring principle for the length register. Furthermore, all required dimensions are saved for the respective web measuring principle. Currently, only Mode 1 of the return of all mark distances is implemented in the mark stream sensor.

Example:

```
LengthRegisterMode: MB_REGI_LENGTH_REGISTER_MODE := (ModeSelect:=1);
```

ModeSelect := 1 means that all mark distances of all marks are returned.

SideRegisterMode: Indicates which mark widths of which marks should be measured for the side register. If ModeSelect := 0, no mark widths are returned. If ModeSelect := 1, all mark widths are returned.

Example 1:

```
SideRegisterMode: MB_REGI_SIDE_REGISTER_MODE := (ModeSelect:=0);
```

No mark widths are returned.

Example 2:

```
SideRegisterMode: MB_REGI_SIDE_REGISTER_MODE := (ModeSelect:=1);
```

All mark widths are returned.

Currently, there is no function available that can specify two marks whose mark widths should be output.

CommunicationConfig: Contains the communication settings for the acyclic and the cyclic communication. The port on the PLC for the measured values and the port for the configuration data and diagnostic data as well as the IP address of the sensor and the PLC are specified.

Example:

```
CommunicationCfg: MB_REGI_COMMUNICATION_CONFIG := (IpSensor:='192.168.2.5', PortConfiguration:=6001, IpPLC:='192.168.2.3', PortMeasurement:=6000);
```

A configuration of the connection data is shown in the example above. The sensor has the IP address 192.168.2.5 and the acyclic communication is executed via the UDP port 6001 of the control. The IP address of the PLC is 192.168.2.3. Thus, the sensor knows that the cyclic measurement results should be sent to this IP address 192.168.2.3 and the UDP port 6000 of the control.

Tolerances:



For the tolerances, the respective default values should be used and only changed in an exceptional case.

Tolerances indicate tolerances for teaching, measuring and the robustness of the measurement:

- **TeachZeroPulseToMark:**
This tolerance is not taken into consideration anymore. Always use default values.
- **TeachMarkToMark:**
Equal this tolerance to the MeasureMarkTolerance.
- **MeasureMarkTolerance:**
During a measurement, an individual mark expectation window is used for each mark. That means that it is searched for the mark exactly in this

RMB_TechRegiExtended

expectation window due to its last position. If the position of the mark kept changing after the last revolution, the mark is not found.

The following three parameters are used for the robustness of the measurement. That means that a mark was detected with a certain width, height and position during the last measurement. If the mark found deviates from the absolute position during the next measurement by the respective percentage, this measurement is ignored and it is searched again with the measured data last valid. This should ensure that a mark can be found during next measurement in case of an disturbance occurring once (e.g. due to a crease).

- DeviationOfMarkWidth:

Maximum deviation of the mark width from the last measurement allowed in percent. If this value is set to 100 %, the monitoring of the mark width is switched off.

- DeviationOfMarkHeight:

Maximum permissible deviation of the mark height (analog value) in percent from the mark height (analog value) of the teach result. If this value is set to 100 %, the monitoring of the mark height is switched off.

- DeviationOfLastPosition:

Maximum absolute deviation of the mark position from the last measurement allowed.

- ShiftSyncPulse:

Indicates the distance between the shifted zero pulse and the beginning of the expectation window. To ensure a high measurement accuracy during acceleration phase, the zero pulse is shifted up to the beginning of the expectation window with regard to the hardware. This value indicates the tolerance between the shifted zero pulse and the beginning of the expectation window. If it is to be expected that the mark stream does not keep moving forward, a default value of 5 mm is recommended.

InternalParameter: This data structure includes internal parameters of the sensor that should only be changed if instructed by Bosch Rexroth. Color and intensity can be scaled here. The default values are values from experience. The structure also contains the gain transmitted during easy-teach-in.

- ColorWeighting:

Indicates the color weighting in the analog signal. Subsequently, the intensity weighting is automatically calculated from $(1 - \text{ColorWeighting})$. This means if the sensor is intended as mere color sensor, set parameter ColorWeighting to 0.99.

- Gain:

The sensor gain can manually be set using 0-7. The sensor automatically determines the best gain value during a TeachIn. Thus, only use manually setting if the oscilloscope data are not displayed correctly upon the first display of the 360° data.

- MarkAnalogLevel:

The MarkAnalogLevel specifies the threshold from which a register mark is detected as register mark during TeachIn. Thus, finding a register mark in noise is to be avoided. If the MarkAnalogLevel is set to 500, the mean value of the signal of the register mark has to have a mean analog value of > 500 during the teaching operation. In this case, the register mark is detected, otherwise the register mark is ignored. Furthermore, the MarkAnalogLevel also refers to the bar code. This means, on

average, the defined bar code block have to exceed the MarkAnalogLevel.

- BarcodeSearchLevel:

The BarcodeSearchLevel specifies from which probability a bar code is detected. A defined pattern is used for the bar code search. The end result is the bar code position as well as a probability with which the bar code matches the defined pattern. If the probability is lower than the BarcodeSearchLevel, the bar code is not detected. The parameter BarcodeFoundProbability subsequently specifies the probability of the found bar code. Consequently, this parameter can be used to specify a new bar code search threshold.

Sending and receiving data packages

The function block indicates with the incrementation from "NewData" to "SendData" that it wants to send new data. Sending is carried out by the MB_RegiSensor01SendRecvDataType02 function block. If it receives new data, it sets the "NewData" value equal to the "RecvData" value. Thus, the function block knows that it is the respected response package. A response package is expected with each transmitted data package.

Timeout:

If a new data package is sent, an internal timer runs in the function block. If it does not get a response within 10 s, it aborts with an error due to timeout.

A possible error case is that the MB_RegiSensor01SendRecvDataType02 function block is not activated or incorrectly connected.

Error handling

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000000	Timeout; no response from the MB_RegiSensor01SendRecvDataType02 function block within 10 seconds
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000001	Timeout while teaching. Teaching could not be completed in the time specified.
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000000	Invalid plate cylinder format
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000001	Invalid mark type
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000002	Invalid mark geometry
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000003	Invalid number of bar code blocks
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000004	Invalid number of marks in the mark stream
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000005	Invalid mark distance
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000006	Invalid length register mode
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000007	Invalid length register parameter
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000008	Invalid side register mode
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000009	Invalid sensor IP
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000000A	Invalid control IP
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000000B	Invalid UDP port for the measurement

RMB_TechRegiExtended

ErrorID	Additional 1	Additional 2	Description
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000000C	Invalid configuration port
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000000D	Invalid number of increments
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000000E	Invalid measurement tolerance
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000000F	Invalid teaching tolerance of a mark
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000010	Invalid teaching tolerance, beginning of expectation window
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000011	Invalid beginning of expectation window
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000012	Invalid size of expectation window
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000013	Invalid tolerance for mark width
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000014	Invalid tolerance for mark height
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000015	Invalid tolerance for mark position
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000016	Invalid "ShiftSyncPulse" tolerance
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000017	Invalid color weighting
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000018	Invalid intensity weighting
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000019	Invalid "Measurement" mode
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000001A	Invalid gain
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000003B	Invalid free space in front of the bar code
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000004D	Invalid BarcodeSearchLevel
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000004E	Invalid MarkAnalogLevel
COMMUNICATION_ERROR (16#0002)	16#0C001D10	16#00000002	Timeout during bar code search
OTHER_ERROR (16#7FFE)	16#00001D12	16#xxxxxxx	Error handling for the mark stream sensor (see table "Mark stream sensor") is not integrated in "F_RELATED_TABLE"

Tab. 3-73: Error codes of the MB_RegiSensor01ConfigType02 function block

Additionally, the function block can output errors of the mark stream sensor.

3.3.9 MB_RegiSensor01SendRecvDataType02

Brief description

The MB_RegiSensor01SendRecvDataType02 function block executes the communication with the sensor if a sensor firmware ≥ 12 is used. Therefore, the function blocks transfer their data to be sent and get the respective data packages in return for the following: sensor configuration, sending velocity information and acceleration information. The diagnostic function block sends its data and gets the respective data packages in return during an acceleration phase. The function block prevents a collision of function blocks accessing the same UDP port simultaneously. Prioritization is still implemented. Sending acceleration information is time-critical. Due to this fact, out of the configuration data of the MB_RegiSensor01SendSpeedAccType01 function block, diagnostic data of the MB_RegiSensor01ConfigType01 function block, teach data of the MB_RegiSensor01ReadDiagType01 function block and data packages of the MB_RegiSensor01ConfigTypeType01 function block, the preference is given to the data packages and they are immediately sent when sending is possible.

RMB_TechRegiExtended

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V14	RMB_TechRegiExtended

Tab. 3-74: Reference table of the MB_RegiSensor01SendRecvDataType02 function block

Interface description

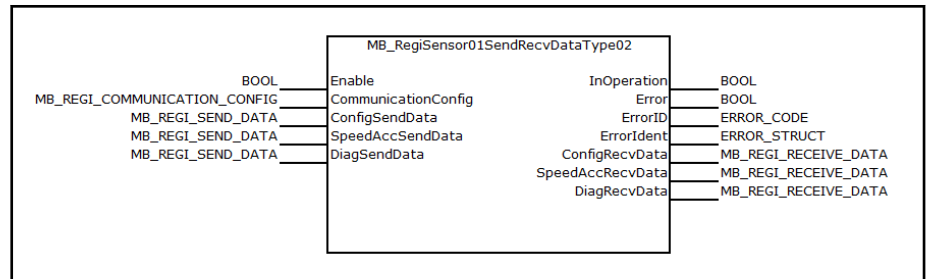


Fig. 3-49: MB_RegiSensor01SendRecvDataType02 function block

I/O type	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Processing of function block enabled.
	CommunicationConfig	MB_REGI_COMMUNICATION_CFG	Contains the UDP port of the control to which the control connects to and to which the sensor sends the measured data.
	ConfigSendData	MB_REGI_SEND_DATA	Includes data that the MB_RegiSensor01ConfigType02 function block wants to send.
	SpeedAccSendData	MB_REGI_SEND_DATA	Includes data that the MB_RegiSensor01SendSpeedAccType01 function block wants to send.
	DiagSendData	MB_REGI_SEND_DATA	Includes data that the MB_RegiSensor01ReadDiagType01 function block wants to send.
	TeachSendData	MB_REGI_SEND_DATA	Includes data that the MB_RegiSensor01TeachDataType01 function block wants to send.
VAR_OUTPUT	InOperation	BOOL	Function block is working.
	Error	BOOL	Processing completed with errors.
	ErrorID	ERROR_CODE	Error group
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics
	ConfigRecvData	MB_REGI_RECEIVE_DATA	Includes data that was received for the MB_RegiSensor01ConfigType02 function block.
	SpeedAccRecvData	MB_REGI_RECEIVE_DATA	Includes data that was received for the MB_RegiSensor01SendSpeedAccType01 function block.
	DiagRecvData	MB_REGI_RECEIVE_DATA	Includes data that was received for the MB_RegiSensor01ReadDiagType01 function block.
	TeachRecvData	MB_REGI_RECEIVE_DATA	Includes data that was received for the MB_RegiSensor01TeachDataType01 function block.

Tab. 3-75: I/O interface of the MB_RegiSensor01SendRecvDataType02 function block

RMB_TechRegiExtended

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
MB_REGI_COMMUNICATION_CONFIG					
IpSensor	STRING	LEN > 0	n. def	""	Rising edge at "Enable"
PortConfigura- tion	UINT	1	n. def	0	Rising edge at "Enable"
IpPLC	STRING	LEN > 0	n. def	""	Rising edge at "Enable"
PortMeasure- ment	UINT	1	n. def	0	Rising edge at "Enable"
MB_REGI_SEND_DATA					
NoOfBytes	UDINT	0	n. def	0	Change at "NewData"
ValueAddr	POINTER TO BYTE	n. def	n. def	0	Change at "NewData"
NewData	UINT	0	65535	0	Continuous
SensorPort	UINT	1	65535	0	Change at "NewData"

Tab. 3-76: Overview on min./max. values and initial values of the inputs

Data structures

The data structures [tab. 3-68 "MB_REGI_SEND_DATA data structure" on page 89](#) and [tab. 3-65 "MB_REGI_COMMUNICATION_CONFIG data structure" on page 88](#) are described in the [chapter 3.3.8 "MB_RegiSensor01ConfigType02" on page 85](#) function block.

Function description

If a communication should be established with a sensor, this function block has to be started at a rising edge at "Enable" before or at the same time with the attached function blocks MB_RegiSensor01SendSpeedAccType01, MB_RegiSensor01ConfigType01, MB_RegiSensor01TeachDataType01 and MB_RegiSensor01ReadDiagType01. At a rising edge at "Enable" and after checking the inputs, the respective UDP port "PortConfiguration" is bound to the control. Thus, the respective data can be received on this port. The function block waits for data. That means that it checks during each call whether the "NewData" input changes for the structures "ConfigSendData", "AccelerationSendData", "DiagSendData" and "TeachSendData". If this is the case, the respective data (NoOfBytes) is sent. The function block waits for the respective data if a response package is expected from the sensor (exception: no response package is expected when transmitting acceleration and velocity). A timeout handling also takes place. If no response package has been received after 2 seconds, the data package is sent again. An error is generated after third try and the transmission is aborted. If a package is received, the number of bytes and the address (ValueAddr) at which the data is present are specified. "NewData " at the output is set to the same value "NewData" had at the input when data was sent. Thus, it can be exactly allocated to which sent data package the response belongs to.

Data buffer: No data buffer is internally implemented since each created function block (if a handshake takes place) sends the next package only after it received the respective response.

Exception: If there is a new package with velocity and acceleration information and if it cannot be sent until a new package is created, the new package is sent and the old one is discarded.

Prioritization:

The transferred configuration data is not time-critical. Only the acceleration and velocity values has to be transferred as fast as possible. Thus, a prioritization is implemented. The data package of the MB_RegiSensor01SendSpeedAccType01 function block has absolute priority and is immediately sent as soon as the "send" function block is empty. If it is waited for a response package of the configuration or diagnostics within this period, it is first stopped, the time-critical package is sent in order to subsequently receive the configuration/diagnostic package. The prioritization is shown in the following figure.

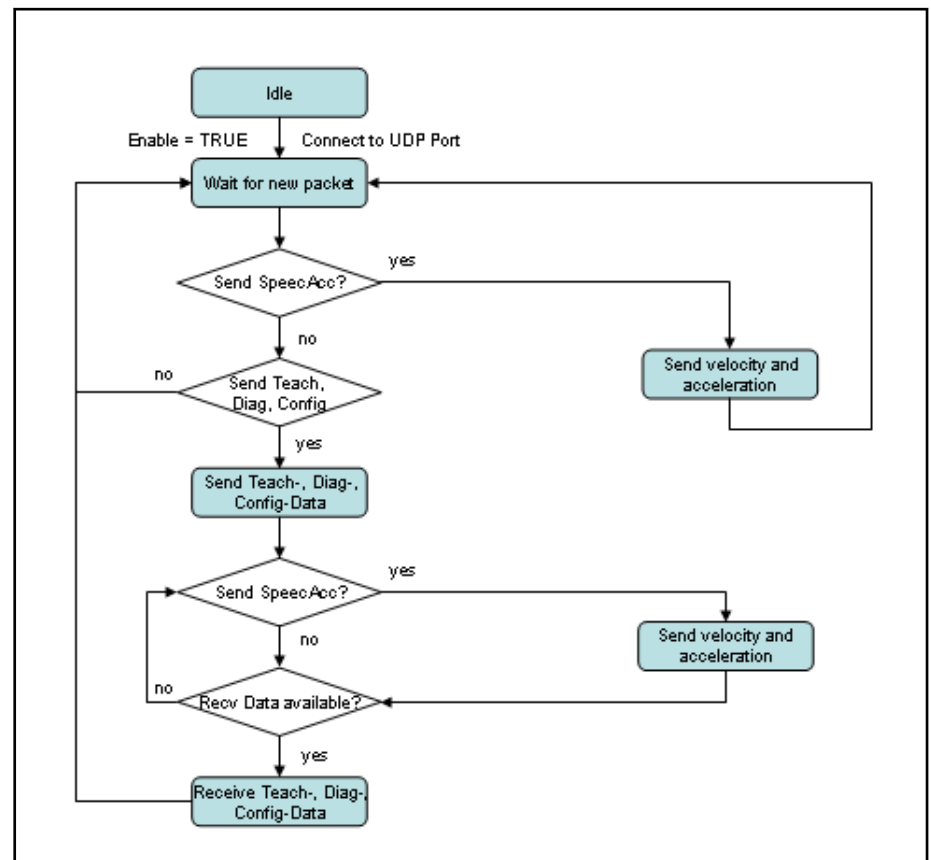


Fig. 3-50: Internal sequence of the MB_RegiSensor01SendRecvDataType02 function block

Error handling

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000003	Timeout while connecting to the UDP port
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000001B	Invalid UDP port, port configuration
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000004	Timeout while sending

RMB_TechRegiExtended

ErrorID	Additional 1	Additional 2	Description
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000005	Timeout while receiving
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000001C	Invalid sensor IP

Tab. 3-77: Error codes of the MB_RegiSensor01SendRecvDataType02 function block

Additionally, the function block can output errors of the mark stream sensor and of the library RIL_SocketComm.lib.

3.3.10 MB_RegiSensor01CycComType04

Brief description

The MB_RegiSensor01SendRecvDataType04 function block executes the communication with the mark stream sensor if a sensor firmware ≥ 12 is used. It executes only the cyclic communication. The sensor configuration is carried out by the MB_RegiSensor01ConfigType02 function block. Only after the sensor has been configured directly, it is started with the measurement and the function block can start the measurement evaluation. All measured values are specified for the length register and the side register at the output. This data can be evaluated by the user according to the configuration. This MB_RegiSensor01CycComType04 function block is especially suitable for applications in which several printed marks are measured to each other. This is for example the case if only one mark stream sensor is located at the end of the machine.

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V14	RMB_TechRegiExtended

Tab. 3-78: Reference table of the MB_RegiSensor01CycComType04 function block

Interface description

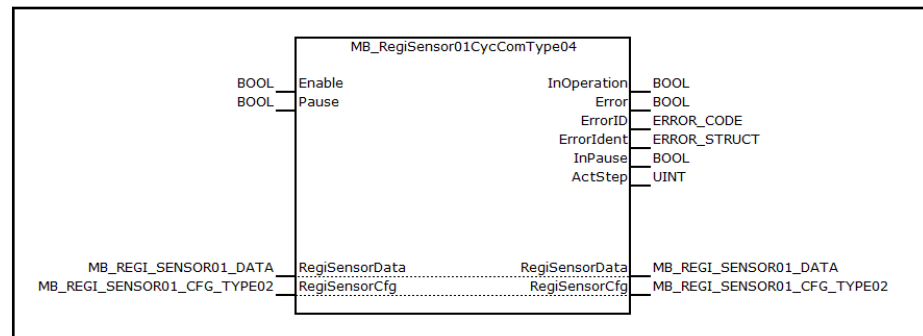


Fig. 3-51: The MB_RegiSensor01CycComType04 function block

I/O type	Name	Data type	Description
VAR_IN_OUT	RegiSensorCfg	MB_REGI_SENSOR01_CFG_TYPE02	Contains the current configuration data of the sensor.
	RegiSensorData	MB_REGI_SENSOR01_DATA	Contains the respective measured values of the sensor.
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled).
	Pause	BOOL	Cancels the function block processing. No new measured values are evaluated. The last measured data is frozen.

I/O type	Name	Data type	Description
VAR_OUTPUT	InOperation	BOOL	Function block is working.
	Error	BOOL	Processing completed with errors.
	ErrorID	ERROR_CODE	Error group
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics
	InPause	BOOL	Indicates that the "Pause" input is active and that the value in "RegiSensorData" does not change.
	ActStep	UINT	Indicates that a new data package with measured values was received and evaluated.

Tab. 3-79: I/O interface of the MB_RegiSensor01CycComType04 function block

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
Pause	BOOL			FALSE	Continuous

Tab. 3-80: Overview on min./max. values and initial values of the inputs

Data structures

MB_REGI_SENSOR01_DATA		
ActStep	UINT	Current step
LengthFeedback	ARRAY [0..16] OF REAL	It contains the mark distance from the zero pulse to the marks 1..16. In [0], the mark distance stands for web/web2 if a mark on the upper side was measured with one on the lower side.
SideFeedback	ARRAY [1..16] OF REAL	Contains the respective mark widths.
MarkState	ARRAY[0..16] OF MB_REGI_MARK_STATE	Status of the registration marks.
SensorState	MB_REGI_SENSOR_STATE	Outputs warnings and diagnostics of the sensor during measurement.

Tab. 3-81: MB_REGI_SENSOR01_DATA data structure

The data structure MB_REGI_SENSOR01_CFG_TYPE02 used (tab. 3-58 "MB_REGI_SENSOR01_CFG_TYPE02 data structure" on page 87) was described in the chapter "Interface description" on page 86 function block.

Enumeration type

The enumeration type MB_REGI_MARK_STATE can assume the following values:

Element	Value	Description
FOUND	0	Mark found
ROTATED	1	Mark rotated
LARGE	2	Mark too large
SMALL	3	Mark too small

RMB_TechRegiExtended

Element	Value	Description
MISSING	4	Mark missing
NOT_CONFIGURED	5	Mark not configured

Tab. 3-82: Enumeration type MB_REGI_MARK_STATE

The enumeration type MB_REGI_SENSOR_STATE can assume the following values:

Element	Value	Description
START	0	Starting state
SEARCHING	1	Search marks
INOP	2	Marks found (operating mode)
STOP	3	Web velocity below 0,4m/s – no evaluation
LAMP	4	Lamp defective
SENSOR	5	Sensor dirty
TEACHING	6	Sensor in teach phase

Tab. 3-83: Enumeration type MB_REGI_SENSOR_STATE

Function description

The function block may only be activated after the MB_RegiSensor01Config-Type02 function block was called to configure at a rising edge at "Enable" since the MB_RegiSensor01CycComType04 function block accesses the UDP port of the RegiSensorCfg structure and then connects. If no configuration has yet been executed, a faulty UDP port is present, since this port has not been written to the structure. In this case, the respective "Error" output of the function block is set.

The PLC works as UDP server in the cyclic communication. That means the control receives the measured value packages of the sensor at the UDP port via the acyclic communication. The sensor sends the measured data cyclically after each cylinder revolution (that means after each measurement evaluation). This measured data is evaluated by the function block. When setting "Enable", a bind() is made on UDP port defined in "RegiSensorCfg". Thus, it can be listened on the UDP port and measured values can be received and evaluated.

Each sensor thus requires 2 ports on the control: One for configuration data and diagnostic data and one for the measured value acquisition. The communication with the sensor to receive measured values is internally takes place in the function block.

The **RegiSensorData** contains all measured values.

LengthFeedback[i]:

The "LengthFeedback" array contains all measured mark distances with regard to the zero pulse as well as a measured value of the web/web2 procedure that are entered into the array as follows:

LengthFeedback[0]: Mark distance between the defined mark below and the mark above.

LengthFeedback[1]: Mark distance between the zero pulse and mark 1.

LengthFeedback[2]: Mark distance between the zero pulse and mark 2.

...

LengthFeedback[15]: Mark distance between the zero pulse and mark 15.

LengthFeedback[16]: Mark distance between the zero pulse and mark 16.

If a measured value is not available since it has not been configured or if less marks are available, the return value is 0.0 and defined correspondingly as "Missing" or "Not_Configured" in the "MarkState".

SideFeedback[i]: The "SideFeedback" array contains all measured mark widths that are entered in the array as follows:

SideFeedback[1]: Mark width of mark 1

SideFeedback[2]: Mark width of mark 2

...

SideFeedback[16]: Mark width of mark 16

If a measured value is not available as the mark has not been configured, the returned value is 0.0 and defined accordingly in MarkState as Not_Configured. If a mark is not found and output as "Missing" in the MarkState, the information about the reason for the mark loss is contained in the SideFeedback.

The diagnostic numbers are listed in the following table:

Diagnostic number	Significance
11	Position tolerance exceeded
24	Height tolerance not fulfilled
26	Width tolerance violated
30	ShiftSyncPulse has been offset to the left due to overflow
31	ShiftSyncPulse has been offset to the right due to overflow
39	Double mark clearance was not found
All other diagnostic numbers	Internal error numbers

Tab. 3-84: Diagnostic numbers at "Missing" mark

SensorState: It is not set.

MarkState[i]: Indicates the validity of the values in the arrays "LengthFeedback" and "SideFeedback". If the measured value of mark 1 is valid for example, then, MarkState[1] = FOUND. Thus, the values "LengthFeedback[1]" and "SideFeedback[1]" can be evaluated. Otherwise, MarkState[1] = Missing or Not_Configured. That depends on the fact whether the mark was omitted or not configured.

Error handling

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000003	Timeout while connecting
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000020	No valid configuration found
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000000B	Invalid UDP port

Tab. 3-85: Error codes of the MB_RegiSensor01CycComType04 function block

RMB_TechRegiExtended

Additionally, the function block can output errors of the mark stream sensor and of the library RIL_SocketComm.lib.

3.3.11 MB_RegiSensor01ReadDiagType01

Brief description

The MB_RegiSensor01ReadDiagType01 function block reads the diagnostic data and the electronic type plate code of the sensor and stores them in the data structure. This function block requires the MB_RegiSensor01SendRecvData function block that is responsible for the UDP communication with the sensor to send and receive data.

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V11	RMB_TechRegiExtended

Tab. 3-86: Reference table of the MB_RegiSensor01ReadDiagType01 function block

Interface description

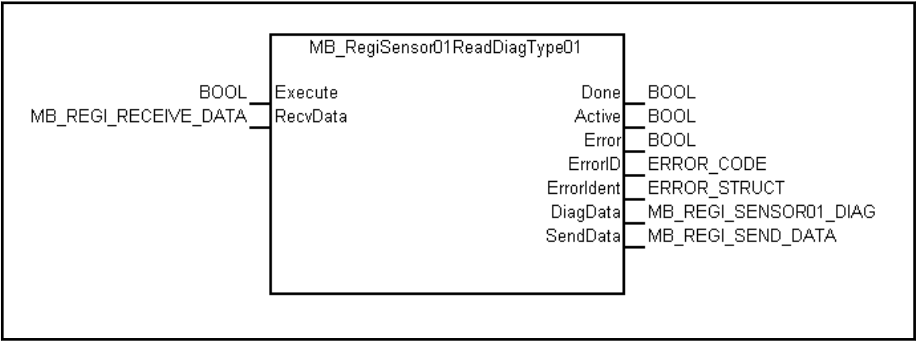


Fig. 3-52: MB_RegiSensor01ReadDiagData Type01 function block

I/O type	Name	Data type	Description
VAR_INPUT	Execute	BOOL	Starts reading the diagnostic data,
	RecvData	MB_REGI_RECEIVE_DATA	Received data from the MB_RegiSensor01SendRecvDataType01 function block.
VAR_OUTPUT	Done	BOOL	Reading the diagnostic data was successful.
	Active	BOOL	Diagnostic data is currently read.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Error group
	ErrorIdent	ERROR_STRUCT	Detailed error cause.
	DiagData	MB_REGI_SENSOR01_DIAG	Diagnostic data of the sensor.
	SendData	MB_REGI_SEND_DATA	Data to be sent via the MB_RegiSensor01SendRecvData function block.

Tab. 3-87: I/O interface of the MB_RegiSensor01ReadDiagData Type01 function block

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
MB_REGI_RECEIVE_DATA					
NoOfBytes	UDINT	0	n. def	0	Change at "NewData"
ValueAdr	POINTER TO BYTE	n. def	n. def	0	Change at "NewData"
NewData	UINT	0	65535	0	Continuous

Tab. 3-88: Overview on min./max. values and initial values of the inputs

Data structures

MB_REGI_SENSOR01_DIAG		
ComponentName	STRING(52)	Indicates the component name
VendorName	STRING(28)	Vendor name
VendorCode	DWORD	Vendor code
DeviceName	STRING(40)	Type code
VendorDeviceID	STRING(24)	BR parts number
FunctionRevision	WORD	Function version
HardwareRevision	STRING(16)	Hardware version
SoftwareRevision	STRING(16)	Software version
SerialNumber	STRING(24)	Serial number
OperationalHours	DWORD	Operating hour counter in hours
MAC address	STRING(17)	Mac address
ChromaticityX	REAL	Current color intensity x (resolution 1/32768)
ChromaticityY	REAL	Current color intensity y (resolution 1/32768)
Intensity	UDINT	Current intensity
Gain	UINT	Current gain (0..7)
Velocity	UDINT	Current velocity (Inc/s)
Status	WORD	Status (bit-coded)
MeasurementState	WORD	Measurement status (start/stop)
SmoothFactorVel	UINT	Smoothing factor, averaging velocity (0..8)
Acceleration	DINT	Current acceleration = (inc/s ²)
SmoothFactorSensor	UINT	Smoothing factor sensor (0..4)
SensorSignals	UINT	Sensor signals

Tab. 3-89: MB_REGI_SENSOR01_DIAG data structure

The data structure [tab. 3-24 "MB_REGI_SEND_DATA data structure"](#) on [page 43](#) used was described in the [chapter 3.3.3 "MB_RegiSensor01Config-Type01"](#) on [page 39](#) function block.

RMB_TechRegiExtended

Function description

The function block reads the diagnostic data of the sensor and stores them in the data structure. The first 11 data sets represent the electronic type code. The following parameters are settings and current states of the sensor that can be read out and shown via these.

The evaluation of the non-self-explanatory errors are described in the following:

Status

"LowByte" and "HighByte" are not distinguished. "LowByte" and "HighByte" can be changed independently from each other:

0x0800: Channel Blue, sensor is overdriven

0x1000: Channel Green, sensor is overdriven

0x2000: Channel Red, sensor is overdriven

0x0001: Idle mode

0x0002: Configuration mode

0x0003: Measurement mode

0x0004: Teach mode

MeasurementState

0x0000: No measurement started

0x0001: Measurement started

SensorSignals

Reserved for further diagnostics

Troubleshooting

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000000	Timeout while receiving a package.

Tab. 3-90: Error codes of the MB_RegiSensor01ReadDiagDataType01 function block

Additionally, the function block can output errors of the mark stream sensor.

3.3.12 MB_RegiSensor01SendSpeedAccType01

Brief description

This function block sends the respective velocity and acceleration information to the mark stream sensor in a defined interval. Data is calculated and sent in the respective intervals dependent on a time interval. The smaller the time interval, the more values are sent and the more data traffic is created. Thus, the time interval should be at least the double of the current cycle time of the task during which the function block is called. Thus, the sensor can even guarantee a highly accurate measurement in acceleration phases. The MB_RegiSensor01SendSpeedAccType01 function block calculates the current acceleration and velocity values and creates the respective data package. This data package is applied at the output and is sent via the MB_RegiSensor01SendRecvDataType01 function block. That means that this function block either has to be started at the same time or before. Since it is about

time-critical data, the data packages are considered immediately with highest priority or sent as soon as the "send" function block is empty.

It is recommended to use the MB_RegiSensor01SendSpeedAccType01 function block only in acceleration phases and to deactivate it at constant run due to data traffic. If the function block should also work at constant run, a time interval of > 250 ms is recommended. If the function block sends data, the sensor does not determine any acceleration data anymore from the incremental encoder information. Only after a falling edge at "Enable", the sensor determines velocity information from the incremental encoder information.



The function block MB_RegiSensor01SendSpeedAccType01 has to be called in a Motion-synchronous task to ensure a correction calculation of the acceleration. If this is not possible due to the runtime, the "mMotionCyclic()" function of the function block from the Motion-synchronous tasks has to be called and the MB_RegiSensor01SendSpeedAccType01 function block can be called in a cyclic task. The call of the MotionTask has to be provided with the following structure: {CyclicTask}.fbInstanz}.mMotionCyclic(). The elements in the curly brackets have to be replaced by the program name in the cyclic task and the instance name of the function block. Thus, the method call from a program running in the Motion task could be as follows: PLCProg.fbRegiSensor01SendSpeedAccType01.mMotionCyclic(). If this call is not executed, a correct calculation of the acceleration cannot be ensured.

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V11	RMB_TechRegiExtended

Tab. 3-91: Reference table of the MB_RegiSensor01SendSpeedAccType01 function block

Interface description

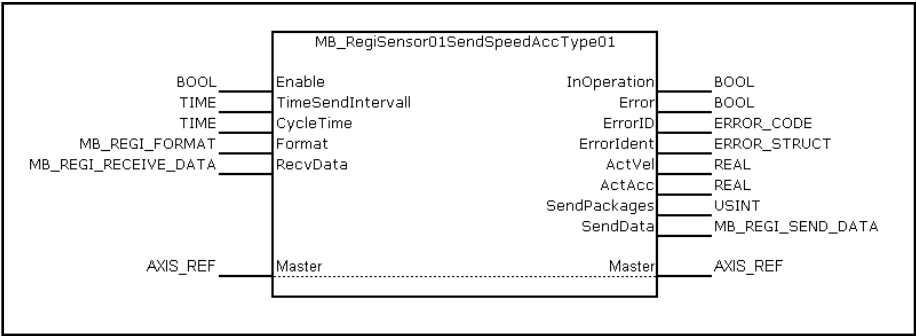


Fig. 3-53: MB_RegiSensor01SendSpeedAccType01 function block

RMB_TechRegiExtended

I/O type	Name	Data type	Description
VAR_IN_OUT CONST	Master	AXIS_REF	Axis whose velocity and acceleration information should be sent. Due to the accuracy, it is recommended to create a virtual master axis.
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled).
	TimeSendIntervall	TIME	Data is calculated and sent in this interval.
	CycleTime	TIME	Cycle time of the PLC task in which the function block is called.
	Format	MB_REGI_FORMAT	Contains the printing cylinder circumference and the number of increments per revolution.
	RecvData	MB_REGI_RECEIVE_DATA	Received data from the MB_RegiSensor01SendRecvDataType01 function block.
VAR_OUTPUT	InOperation	BOOL	Function block is working.
	Error	BOOL	Processing completed with errors.
	ErrorID	ERROR_CODE	Error group
	ErrorIdent	ERROR_STRUCTURE	Detailed diagnostics
	ActVel	REAL	Current velocity in [Inc/s] transmitted to the sensor.
	ActAcc	REAL	Current acceleration in [Inc/s^2] transmitted to the sensor.
	SendPackages	USINT	Package counter if new data is calculated and transferred for sending.
	SendData	MB_REGI_SEND_DATA	Data to be sent via the MB_RegiSensor01SendRecvData function block.

Tab. 3-92: I/O interface of the MB_RegiSensor01SendSpeedAccType01 function block

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
TimeSendIntervall	TIME	$\geq 2 * \text{CycleTime}$	n. def	t#100ms	Rising edge at "Enable"
CycleTime	TIME	> 0	n. def	t#8ms	Rising edge at "Enable"
MB_REGI_FORMAT					
FormatLength	REAL	>0	n. def	0	Rising edge at "Enable"
Increments	UDINT	>0	n. def	0	Rising edge at "Enable"
MB_REGI_RECEIVE_DATA					
NoOfBytes	UDINT	0	n. def	0	Change at "NewData"
ValueAddr	POINTER TO MB_REGI_SENSOR01_ACK_BUFFER	n. def	n. def	0	Change at "NewData"
NewData	UINT	0	65535	0	Continuous

Tab. 3-93: Overview on min./max. values and initial values of the inputs

Data structures

The data structures [tab. 3-15 "MB_REGI_FORMAT data structure" on page 41](#) and [tab. 3-25 "MB_REGI_RECEIVE_DATA data structure" on page 43](#) are described in the [chapter 3.3.3 "MB_RegiSensor01ConfigType01" on page 39](#) function block.

Function description

The inputs are checked and applied at a rising edge at "Enable". First, a time-out value is transferred to the sensor (c.f. Init package). That means if no new acceleration or velocity value is sent anymore, the velocity is determined again by the sensor after this period. This value is fixedly determined with 10 s. The sensor waits for the response package to subsequently determine the first current acceleration and to provide it to the function block MB_RegiSensor01SendRecvDataType01 so that the value is transferred to the sensor. From this point in time onwards, a new velocity and acceleration is determined after the time interval has been completed. The time interval has to be at least twice the cycle time of the task in which the function block is called to control data traffic and that other function blocks can still send and receive data. At a falling edge at "Enable", an end package is sent to the sensor. This end package notifies the sensor that it should determine its velocity from the incremental encoder signals. It is waited for the acknowledgement package of the sensor. Thus, it can happen that after a falling edge at "Enable", x cycle times are still present "InOperation".

RMB_TechRegiExtended

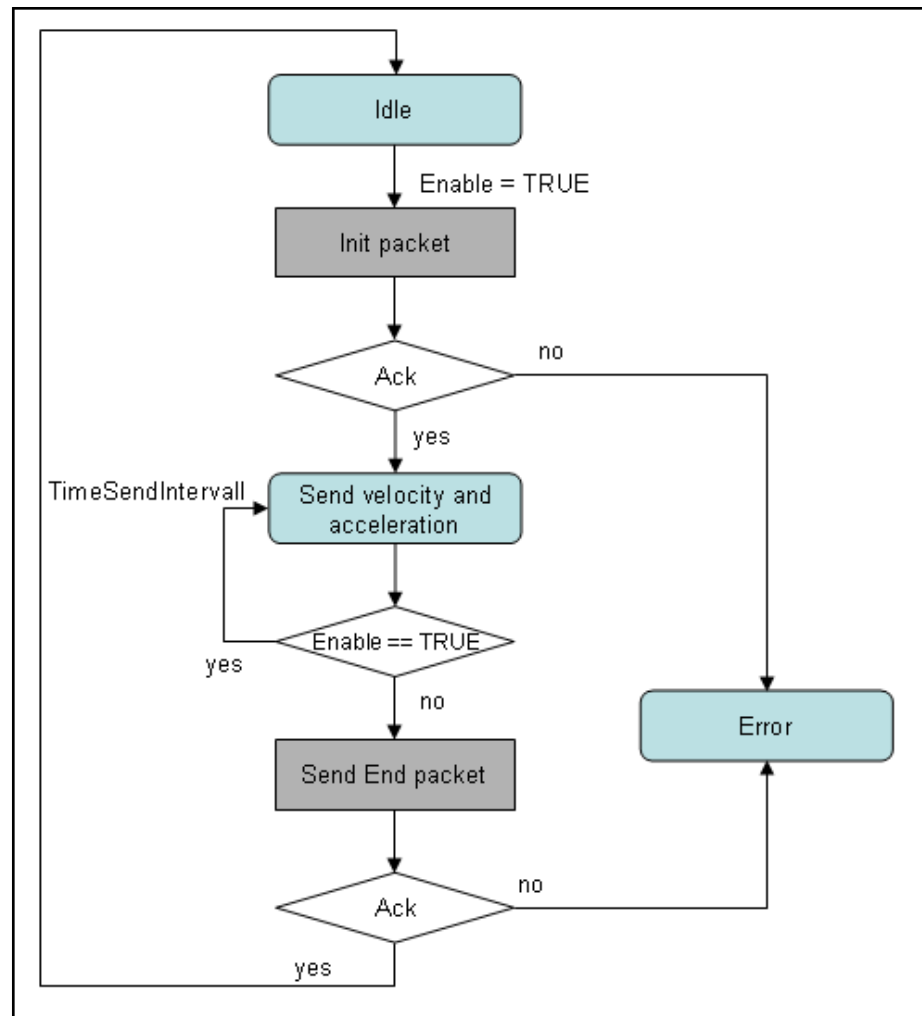


Fig. 3-54: Behavior of the `MB_RegiSensor01SendSpeedAccType01` function block

Handshake: The function block indicates with the incrementation from "NewData" to "SendData" that it wants to send new data. This is executed by the `MB_RegiSensor01SendRecvDataType01` function block. If it receives new data, it sets the "NewData" value equal to the "RecvData" value. Thus, the function block knows that it is the respected response package. A handshake is only executed when transferring the timeout and the end diagram. When sending the acceleration and velocity, no response package of the sensor is generated, since it is time-critical data and thus, old data would be transferred when repeating the sending.

Timeout: If a new data package is sent, an internal timer runs in the function block. If it gets no response within 10 seconds, it outputs a timeout and aborts with an error. A possible error case is that the `MB_RegiSensor01SendRecvDataType01` function block is not activated or incorrectly connected.

Error handling

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00001D11	16#00000000	Error in the state machine
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000022	Invalid time interval
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000023	Invalid cycle time
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000000	Invalid plate cylinder format
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000000D	Invalid number of increments
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000006	Timeout; no response from the MB_RegiSensor01SendRecvDataType01 function block within 10 seconds
INPUT_RANGE_ERROR (16#0006)	16#0000000D	16#00000000	AxisNo of "Master" is not within the valid range

Tab. 3-94: Error codes of the MB_RegiSensor01SendSpeedAccType01 function block

Additionally, the function block can output errors of the mark stream sensor.

3.3.13 MB_RegiSensor01TeachDataType01

Brief description

The MB_RegiSensor01TeachDataType01 function block reads and writes measured data. Currently valid measured data can be read out from the mark stream sensor and can be directly loaded to the sensor after machine stop or power failure for example. Thus, the measurement can be started immediately without transferring a configuration or teaching again.

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V11	RMB_TechRegiExtended

Tab. 3-95: Reference table of the MB_RegiSensor01TeachDataType01 function block

Interface description

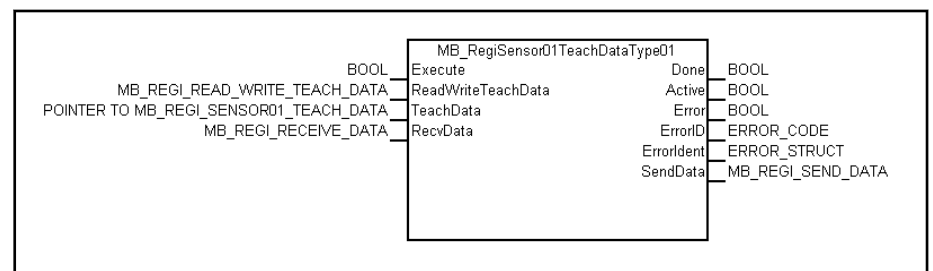


Fig. 3-55: MB_RegiSensor01TeachDataType01 function block

RMB_TechRegiExtended

I/O type	Name	Data type	Description
VAR_INPUT	Execute	BOOL	Starts loading or saving of measured data.
	ReadWrite-TeachData	MB_REGI_READ_WRITE_TEACH_DATA	Selection whether the data should be uploaded to the sensor or downloaded from it.
	TeachData	POINTER TO MB_REGI_SENSOR01_TEACH_DATA	Address of the data type MB_REGI_SENSOR01_TEACH_DATA in which data is stored and from where data can be loaded.
	RecvData	MB_REGI_RECEIVE_DATA	Received data from the MB_RegiSensor01SendRecvDataType01 function block.
VAR_OUT-PUT	Done	BOOL	Loading or saving measured data was successful.
	Active	BOOL	Loading or saving measured data is currently running.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Error group
	ErrorIdent	ERROR_STRUCT	Detailed error cause.
	SendData	MB_REGI_SEND_DATA	Data to be sent via the MB_RegiSensor01SendRecvData function block.

Tab. 3-96: I/O interface of the MB_RegiSensor01TeachDataType01 function block

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
ReadWrite-TeachData	MB_REGI_READ_WRITE_TEACH_DATA	0	1	0	At a rising edge at "Execute"
TeachData	POINTER TO MB_REGI_SENSOR01_TEACH_DATA	> 0	n. def	0	At a rising edge at "Execute"
MB_REGI_RECEIVE_DATA					
NoOfBytes	UDINT	0	n. def	0	Change at "NewData"
ValueAdr	POINTER TO BYTE	0	n. def	0	Change at "NewData"
NewData	UINT	0	65535	0	Continuous

Tab. 3-97: Overview on min./max. values and initial values of the inputs

Data structures

MB_REGI_SENSOR01_TEACH_DATA		
TeachData	ARRAY [0..1310] OF BYTE	Measured data (teach data)

Tab. 3-98: MB_REGI_SENSOR01_TEACH_DATA data structure

The data structures [tab. 3-24 "MB_REGI_SEND_DATA data structure" on page 43](#) and [tab. 3-25 "MB_REGI_RECEIVE_DATA data structure" on page 43](#) are described in the [chapter 3.3.3 "MB_RegiSensor01ConfigType01" on page 39](#) function block.

Enumeration type

The enumeration type **MB_REGI_READ_WRITE_TEACH_DATA** can assume the following values:

Element	Value	Description
READ_TEACH_DATA	0	Reads data from the sensor
WRITE_TEACH_DATA	1	Writes data to the sensor

Tab. 3-99: Enumeration type **MB_REGI_READ_WRITE_TEACH_DATA**

Function description

The function block reads/writes the current measured data from/to the mark stream sensor at a rising edge.

MB_READ_WRITE_TEACH_DATA:

If **READ_TEACH_DATA** is selected, the measured data valid last is read from the mark stream sensor. Data is stored under the given "TeachData" address.

If **WRITE_TEACH_DATA** is selected, the measured data is written to the mark stream sensor. Therefore, the data at the specified "TeachData" address is sent to the mark stream sensor. After sending the data, the measurement can be started immediately without transferring the configuration data or starting a new teaching.

Error handling

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000029	Enumeration type MB_REGI_READ_WRITE_TEACH_DATA is invalid
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000028	TeachData points to 0 (zero pointer)
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000005	Timeout while receiving a package.

Tab. 3-100: Error codes of the **MB_RegiSensor01TeachDataType01** function block

Additionally, the function block can output errors of the mark stream sensor.

3.3.14 MB_RegiSensor01SetIPAddressType01

Brief description

The **MB_RegiSensor01SetIPAddressType01** function block stores the IP address of the sensor in the permanent sensor memory. The MAC address is required to select the sensor and it is printed on the casing of the evaluation electronics.

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V11	RMB_TechRegiExtended

Tab. 3-101: Reference table of the **MB_RegiSensor01SetIPAddressType01** function block

RMB_TechRegiExtended

Interface description

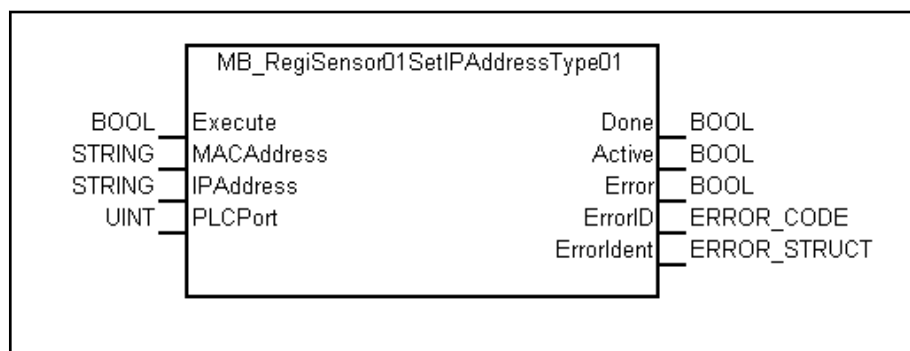


Fig. 3-56: MB_RegiSensor01SetIPAddressType01 function block

I/O type	Name	Data type	Description
VAR_INPUT	Execute	BOOL	Initiates the setting of the IP address.
	MACAddress	STRING	MAC address as HEX string. Colon to separate the six hex values allowed.
	IPAddress	STRING	4-digit IP address as string separated with points.
	PLCPort	UINT	Port from which the IP address is sent from the control to the sensor.
VAR_OUTPUT	Done	BOOL	Setting the IP address was successful.
	Active	BOOL	IP address is currently set.
	Error	BOOL	Indicates that an error occurred.
	ErrorID	ERROR_CODE	Error group
	ErrorIdent	ERROR_STRUCT	Detailed error cause.

Tab. 3-102: I/O interface of the MB_RegiSensor01SetIPAddressType01 function block

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
MACAddress	STRING	LEN = 0	LEN = 17 (only the lengths 0, 12 and 17 are valid)	""	Rising edge at "Execute"
IPAddress	STRING	LEN > 0	n. def	""	Rising edge at "Execute"
PLCPort	UINT	> 0	n. def	6000	Rising edge at "Execute"

Tab. 3-103: Overview on min./max. values and initial values of the inputs

Function description

The function block sets the IP address of the mark stream sensor. The sensor is selected via the MAC address. The MAC address consists of six hex values that can be separated by colons. Leading zeros may not be omitted. If no MAC address is specified, only one sensor may be in the Ethernet.

The IP address consists of four decimal values between 0 and 255 that are separated by points.

The function starts at a rising edge at "Execute". The sensor sends **no** response on the setting of the IP address. "Done" becomes TRUE after a suc-

cessful execution. The IP address can be permanently saved in the sensor. It is possible to set a new IP address by calling the function block again.

If an error occurs, "Error" is TRUE and "ErrorIdent" contains exact information on the error.



The function block sends the new IP address to the sensor. The sensor does not send any response. The sensor has to be reset after the transmission of the IP address. The "Done" output only means that the package was successfully sent. The user has to find out whether the sensor is provided with the newly assigned IP after reset.

The length of the MAC address may vary between 12 and 17 characters. Other lengths are not allowed. The separator can be arbitrarily selected. There has to be one separator after two characters.

Examples of MAC addresses in the "MACAddress" input:

- "00:60:35:08:35:9E"
- "00603508359E"

It is strongly recommended to always specify the respective MAC address to ensure that the same IP address is not assigned to several sensors. The MAC address can only be omitted in case of a direct point-to-point connection.

Error handling

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000000	Timeout while reading
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000009	Specify invalid IP address
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000026	Invalid PLC port
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000003	Connection error
COMMUNICATION_ERROR (16#0002)	16#00001D10	16#00000004	Timeout while sending
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000027	MAC address format incorrect

Tab. 3-104: Error codes of the MB_RegiSensor01SetIPAddressType01

Additionally, the function block can output errors of the mark stream sensor and of the library RIL_SocketComm.lib.

3.4 Function block to create defined velocity characteristics

3.4.1 Introduction and overview

To illustrate different velocity characteristics (linear, s-curve), the following function block is provided in the RMB_TechRegiExtended library:

- MB_ReachVelocityType01

This is used to calculate values for acceleration, velocity and jerk depending on the input variables.

RMB_TechRegiExtended

3.4.2 MB_ReachVelocityType01

Brief description

The MB_ReachVelocityType01 function block calculates the respective jerk, acceleration and target velocity to travel along an S-curve. Therefore, a jerk limitation from 0% to 100 % can be selected. If the input value is 0, the velocity is linear. If it is 100%, it is shaped as S-curve. The command values for velocity, acceleration and jerk are calculated from the acceleration period, the current velocity, the target velocity and the JerkFactor. The acceleration phase can be stopped via "StopAcc".

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V11	RMB_TechRegiExtended

Tab. 3-105: Reference table of the MB_ReachVelocityType01 function block

Interface description

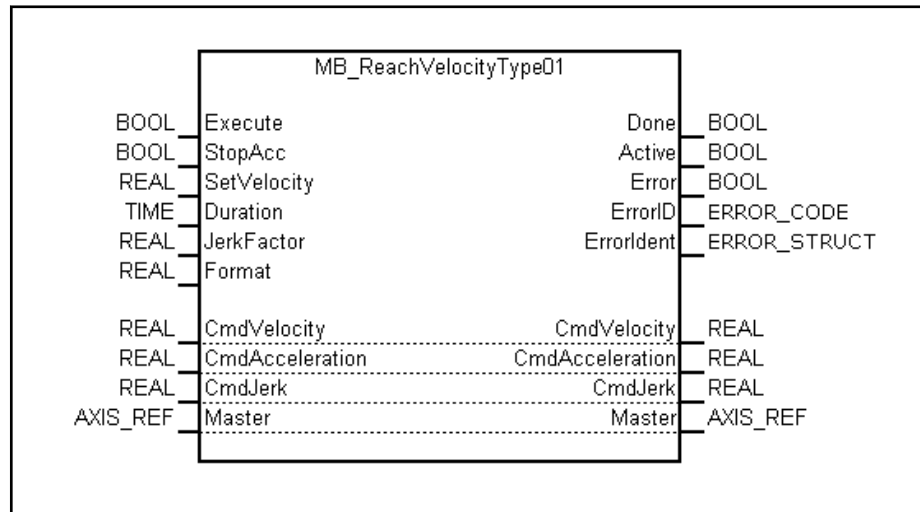


Fig. 3-57: MB_ReachVelocityType01 function block

I/O type	Name	Data type	Description
VAR_IN_OUT CONST	Master	AXIS_REF	Contains information on the axis whose velocity should be changed accordingly. It is recommended to create a virtual master axis.
VAR_IN_OUT	CmdVelocity	REAL	Command velocity [rpm]
	CmdAcceleration	REAL	Command acceleration [rad/s ²]
	CmdJerk	REAL	Command jerk [rad/s ³]
VAR_INPUT	Execute	BOOL	Processing of function block enabled.
	StopAcc	BOOL	Stops further acceleration.
	SetVelocity	REAL	Target velocity [m/min]
	Duration	TIME	Acceleration time
	JerkFactor	REAL	Jerk limitation of the acceleration profile in %.
	Format	REAL	Format [mm]

I/O type	Name	Data type	Description
VAR_OUTPUT	Done	BOOL	Acceleration phase successfully completed.
	Active	BOOL	Function block is working and currently in an interim state.
	Error	BOOL	Processing completed with errors.
	ErrorID	ERROR_CODE	Error group
	ErrorIdent	ERROR_STRUCT	Diagnostic description in case of error.

Tab. 3-106: I/O interface of the MB_ReachVelocityType01 function block

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
StopAcc	BOOL			FALSE	Continuous
SetVelocity	REAL	n. def	n. def	0	Rising edge at "Execute"
Duration	TIME	> t#0s	n. def	t#30s	Rising edge at "Execute"
JerkFactor	REAL	0	100	50	Rising edge at "Execute"
Format	REAL	> 0	n. def	1000	Rising edge at "Execute"

Tab. 3-107: Overview on min./max. values and initial values of the inputs

Function description

The acceleration phase starts at a rising "Execute" edge. The acceleration period (duration), the web velocity to be set (SetVelocity) as well as the acceleration profile (JerkFactor) can be previously specified and applied. A special feature is that "Active" is set at a rising edge at "Execute" but that the acceleration phase starts only after 500 ms. That means that the following function blocks (e.g. the register control) can use "Active" to see the beginning of the acceleration phase in advance. After the target velocity has been reached, "Active" is also present for further 500 ms until the processing is completed with "Done".

The JerkFactor effect is shown in the following figure:

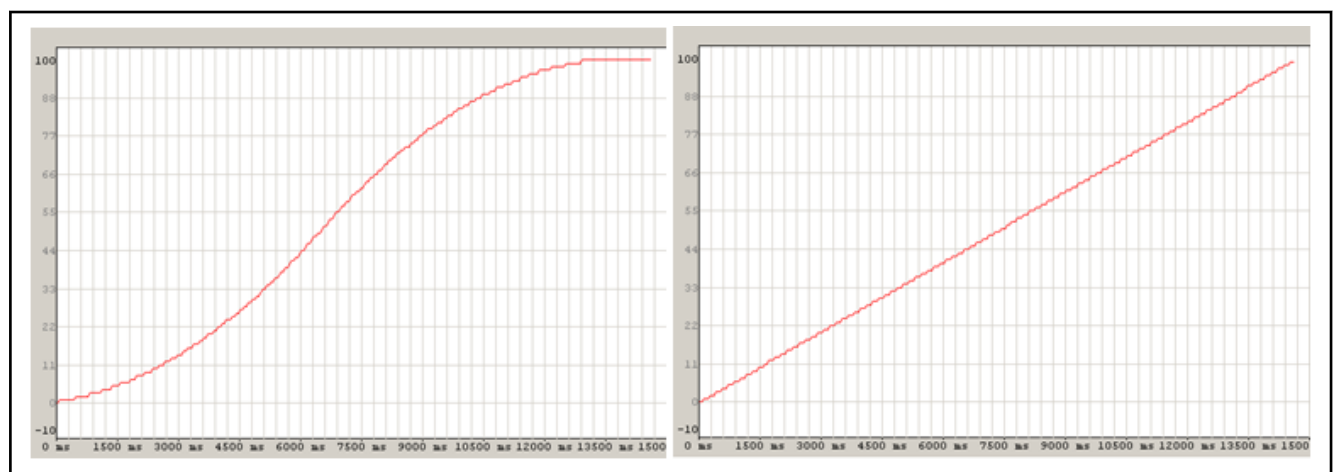


Fig. 3-58: Exemplary acceleration phases

The figure on the left shows an S-curve in its velocity characteristics during an acceleration phase from 0 to 100 m/min (SetVelocity) in 15 seconds (duration) with the JerkFactor = 100 % and a format of 1 m.

The figure on the right shows the linear velocity characteristics during an acceleration phase from 0 to 100 m/min (SetVelocity) in 15 s (duration) with the JerkFactor = 0 % and a format of 1 m.

The acceleration phase triggered with "Execute" can only be stopped with "StopAcc". The function block immediately stops further acceleration. "Done" becomes active. If the set velocity value "SetVelocity" should still be approached, "StopAcc" has to be deactivated and another rising edge of "Execute" has to be set.

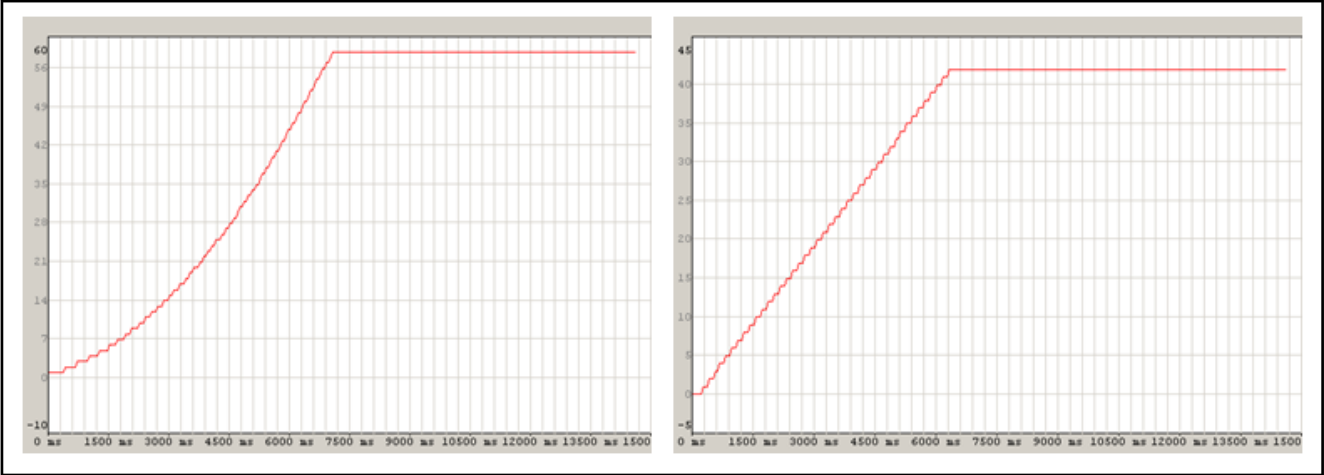


Fig. 3-59: Aborting the acceleration phases (left an abort at 7000 ms, right an abort at 6300 ms)

The figure shows an abort of the acceleration phase for an S-curve and linear velocity characteristics.

The meaning of the jerk is explained in the following: The jerk is the temporal derivation of the acceleration. The JerkFactor indicates the temporal part of the acceleration phase in percent in which the jerk is not zero. The following figure shows the meaning of the jerk. If the acceleration does not change anymore, the jerk is 0.

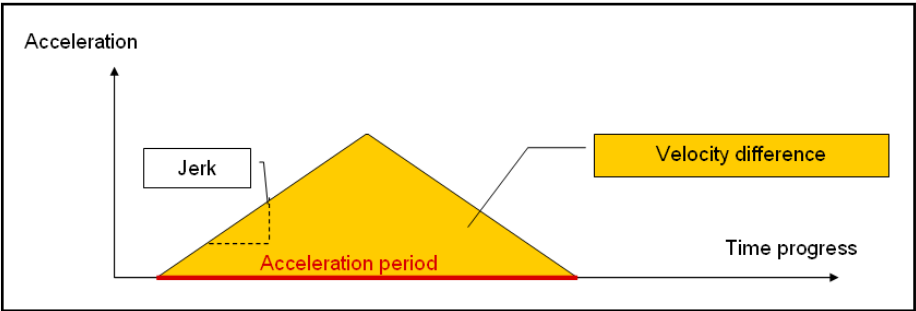


Fig. 3-60: Importance of the jerk in the acceleration phase

Function handling

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000000	Invalid printing cylinder format
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000024	Invalid period

RMB_TechRegiExtended

ErrorID	Additional 1	Additional 2	Description
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000025	Invalid jerk factor
INPUT_RANGE_ERROR (16#0006)	16#0000000D	16#00000000	AxisNo of "Master" is not within the valid range

Tab. 3-108: Error codes of the MB_ReachVelocityType01 function block

4 RMB_TechRegiAdvanced

4.1 RMB_TechRegiAdvanced library subject to license



The use of the functions blocks of the RMB_TechRegiAdvanced is subject to license. Depending on the application, one of the three licenses, either FWS-IM*MLC-TEC-NNVRS-NN-REGI*EXT, FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV1 or FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV2 has to be purchased to use the function blocks. For each control on which the function blocks are used, one license is required each (runtime license).

Einzellizenz Technologiebibliothek
Berechtigt zur Verwendung mit einer einzigen Steuerungsfirmware.

Single licence for technology library
Authorize the use with exactly one controller firmware.

TechLib RegisterControl Extended TechLib RegisterControl Extended		
Licence key (MNR): R911xxxxxx	Serial number: 1234567	Type code: FWS-IM*MLC-TEC-NNVRS-NN-REGI*EXT
		
Registration key WAHOF2-FBJYZF-8CSB0F-9DQO0D-9E8ZER		

Fig. 4-1: FWS-IM*MLC-TEC-NNVRS-NN-REGI*EXT license key

Einzellizenz Technologiebibliothek
Berechtigt zur Verwendung mit einer einzigen Steuerungsfirmware.

Single licence for technology library
Authorize the use with exactly one controller firmware.

TechLib RegisterControl Advanced 1 TechLib RegisterControl Advanced 1		
Licence key (MNR): R911xxxxxx	Serial number: 1234567	Type code: FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV1
		
Registration key WAHOF2-FBJYZF-8CSB0F-9DQO0D-9E8ZER		

Fig. 4-2: FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV1 license key

Einzellizenz Technologiebibliothek Berechtigt zur Verwendung mit einer einzigen Steuerungsfirmware.		
Single licence for technology library Authorize the use with exactly one controller firmware.		
TechLib RegisterControl Advanced 2 TechLib RegisterControl Advanced 2		
Licence key (MNR): R911xxxxxx	Serial number: 1234567	Type code: FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV2
		
Registration key WAHOF2-FBJYZF-8CSB0F-9DQO0D-9E8ZER		

Fig. 4-3: FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV2 license key

The figures show the corresponding license keys. One license has to be purchased for each control authorizing the use of function blocks of the RMB_TechRegiExtended library or the RMB_TechRegiAdvanced on this control system.

The libraries RMB_TechRegiExtended and RMB_TechRegiAdvanced are provided with the three licenses FWS-IM*MLC-TEC-NNVRS-NN-REGI*EXT, FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV1 and FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV2. The content of the licenses is shown in the following:

FWS-IM*MLC-TEC-NNVRS-NN-REGI*EXT

- HMI for narrow web applications (up to twelve printing units, no decoupling and acceleration compensation)
- Firmware Update Tool
- Documentations: Application Manual and Library Description
- Libraries: RMB_TechRegiExtended and RMB_TechRegiAdvanced

FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV1

- HMI for applications with clamping units (such as gravure printing) (up to twelve printing units, with decoupling and acceleration compensation)
- Firmware Update Tool
- Documentations: Application Manual and Library Description
- Libraries: RMB_TechRegiExtended and RMB_TechRegiAdvanced

FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV2

- HMI for applications with clamping units (such as gravure printing) (up to 24 printing units, with decoupling and acceleration compensation)
- Firmware Update Tool
- Documentations: Application Manual and Library Description
- Libraries: RMB_TechRegiExtended and RMB_TechRegiAdvanced

If either the RMB_TechRegiExtended library or the RMB_TechRegiAdvanced library is used, one of the licenses FWS-IM*MLC-TEC-NNVRS-NN-REGI*EXT, FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV1 or FWS-IM*MLC-TEC-NNVRS-NN-REGI*ADV2 is required for each control on which one of the function blocks is used.

4.2 General information

The RMB_TechRegiAdvanced library provides function blocks to control and optimize the printing processes.

It is a library that is charged for, requiring and containing the RMB_TechRegiExtended library. One license is required for each control system used.

This documentation describes the inputs and outputs of the individual function blocks and provides notes on their use.

The following overview shows all function blocks of the RMB_TechRegiAdvanced library requiring a license:

Function block	Description
MB_RegisterControllerAdvType01	Register controller for applications with and without clamping units
MB_RegiAccelerationCompensationType01	Model-based and self-regulating acceleration compensation
MB_RegiGravureRegulateType01	Dynamic decoupling network
MB_RegiGravureRegulateType02	Dynamic decoupling network with preset of a single printing unit and disengageable decoupling values of upstream printing units

Tab. 4-1: Function blocks of the RMB_TechRegiAdvanced library that is subject to license



The function block MB_RegiAccelerationCompensationType01 has to be called in a motion-synchronous task to ensure a correction calculation of the acceleration. If this is not possible due to the runtime, the "mMotionCyclic()" function of the function block from the Motion-synchronous tasks can be called and the MB_RegiAccelerationCompensationType01 function block can be called in a cyclic task.

4.3 Function blocks to control and optimize the printing process

4.3.1 Introduction and overview

The MB_RegisterControllerAdvType01 function block is responsible for controlling the printing process. The user does not have to parameterize the controllers since the controller parameters calculate autonomously by taking process data (web length, web velocity, dead times...) into account. This ensures that optimum controller parameters are used at any time in the printing process. The user can scale the controller parameters with a damping factor and also select between different control algorithms via different controller modes.

The MB_RegiAccelerationCompensationType01 FB executes a model-based and adaptively learning compensation during the acceleration phase. Thus, occurring dynamic register errors can be compensated and paper waste can be strongly reduced during the acceleration phase. Due to the self-learning functionality, the compensation value adjusts to the product conditions after each acceleration phase.

The MB_RegiGravureRegulateType01 and MB_RegiGravureRegulateType02 function block decouple control movements of register controllers in coupled subsystems. The function blocks support the control on the reference color as well as on the previous color. Regulating movements of register controllers located in the front are fed forward on the following printing units to decouple these printing units from the regulation.

RMB_TechRegiAdvanced

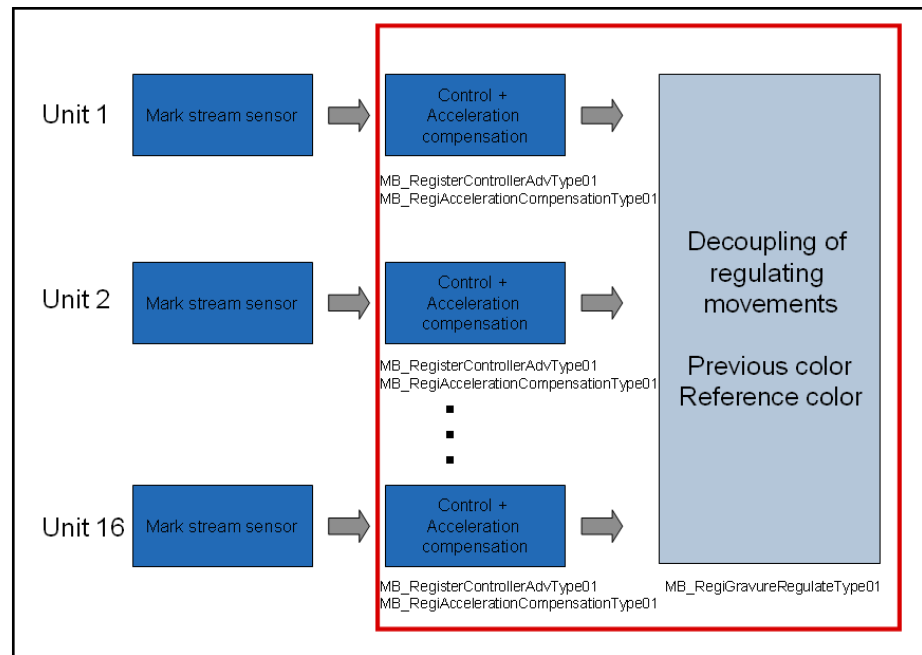


Fig. 4-4: Using control function blocks

The figure shows the use of the control function blocks. The measured values of the mark stream sensor determined by the function blocks of the RMB_TechRegiExtended library are provided to the control. The calculated feedforward control values of the velocity compensation and the control variable of the control are transferred to a decoupling network which calculates the regulations for the drive systems and decouples the regulations for following printing units.

4.3.2 MB_RegisterControllerAdvType01

Short description

The register controller is designed for applications with clamping units (e.g. for gravure printing) and without clamping units. Therefore, different controller parameterizations can be selected. These are adapted according to the velocity. Thus, no controller parameters are applied at the function block. If the controller parameterization of the symmetrical optimum is applied, a prefilter is used for the command value. Furthermore, the control variable of the register controller is adjusted in a ramp shape over a cylinder revolution. This means that the calculated control variable is not regulated in jerks but during the period required for a cylinder revolution. The register controller is divided into the following blocks:

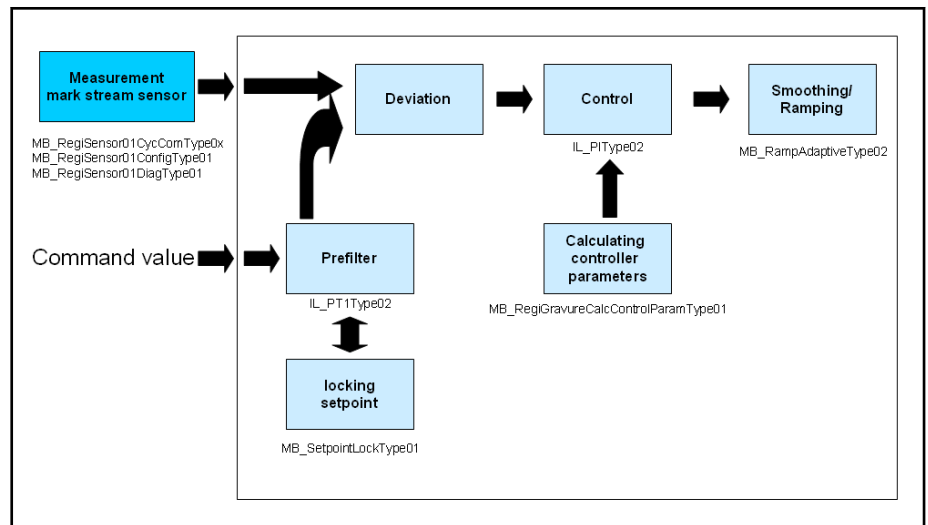


Fig. 4-5: Structure of the MB_RegisterControllerAdvType01 FB

MB_RegiGravureCalcControlParamType01

- Calculation of the controller parameters according to physical parameters.

IL_PT1Type02

- Calculation of the command value filter during controller parameterization according to the symmetrical optimum

IL_PIType02

- PI controller
- Dead zone
- Limitation

MB_RampAdaptiveType02

- Ramp-shaped modification of the control variable of the register controller over a cylinder revolution.

MB_RegiSetpointLockType01

- It applies the actual value as new command value

The "MB_RegisterControllerAdvType01" register controller FB supports the following functionality:

- Calculation of a controller control variable by a PI control
- Preset feature
- Pause feature
- Lock setpoint Feature
- Min./Max. limitation of the controller control variable
- Linear ramp-shaped modification of the controller control variable
- Dead zone
- Different controller parameterizations
- Automatic adaptation of the controller parameters at process data
- Calculation of a prefilter for a command value change
- Damping of optimum controller parameters
- Switching to command value response

RMB_TechRegiAdvanced

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V11	RMB_TechRegiAdvanced

Tab. 4-2: Reference table of the MB_RegisterControllerAdvType01 FB

Interface description

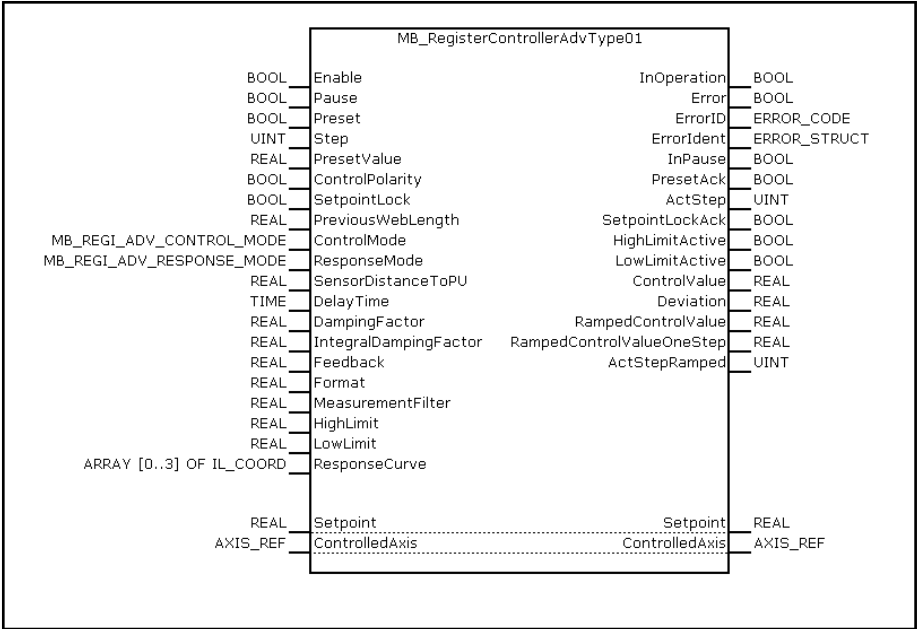


Fig. 4-6: MB_RegisterControllerAdvType01 FB

I/O type	Name	Data type	Description
VAR_IN_ OUT CONST	ControlledAxis	AXIS_REF	Reference to the controlled axis
VAR_IN_ OUT	Setpoint	REAL	Command value It is compared to "Feedback" (actual value) and can be set to the next "Feedback" value (lock setpoint Feature).

I/O type	Name	Data type	Description
VAR_IN-PUT	Enable	BOOL	If TRUE, the controller parameters are calculated first and "ControlValue" is either set to the preset value of "PresetValue" (if there is a rising edge at "Preset") or to the calculated value. "RampedControlValue" is set to the value of "ControlValue".
	Pause	BOOL	The "Pause" input is only evaluated if the "Enable" input is active. In this case, no new controller parameters are calculated, the value "ControlValue" is not changed and the ramp-shaped modification of this value is stopped which means that "RampedControlValue" remains unchanged. "ActStep" is changed, but "ActStepRamped" is not changed.
	Preset	BOOL	The "Preset" input is only evaluated if the "Enable" input is active. A positive edge of this input sets the I value of the controller to "PresetValue". That means that the controller output is set to the value that can be parameterized in "PresetValue".
	Step	UINT	Controller cycle step. If it is increased, a new measured value of the sensor is present. In this case, controller parameters are calculated again and a new "ControlValue" is set at the controller output. This new value is modified in a ramp shape over the new time interval.
	PresetValue	REAL	The "ControlValue" output is set to this value at a rising edge at "Preset"
	ControlPolarity	BOOL	The polarity of "ControlValue" is inverted as long as this value is "TRUE".
	SetpointLock	BOOL	If the input is TRUE, the next actual value (change of "Step") is used as new command value for the register controller.
	PreviousWebLength	REAL	It indicates the web length with regard to the printing unit located in front in [mm].
	ControlMode	MB_REGI_ADV_CONTROL_MODE	It indicates the respective algorithm to calculate the control parameters.
	ResponseMode	MB_REGI_ADV_RESPONSE_MODE	It switches between the command value response and the disturbance response
	SensorDistance-ToPU	REAL	Distance from the sensor to the printing unit [mm]
	DelayTime	REAL	Constant dead time occurring in the control loop
	DampingFactor	REAL	It indicates how the optimally calculated controller parameters should be damped
	IntegralDamping-Factor	REAL	Specifies, to what degree the integral action time (T_N) is to be damped
	Feedback	REAL	The actual measured value
	Format	REAL	Printing cylinder format in [mm]
	MeasurementFilter	REAL	Number of measured values to be filtered
	HighLimit	REAL	Upper limit for "ControlValue". If "ControlValue" exceeds this value, it is limited to this value.
	LowLimit	REAL	Lower limit for "ControlValue". If "ControlValue" falls below this value, it is limited to this value.
	ResponseCurve	ARRAY[0..3] OF IL_COORD	Command value curve for the proportional gain

RMB_TechRegiAdvanced

I/O type	Name	Data type	Description
VAR_OUTPUT	InOperation	BOOL	If TRUE, the PI controller works and the controller outputs "ControlValue" and "RampedControlValue" are valid.
	Error	BOOL	Indicates that an error occurred in the function block.
	ErrorID	ERROR_CODE	Error group
	ErrorIdent	ERROR_STRUCT	Error structure with further subdivision of the error.
	InPause	BOOL	It indicates that the "Pause" input is active and that the values in "ControlValue" and "RampedControlValue" do not change.
	PresetAck	BOOL	It indicates that "PresetValue" was applied.
	ActStep	UINT	If a new "ControlValue" was calculated, "Step" value refers to the "Step" input.
	SetpointLockAck	BOOL	New command value was applied
	HighLimitActive	BOOL	"ControlValue" reached upper limit
	LowLimitActive	BOOL	"ControlValue" reached lower limit
	ControlValue	REAL	The calculated controller output value of the PI controller
	Deviation	REAL	Register deviation = Command value - actual value
	RampedControlValue	REAL	The currently modified value of the register controller in ramp shape.
	RampedControlValueOneStep	REAL	It only returns the change of "RampedControlValue" (differentiated) since the last PLC cycle. This is the interface to the decoupling network.
	ActStepRamped	UINT	"Step" value if a new "RampedControlValueOneStep" was calculated. This value does not depend on "Step", since the ramp-shaped value can change in each cycle.

Tab. 4-3: I/O interface of the MB_RegisterControllerAdvType01 FB

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
Pause	BOOL			FALSE	Continuous
Preset	BOOL			FALSE	Continuous
Step	UINT	0	65335	0	Continuous
PresetValue	REAL	n. def	n. def	0.0	Rising edge at "Preset"
ControlPolarity	BOOL			FALSE	Change at "Step"
SetpointLock	BOOL			FALSE	Change at "Step"
PreviousWebLength	REAL	> 0.0	n. def	0.0	Rising edge at "Enable"
ControlMode	MB_REGI_ADV_CONTROL_MODE	1	4	1	Rising edge at "Enable"
ResponseMode	MB_REGI_ADV_RESPONSE_MODE	1	2	1	Continuous

Name	Type	Min. value	Max. value	Default value	Effective
SensorDistance-ToPU	REAL	0	n. def	400	Rising edge at Enable
DelayTime:	TIME	0	n. def	T#60ms	Rising edge at Enable
DampingFactor	REAL	0	200	100	Change at "Step"
IntegralDampingFactor	REAL	0	1000	100	Change at "Step"
Feedback	REAL	n. def	n. def	0.0	Change at "Step"
Format	REAL	> 0		0.0	Rising edge at Enable
MeasurementFilter	REAL	0	n. def	0	Change at "Step"
HighLimit	REAL	LowLimit	n. def	20000000	Change at "Step"
LowLimit	REAL	n. def	HighLimit	-20000000	Change at "Step"
ResponseCurve	ARRAY[0..3] OF IL_COORD	n. def	n. def	0.0	Rising edge at "Enable"
Setpoint	REAL	n. def	n. def	0.0	Change at "Step"

Tab. 4-4: Overview on min./max. values and initial values of the inputs

Enumeration type

The enumeration type MB_REGI_ADV_CONTROL_MODE can assume the following values:

Element	Value	Description
CONTROL_MODE_CHR	1	CHR method
CONTROL_MODE_AMPLITUDE_OPTIMUM	2	Amplitude optimum
CONTROL_MODE_SYMMETRICAL_OPTIMUM	3	Symmetrical optimum
CONTROL_MODE_WOK_APERIODIC_FAIL	4	Root locus curve method
CONTROL_MODE_LOOP_NO_LAG	5	Control without clamping unit
CONTROL_MODE_CHR_WITH_ADD_INTEGRAL_ADAPTION	6	CHR with automatic adjustment of the P- and I-ration in dependency of the DampingFactor
CONTROL_MODE_LOOP_NO_LAG_WITH_ADD_INTEGRAL_ADAPTION	7	Control without clamping unit with automatic adjustment of the P- and I-value in dependency of the DampingFactor

Tab. 4-5: Enumeration type MB_REGI_ADV_CONTROL_MODE

The enumeration type MB_REGI_ADV_RESPONSE_MODE can assume the following values:

Element	Value	Description
RESPONSE_MODE_COMMAND	1	Command value response during setup
RESPONSE_MODE_PRODUCTION	2	Disturbance response in the production

Tab. 4-6: Enumeration type MB_REGI_ADV_RESPONSE_MODE

RMB_TechRegiAdvanced

Function description

Control algorithm

The control algorithm is represented in the following diagram:

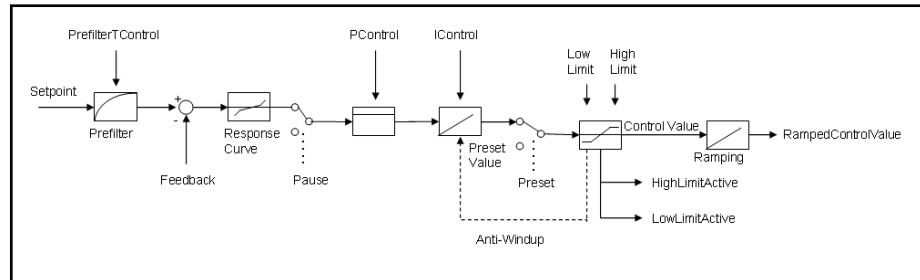


Fig. 4-7: Control algorithm MB_RegisterControllerAdvType01

The input values of the prefilter and the P- and I-input variables of the register controller are calculated in the MB_RegiGravureCalcControlParamType01 FB. Therefore, different parameterization modes can be selected. The controller parameters are calculated again for each change at the "Step" input according to the physical parameters. The register controller and the prefilter also calculate the output variables only again if the "Step" input changes. It has to be considered that the MB_RampAdaptiveType02 FB calculates a new "RampedControlValue" in each cycle. It modifies the control variable of the register controller in a ramp-shape over a cylinder revolution. Thus, the "Step" input and the "ActStepRamped" output change in different cycles. The prefilter is only used if the symmetrical optimum was selected as control mode to impede or minimize overshooting in case of a command value modification.

The difference signal from the filtered "Feedback" input and the "Setpoint" is scaled by a characteristic curve given in "ResponseCurve". The characteristic curve is point-symmetrical to the origin. The x-values of the characteristic curve correspond to the differential signal. The y-values correspond to the scaled difference signal. Up to four points can be specified. The sections between the points are linearly interpolated. If "ResponseCurve" is not applied, there is no scaling.

Initialization

During initialization, the inputs "Setpoint" and "Feedback" have to have valid data. The "ControlValue" is written to the "RampedControlValue" at the output of the PI controller. "RampedControlValue" is only modified in a ramp shape after the second change at the "Step" input, since the period of the cylinder revolution was unknown till then. This means that the first "ControlValue" is not modified in a ramp shape at the first change at the "Step" input.

PresetValue

For a trouble-free start of the controller output "ControlValue", the "Preset" input has to be set if "Enable" is activated.

During this time, the inputs "Setpoint" and "Feedback" have to have valid data.

Checking upper/lower limit values is also still active to ensure that the "ControlValue" is limited if the preset value ("PresetValue") is outside the limits of "HighLimit" or "LowLimit".

"PresetAck" becomes active if the value at "PresetValue" was applied. "Preset" has priority over "Pause". The value in "PresetValue" is applied at a rising edge at "Preset" if "Enable" is set. A completed value transfer is displayed at the "PresetAck" output.

Pause

The "Pause" input is only evaluated if the "Enable" input is active. In this case, the control deviation ("Setpoint" – "Feedback") is set to 0. This leads to the "ControlValue" being fixed. It can be used to keep the output constant stable. The "ActStep" output is updated if the "Pause" input is activated. The "ActStepRamped" does not change. As long as "Pause" is active, the "In-Pause" output is set. Furthermore, no new controller parameters are calculated and the value at the "RampedControlValue" output is also frozen.

Lock Setpoint

The current "Feedback" value can be applied as command value with the "SetpointLock" feature. If the input becomes "TRUE", the value of "Setpoint" is overwritten with the value of "Feedback" that is present when "Step" is changed next time.

This feature allows the user to switch the controller to "Pause" mode, to jog the system manually to the register and to apply the next "Feedback" value as new "Setpoint".

Dead zone

The deviation calculated from the difference between "Setpoint" and "Feedback" is scaled via "ResponseCurve". That means that the register controller is "frozen" in case of a small register deviation or it takes back the control dynamics. The latter is useful to get "smooth" control characteristics in a steady state and to settle greater deviations fast.

Normal mode

The controller only calculates a new "ControlValue" if the value changes at the "Step" input. This value is modified in a ramp shape over a period of the last cylinder revolution. That means that the value "RampedControlValue" changes in each cycle.

If "ControlValue" exceeds or falls below the value of "HighLimit" or "LowLimit", "ControlValue" is limited to one of these values. If the controller has an integration part, an anti windup becomes active. That means the integral part does not continue to integrate. In this case, the "HighLimitActive" and "LowLimitActive" outputs are set.

ResponseMode

The default value is set to the command value response (RESPONSE_MODE_COMMAND). This mode should be used when switching on the controller and jogging the command value. At the end of the setup phase, it should be switched to RESPONSE_MODE_PRODUCTION to achieve better control dynamics. In RESPONSE_MODE_PRODUCTION, the mode for the disturbance response set in "ControlMode" is selected. For RESPONSE_MODE_COMMAND, it is only differentiated between the CONTROL_MODE_CHR, CONTROL_MODE_LOOP_NO_LAG, CONTROL_MODE_CHR_WITH_ADD_INTEGRAL_ADAPTION and CONTROL_MODE_LOOP_NO_LAG_WITH_ADD_INTEGRAL_ADAPTION modes when calculating the controller parameters. If another mode is selected, the controller parameters are calculated after CONTROL_MODE_CHR.

Control Mode / Calculating the controller parameters

For the controller parameter calculation, the selected mode, velocity, web length, plate cylinder circumference, distance from the sensor to the printing unit, the measured value filter and a constant dead time have to be considered due to transmission and computing times. The default value for constant dead times of 60 ms is an empirical value that should not be changed at first. The distance of the sensor to the printing unit causes an additional delay and should thus be entered. Moreover, the number of measured values used for filtered is to be entered.

RMB_TechRegiAdvanced

Control Mode	Printing process	Influencing the controller parameters
CONTROL_MODE_CHR	With clamping unit	DampingFactor, influences P-value IntegralDampingFactor, influences I-ratio
CONTROL_MODE_AMPLITUDE_OPTIMUM	With clamping unit	DampingFactor, influences P-value IntegralDampingFactor, influences I-ratio
CONTROL_MODE_SYMMETRICAL_OPTIMUM	With clamping unit	DampingFactor, influences P-value IntegralDampingFactor, influences I-ratio
CONTROL_MODE_WOK_APERIODIC_FAIL	With clamping unit	DampingFactor, influences P-value IntegralDampingFactor, influences I-ratio
CONTROL_MODE_LOOP_NO_LAG	Without clamping unit	DampingFactor, influences P-value IntegralDampingFactor, influences I-ratio
CONTROL_MODE_CHR_WITH_ADD_INTEGRAL_ADAPTION	With clamping unit	DampingFactor, influences P-value and I-value IntegralDampingFactor has no effect
CONTROL_MODE_LOOP_NO_LAG_WITH_ADD_INTEGRAL_ADAPTION	Without clamping unit	DampingFactor, influences P-value and I-value IntegralDampingFactor has no effect

Tab. 4-7: Using Control Modes

In the control modes, it is differentiated between the printing process. If the printing process has a clamping unit (e.g. gravure printing), an adjustment does not become active immediately and special methods for calculating the controller parameters are used (modes 1,2,3,4,6). If the printing process does not have a clamping unit, an adjustment becomes active almost immediately. This is also taken into consideration when calculating the controller parameters (modes 5,7).

The controller parameters are automatically calculated depending on the control mode. The user can influence these parameters by means of the DampingFactor or the IntegralDampingFactor. The DampingFactor only has an effect on the P-value and the IntegralDampingFactor on the I-ratio during the first five modes. Example: If the DampingFactor is 50%, the automatically calculated P-value is reduced by half.

Exceptions: "CONTROL_MODE_CHR_WITH_ADD_INTEGRAL_ADAPTION" and "CONTROL_MODE_LOOP_NO_LAG_WITH_ADD_INTEGRAL_ADAPTION". The IntegralDampingFactor does not have any effect. If the DampingFactor is changed, the P- and I-ratio is automatically changed, too. Example: If the DampingFactor < 100 %, the P-value is reduced and the I-ratio is increased accordingly, as a lower control loop dynamics is to be achieved.

If the DampingFactor is set to 0 for the modes 1-5, the P-controller is enabled. If the IntegralDampingFactor is set to 0 for the modes 1-5, the I-controller is enabled. If the DampingFactor is set to 0 for the modes 6 or 7, the P-controller and the I-controller are disabled but the IntegralDampingFactor does not have any effect in these modes.



It is recommended to use the automatically calculated control parameters. I.e., the DampingFactor and the IntegralDampingFactor are to be set to 100 %. If the control parameters have to be manually adjusted due to the printing process, one of the following modes "CONTROL_MODE_CHR_WITH_ADD_INTEGRAL_ADAPTION" or "CONTROL_MODE_LOOP_NO_LAG_WITH_ADD_INTEGRAL_ADAPTION" are to be used, as the P- and the I-ratio are automatically adjusted in an optimized ratio.

Using output variables

The following figure shows the output variables generated by the controller. The "ControlValue" is the output value of the controller calculated each time after a new measured value is present. It is the already integrated value and does thus not have to be added in the following. "ActStep" is incremented by one with each new measured value. The control variable of the "ControlValue" is modified in a ramp shape over the cylinder revolution period. This value is output with "RampedControlValue". "RampedControlValue" is updated in each computing cycle and the respective "ActStepRamped" is incremented. Furthermore, "RampedControlValueOneStep" is output indicating the change of "RampedControlValue". This value has to be connected to the decoupling network, since this value is integrated into this network. It is also updated in each computing cycle and refers to "ActStepRamped".

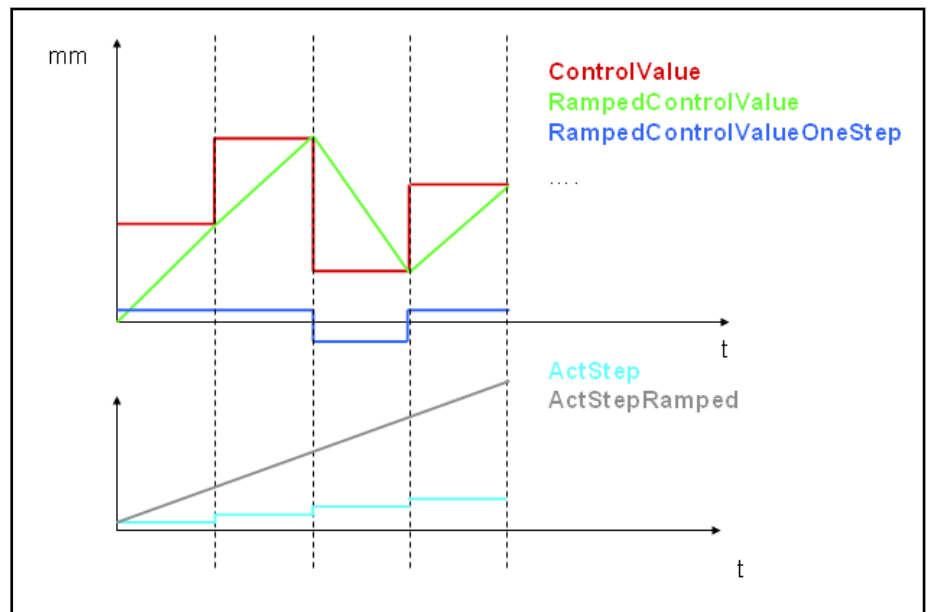


Fig. 4-8: Using controller output variables

HighLimit and LowLimit

This value limits the controller control variable from one step input change to the next change. I.e., if a value of 1 mm is selected as HighLimit, the controller control variable change is limited to a maximum of 1 mm per computing step (one computing step corresponds to a new measured value).

Error handling

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

RMB_TechRegiAdvanced

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000002A	Invalid damping factor
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000002B	Invalid web length
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000002C	Invalid Control mode
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000002D	Invalid velocity
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000000	Invalid plate cylinder format
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000002E	Invalid Response mode
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000002F	Invalid sensor distance
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000030	Invalid delay time
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000045	Invalid integral damping factor
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000046	Invalid measured value filter
INPUT_RANGE_ERROR (16#0006)	16#0000000D	16#00000000	AxisNo of "Master" is not within the valid range

Tab. 4-8: Error codes of the MB_RegisterControllerAdvType01 FB

4.3.3 MB_RegiAccelerationCompensationType01

Short description

The MB_RegiAccelerationCompensationType01 calculates a feedforward control that compensates the register error in the acceleration phase and in the production phase. A model-based compensation can be carried out if the respective machine data and material data are available. An adaptive and self-adjusting compensation is implemented that determines the compensation values from the controller control variable and the velocity. This compensation is optimized in each acceleration phase.

The MB_RegiAccelerationCompensationType01 feedforward control is located in front of a control loop to minimize the register error. Control signals of the controller and the feedforward control are added first and then added to the plant controlled system.

There are two useful reasons to use a feedforward control:

1. A non-stationary register error occurs during the acceleration phase that subsequently does not occur anymore. The controller is designed for stationary web characteristics.
2. The dynamics of the control loop is increased in each operating phase.



The function block MB_RegiAccelerationCompensationType01 has to be called in a motion-synchronous task to ensure a correction calculation of the acceleration. If this is not possible due to the runtime, the "mMotionCyclic()" function of the function block from the motion-synchronous tasks has to be called and the MB_RegiAccelerationCompensationType0 function block can be called in a cyclic task. The call of the MotionTask has to be provided with the following structure: {CyclicTask}.{fbInstanz}.mMotionCyclic(). The elements in the curly brackets have to be replaced by the program name in the cyclic task and the instance name of the function block. Thus, the method call from a program running in the Motion task could be as follows: PLCProg.fbRegiAccelerationCompensationType01.mMotionCyclic(). If this call is not executed, a correct calculation of the acceleration and thus an error-free acceleration compensation cannot be ensured.

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V11	RMB_TechRegiAdvanced

Tab. 4-9: Reference table of the MB_RegiAccelerationCompensationType01 FB

Interface description

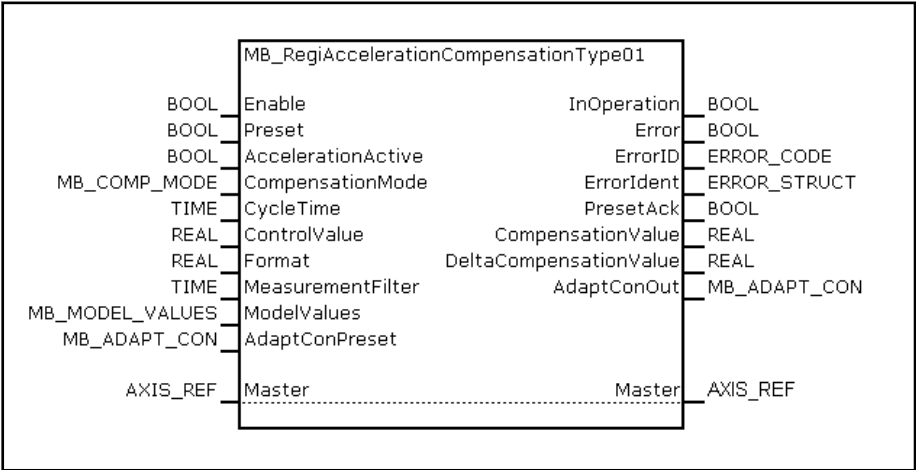


Fig. 4-9: MB_RegiAccelerationCompensationType01 FB

RMB_TechRegiAdvanced

I/O type	Name	Data type	Description
VAR_IN_OUT CONST	Master	AXIS_REF	Reference to the axis to determine velocity and acceleration compensation. Due to accuracy reasons, it is recommended to create a virtual master axis.
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled).
	Preset	BOOL	Preset of adaptive control values.
	AccelerationActive	BOOL	It indicates the function block that the acceleration phase starts or ends.
	CompensationMode	MB_COMP_MODE	Selection between adaptive and/or model-based compensation.
	CycleTime	TIME	Cycle time of the PLC task in which the function block is called to calculate the acceleration.
	ControlValue	REAL	Controller control signal in [mm]
	Format	REAL	Printing cylinder circumference in [mm]
	MeasurementFilter	TIME	Filter for "ControlValue".
	ModelValues	MB_MODEL_VALUES	Physical parameter for model-based compensation.
	AdaptConPreset	MB_ADAPT_CON	Adaptive control signals set at "Preset".
VAR_OUTPUT	InOperation	BOOL	Function block is working.
	Error	BOOL	Processing completed with errors.
	ErrorID	ERROR_CODE	Error group
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics
	PresetAck	BOOL	"Preset" input active
	CompensationValue	DINT	Phase offset (modulo 360°) [degrees]
	DeltaCompensationValue	REAL	Interface to the decoupling network in [mm]. It indicates the differentiated change.
	AdaptConOut	STRUCT	Adaptively improved control signals currently in use.

Tab. 4-10: I/O interface of the MB_RegiAccelerationCompensationType01 FB

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
Preset	BOOL			FALSE	Continuous
AccelerationActive	BOOL			FALSE	Continuous
Compensation-Mode	MB_COMP_MODE	0	3	MODEL_AND_ADAPTIVE=0	Enable rising edge
CycleTime	TIME	> 0 ms	n. def	8 ms	Enable rising edge
ControlValue	REAL	n. def	n. def	0	Continuous
Format	REAL	>0	n. def	0	Enable rising edge

Name	Type	Min. value	Max. value	Default value	Effective
MeasurementValues	TIME	n. def	n. def	0ms	Continuously (when entering 0, the filter is deactivated)
MB_MODEL_VALUES					
L_DK	REAL	≥ 0	n. def	0	Enable rising edge
R_kuehl	REAL	> 0	n. def	1	Enable rising edge
J_kuehl	REAL	≥ 0	n. def	0	Enable rising edge
E	REAL	> 0	n. def	1	Enable rising edge
A	REAL	> 0	n. def	1	Enable rising edge
MB_ADAPT_CON					
CA	REAL	n. def	n. def	0	Preset rising edge
CP	REAL	n. def	n. def	0	Preset rising edge

Tab. 4-11: Overview on min./max. values and initial values of the inputs

Data structures

MB_MODEL_VALUES		
L_DK	REAL	Length between DW_links and cooling roll [m]
R_kuehl	REAL	Cooling roll radius [mm]
J_kuehl	REAL	Rotating inertia of cooling roll [$\text{kg} \cdot \text{m}^2$]
E	REAL	Modulus of elasticity of the printing web [N/m^2]
A	REAL	Cross-sectional area of the printing web [m^2]

Tab. 4-12: MB_MODEL_VALUES data structure

MB_ADAPT_CON		
CA	REAL	Compensation value for the acceleration compensation
CP	REAL	Compensation value for the velocity compensation

Tab. 4-13: MB_ADAPT_CON data structure

Enumeration type

The enumeration type MB_COMP_MODE can assume the following values:

Element	Value	Description
MODEL_AND_ADAPTIVE	0	Model-based and adaptive compensation
ADAPTIVE	1	Only adaptive compensation
MODELBASED	2	Only model-based compensation
ADAPTIVE_WITHOUT_LEARNING	3	The values CA and CP are used, but not always updated after each acceleration phase.

Tab. 4-14: Enumeration type MB_COMP_MODE

Function description

The FB contains the following functionalities:

- A **model-based acceleration compensation** and one

- **adaptive velocity and acceleration compensation.**

This iteratively determines suitable compensation values by evaluating the controller control signal to increase the dynamics of the control loop. The feedforward control is carried out in a linear dependency on acceleration and velocity.

At a rising edge at "Preset" or "Enable", the physical parameters of the physical acceleration compensation MB_MODEL_VALUES are applied and checked for validity. If the printing material has a time-dependent creep and relaxation behavior, a dynamic modulus of elasticity from a fast executed tension test has to be used. If no such value exists, it has to be estimated first. Without specifying the physical parameters, the MB_MODEL_VALUES input is initialized in a way that no model-based compensation takes place.

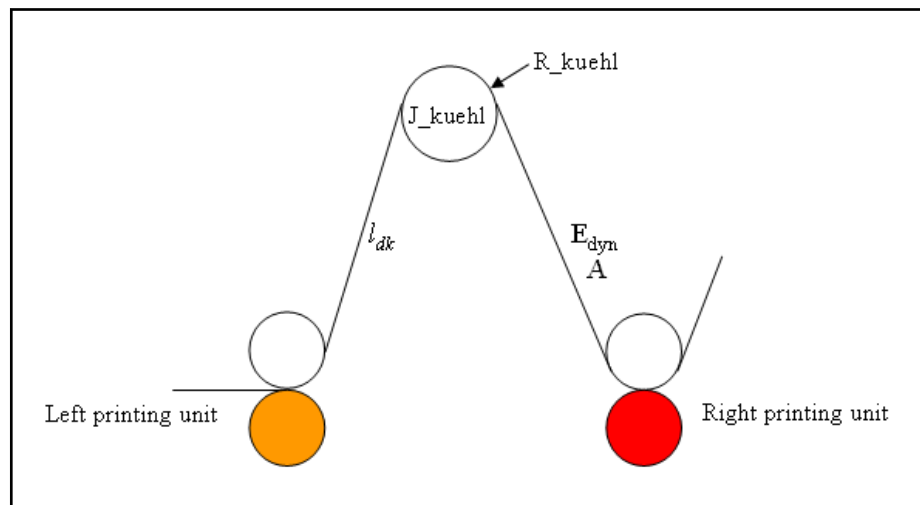


Fig. 4-10: Physical parameters of the model-based compensation

Preset

The adaptive control parameters "AdaptConPreset" are only newly applied at "Preset". This can be useful when exchanging the web.

AccelerationActive

BOOL "AccelerationActive" has to be set to indicate that a new acceleration phase starts. This should be carried out before the acceleration phase is started, since the first controller control variable is applied here. After the completion of the acceleration phase (ideally also a short time span after ~1 s), BOOL is set to FALSE again. When disabling the "AccelerationActive" input, the compensation values (for further accelerations) are updated. As long as "AccelerationActive" is set, the respective compensation values are calculated and output.

The adaptive control values are output as "AdaptConOut". For frequent changes of printing material, it is recommended to save "AdaptConOut" in a receipt file. In case of material re-usage, the compensation parameters can be used again.

MB_REGI_ADV_CONTROL_MODE

It is recommended to use the functionality of the model-based compensation as well as the functionality of the adaptive compensation. Therefore, the "CompensationMode" input is initialized on MODEL_AND_ADAPTIVE. If only the model-based compensation should be used, the input is to be set to MODELBASED. If a completely adaptive compensation should be used, select ADAPTIVE. If the determined and entered values for CA and CP should be used, but learning should be switched off, the ADAPTIVE_WITHOUT_LEARNING mode has to be selected.

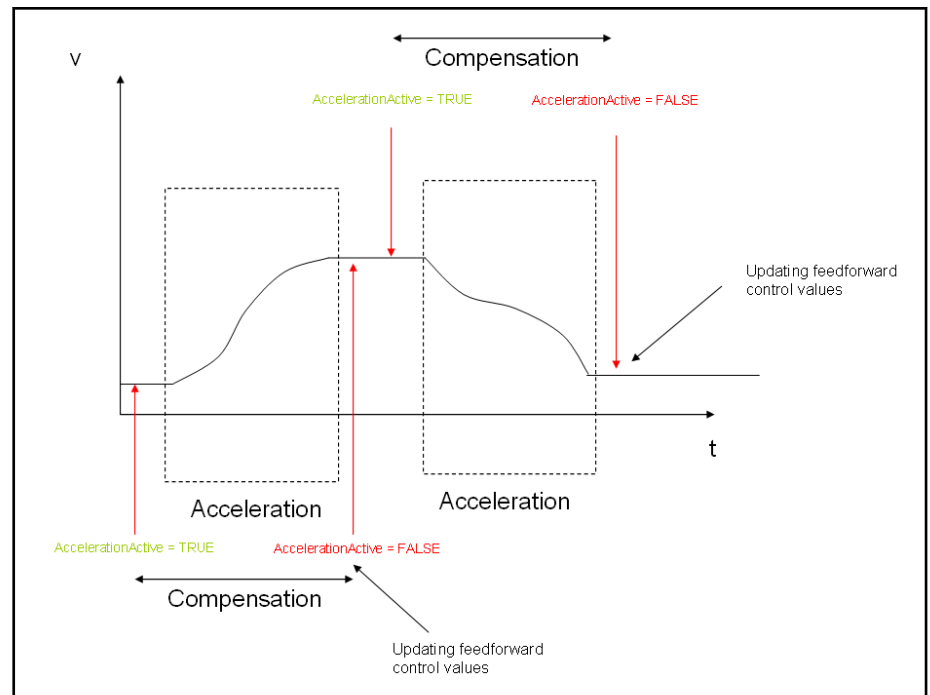


Fig. 4-11: Operating procedure of the MB_RegiAccelerationCompensation-Type01

The operating procedure of the MB_RegiAccelerationCompensationType01 is shown above. As long as AccelerationActive = FALSE, the FB is in its idle phase (Enable = TRUE). If a rising edge is set at "AccelerationActive", the current controller control variable is applied and, when changing acceleration and velocity, the respective compensation is carried out. Therefore, the value is specified in mm at the "DeltaCompensationValue" output. This value is transferred to the decoupling network in which the value is integrated and decoupled. "AccelerationActive" is set to FALSE at the end of the acceleration phase. Thus, the compensation mode is exited and the acceleration phase is evaluated. Values are updated for CA and CP and it will be gone in idle phase again until there is a rising edge at "AccelerationActive".



Ideally, the MB_ReachVelocityType01 FB is used for acceleration. Here, the "Active" output can be connected to "AccelerationActive". The "Active" output is set 500 ms before the acceleration phase starts and is reset 500 ms after the end of the acceleration phase.

Error handling

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000031	Invalid length between printing unit and cooling roll
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000032	Invalid radius of the cooling roll
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000033	Invalid inertia of the cooling roll

RMB_TechRegiAdvanced

ErrorID	Additional 1	Additional 2	Description
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000034	Invalid modulus of elasticity
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000035	Invalid area of the web
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000000	Invalid plate cylinder format
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000036	Invalid cycle time
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000037	Invalid compensation mode
INPUT_RANGE_ERROR (16#0006)	16#0000000D	16#00000000	AxisNo of "Master" is not within the valid range

Tab. 4-15: Error codes of the MB_RegiAccelerationCompensationType01 FB

4.3.4 MB_RegiGravureRegulateType01

Short description

The decoupling network is design for applications with clamping units at the printing units. It is used to decouple regulating movements in coupled sub-systems. Thus, the network can also be used in other applications than gravure printing machines in which clamping units exist causing coupled subsystems. There are different algorithms available describing how control variables of the register controller are cascaded.

The decoupling network is used as follows:

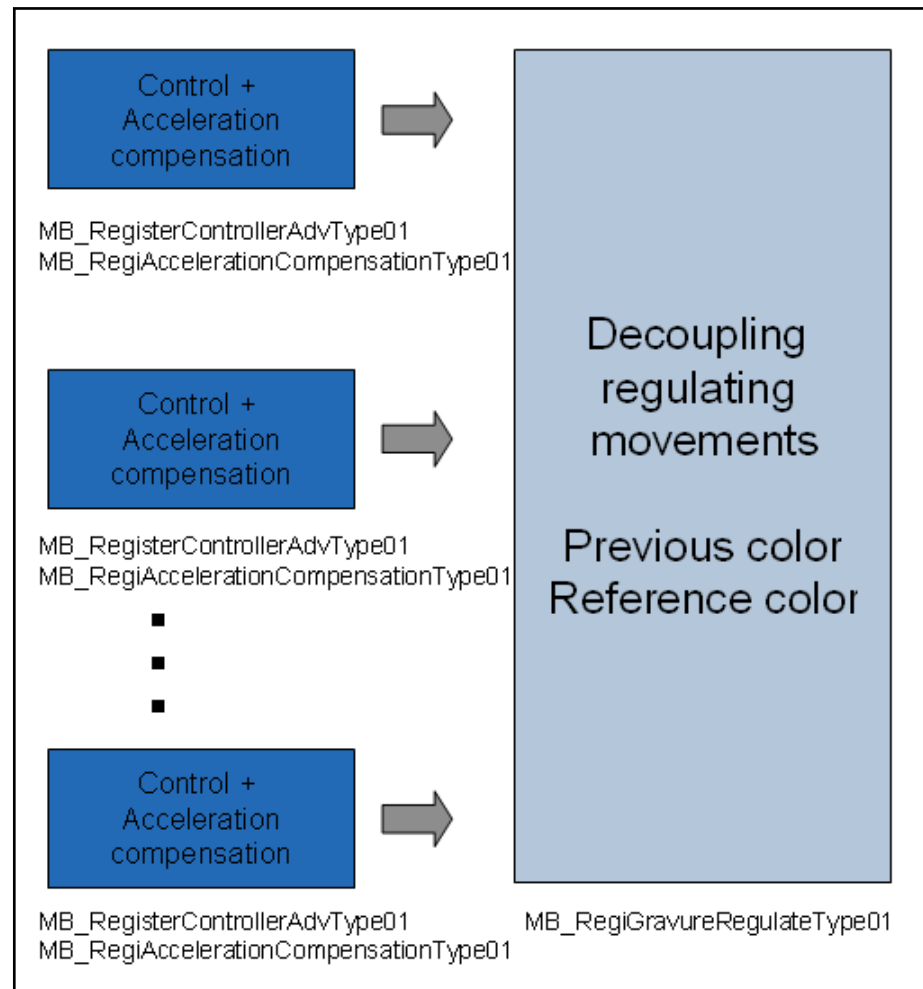


Fig. 4-12: Using the decoupling network

The control variables of the register controllers are cascaded in the decoupling network according to the algorithm selected. A phase offset - that is output - is calculated for each printing unit.

The decoupling network MB_RegiGravureRegulateType01 supports the following functionality:

- Dynamic cascading of the control variables of a register controller or of the acceleration compensation
- The dynamic downstream variants are available for the previous color and the reference color
- Preset Feature for preregistration
- Support of one manual mode per printing unit
- Different regulating velocities per mode and per printing unit for manual regulation or use of an external register controller
- The use of the Rexroth RegisterControl controller as well as of an external register controller is possible
- Up to 16 printing units (axes) are possible

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V11	RMB_TechRegiAdvanced

Tab. 4-16: Reference table of the MB_RegiGravureRegulateType01 FB

Interface description

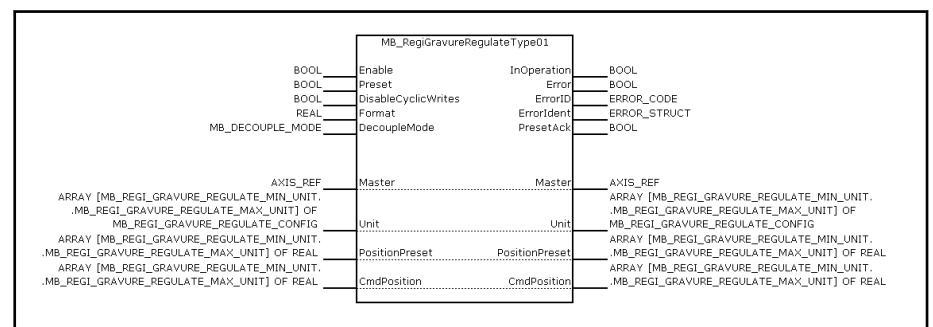


Fig. 4-13: MB_RegiGravureRegulateType01 FB

RMB_TechRegiAdvanced

I/O type	Name	Data type	Description
VAR_IN_OUT CONST	Master	AXIS_REF	Reference to the master axis
VAR_IN_OUT	Unit	ARRAY [MB_REGI_GRA- VURE_REGU- LATE_MIN_UNIT .. MB_REGI_GRAVURE_REG- ULATE_MAX_UNIT] OF MB_REGI_GRAVURE_REG- ULATE_CONFIG	The array contains a data structure for each printing unit. In this data structure, the axis of the respective printing unit is created as well as all other required parameters of a printing unit.
	PositionPreset	ARRAY [MB_REGI_GRA- VURE_REGU- LATE_MIN_UNIT .. MB_REGI_GRAVURE_REG- ULATE_MAX_UNIT] OF RE- AL	Array with the preset values for all printing units in [°].
	CmdPosition	ARRAY [MB_REGI_GRA- VURE_REGU- LATE_MIN_UNIT .. MB_REGI_GRAVURE_REG- ULATE_MAX_UNIT] OF RE- AL	Array with the offset values for all printing units in [°]. If DisableCyclicWrites = TRUE, this value must be added manually by the user.
VAR_INPUT	Enable	BOOL	Processing of function block enabled
	Preset	BOOL	The array "CmdPosition" is reset and the values from "PositionPreset" are applied. This value will also still be written to the drive in future if DisableCyclicWrites = FALSE.
	DisableCyclicWrites	BOOL	If the input is set, the cyclic data is not written anymore to the axes, but only output in "CmdPosition".
	Format	REAL	Printing cylinder format [mm]
	Decouple-Mode	MB_DECOUPLE_MODE	It indicates the respective algorithm to cascade the controller control variables.
VAR_OUT- PUT	InOperation	BOOL	The decoupling function block operates and the outputs are valid.
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Error group
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics
	PresetAck	BOOL	Values from "PositionPreset" were written in "CmdPosition" or to the axis.

Tab. 4-17: I/O interface of the MB_RegiGravureRegulateType01 FB

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
Preset	BOOL			FALSE	Continuous
DisableCyclic-Writes	BOOL			FALSE	At a rising edge at "Enable"
Format	REAL	> 0	n. def	1.0	At a rising edge at "Enable"
DecoupleMode	MB_DECOUPLE_MODE	0	1	COLOR_TO_COLOR := 0	At a rising edge at "Enable"
MB_REGI_GRAVURE_REGULATE_CONFIG					
Slave	AXIS_REF	NoAxis			At a rising edge at "Enable" Rising edge Enable
Active	BOOL			FALSE	Continuous
Inc	BOOL			FALSE	Continuous
Dec	BOOL			FALSE	Continuous
RegiGravure-Control	BOOL			FALSE	Continuous
RampedControl-ValueOneStep	REAL	n. def	n. def	0.0	Continuous
ManualMode	BOOL			FALSE	Continuous
RedSpeed	BOOL			FALSE	Continuous
NormalSpeed	REAL	0.0	n. def	1	Continuous
ReducedSpeed	REAL	0.0	n. def	0.5	Continuous
WebLength	REAL	1.0	n. def	10000	At a rising edge at "Enable"

Tab. 4-18: Overview on min./max. values and initial values of the inputs

Data structures

MB_REGI_GRAVURE_REGULATE_CONFIG		
Slave	AXIS_REF	Axis of the printing unit
Active	BOOL	Printing unit is active (printing).
Inc	BOOL	Jogging, incrementing the position (Manual mode or external register controller).
Dec	BOOL	Jogging; decrementing the position (Manual mode or external register controller).
RegiGravureControl	BOOL	The register controller MB_RegiControllerAdvType01 is used if the "Regi-GravureControl " is enabled. Thus, in Automatic and Manual mode, the RampedControlValueOneStep input is applied indicating the register regulation in [mm].
RampedControlValueOneStep	REAL	The differentiated control variable of the register controller in [mm].
ManualMode	BOOL	It activates Manual mode. Thus, the control variable is not decoupled.
RedSpeed	BOOL	Selection of the regulating velocity

RMB_TechRegiAdvanced

MB_REGI_GRAVURE_REGULATE_CONFIG		
NormalSpeed	REAL	Regulating velocity [1/s] in the mode RedSpeed = FALSE.
ReducedSpeed	REAL	Regulating velocity [1/s] in the mode RedSpeed = TRUE.
WebLength	REAL	Web length from the active printing unit located in the front to the own printing unit [mm].

Tab. 4-19: MB_REGI_GRAVURE_REGULATE_CONFIG data structures

Enumeration type

The enumeration type MB_DECOUPLE_MODE can assume the following values:

Element	Value	Description
COLOR_TO_COLOR	0	Previous color control
REFERENCE_COLOR	1	Reference color control

Tab. 4-20: Enumeration type MB_DECOUPLE_MODE

Definition of the defines:

MB_REGI_GRAVURE_REGULATE_MIN_UNIT := 1;

MB_REGI_GRAVURE_REGULATE_MAX_UNIT := 16;

Function description

The MB_RegiGravureRegulateType01 FB is used as an interface between the register controllers and the drives. The register controller used can be an external register controller, but should ideally be the register controller MB_RegisterControllerAdvType01 optimized for rotogravure printing. The control variables are cascaded as feedforward control to the following printing units with different algorithms according to the control strategy.

Initialization

The FB starts with the values in "CmdPosition" at a rising edge at "Enable". This value corresponds to the phase offset for each printing unit.

Preset

To reset the values for "CmdPosition", "Preset" must be set to "TRUE". Thus, the "PositionPreset" values are copied to the "CmdPosition" array.

DisableCyclicWrites

The "DisableCyclicWrites" input determines whether the values calculated by the function block are directly written on the axes (FALSE) or only available at the FB output in "CmdPosition" (TRUE). If the AxisInterface or further functionalities that write on this axis are used, "DisableCyclicWrites" must be enabled. Subsequently, the user can autonomously write the value calculated in "CmdPosition" to the axis (for example with the AxisInterface via PhaseOffset in SyncMode). This allows to autonomously copy the phase offset to the respective parameter by setting "DisableCyclicWrites" to TRUE. Subsequently, only the output array "CmdPosition" is to be used and further modified.

DecoupleMode

It can be chosen between two different cascading algorithms. It is recommended to use the previous color control.

- COLOR_TO_COLOR

In this mode, the correction values are not statically, but dynamically cascaded. That means that the correction values of a register controller are cascaded to the following printing units via several timing elements.

Thus, an ideal decoupling of the control variable of a previous register controller can be carried out. Therefore, the web length between the respective printing units has to be specified to calculate the time constants for the dynamic actuators.

The following figure shows the cascading of the controller control variable:

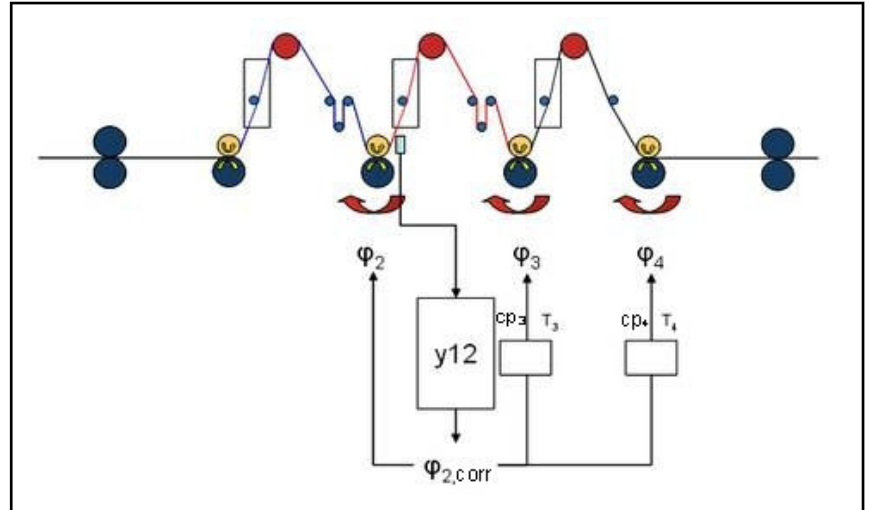


Fig. 4-14: Dynamic cascading

The white boxes stand for several dynamic timing elements that completely decouple the regulation of the register controller 2 for the register controllers 3 and 4.

This mode can only be used if the sensor compares the marks of the printing unit located in the front with the current printing unit and subsequently controls the register.

- **REFERENCE_COLOR**

This mode can only be used if the sensor compares the marks of the own printing unit with a reference color (mostly printing unit 1).

This mode basically acts like COLOR_TO_COLOR, but is provided with a different combination of dynamic elements. The acceleration compensation can only be used for the COLOR_TO_COLOR mode. Due to this fact, this mode should only be used if no high accuracy is required in the acceleration phase.

Functionalities that can be set for the respective printing unit:

- **RegiGravureControl:**

If RegiGravureControl = TRUE, the RampedControlValueOneStep input is used as controller control variable in Automatic and Manual mode. This is the interface to the internal rotogravure printing controller MB_RegisterControllerAdvType01.

If RegiGravureControl = FALSE, only the "Inc" and "Dec" inputs are used as controller control variables in Manual as well as in Automatic mode (interface to an external register controller).

- **ManualMode**

If ManualMode = TRUE, the input variables are not decoupled for subsequent printing units.

If ManualMode = FALSE, the controller control variables are decoupled for subsequent printing units according to the selected control strategy.

- **RedSpeed**

RMB_TechRegiAdvanced

Two different regulating velocities can be entered for each printing unit. If RedSpeed = TRUE, "ReducedSpeed" is used as regulating velocity, otherwise "NormalSpeed". This only applies if the jog inputs "Inc" and "Dec" are used which is the case when RegiGravureControl = FALSE.

- **Active**
Each printing unit participating in the printing process has to be enabled with "Active".
- **WebLength**
Web length between the printing unit located in the front and the printing unit to be controlled. This web length is used to determine the dynamic time constants.

Error handling

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000000	Invalid printing cylinder format
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000003A	Invalid Decouple mode
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000038	Invalid NormalSpeed
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000039	Invalid ReducedSpeed
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000002B	Invalid web length
RESOURCE_ERROR (16#0003)	16#00000001	16#00000000	No power is added to the drive specified in "Slave". A register control is thus not possible.
INPUT_RANGE_ERROR (16#0006)	16#0000000D	16#00000000	AxisNo of "Axis" is not in the valid range

Tab. 4-21: Error codes of the MB_RegiGravureRegulateType01 FB

Furthermore, errors can occur in the libraries RIL_LoopControl and MS_TechBase.

4.3.5 MB_RegiGravureRegulateType02

Short description

The function block contains the complete functionality of the MB_RegiGravureRegulateType01 function block. Due to this reason it is recommended to refer to the documentation of the MB_RegiGravureRegulateType01 function block.

Only the additional functions of the MB_RegiGravureRegulateType02 function block are explained:

- Executing a preset for every printing unit (use case: preregistration).
- Ignoring the pre-control values of upstream printing units (use case: processing unit with upstream decoupling to the printing process).

The extensions refer to the MB_REGI_GRAVURE_REGULATE_CONFIG data structure. Additional variables triggering actions are inserted in the MB_REGI_GRAVURE_REGULATE_CONFIG_TYPE02 data structure.

Target system	Library
Rexroth RegisterControl MLC L65/L45 from V13	RMB_TechRegiAdvanced

Tab. 4-22: Reference table of the MS_RegiGravureRegulateType02 function block

Interface description

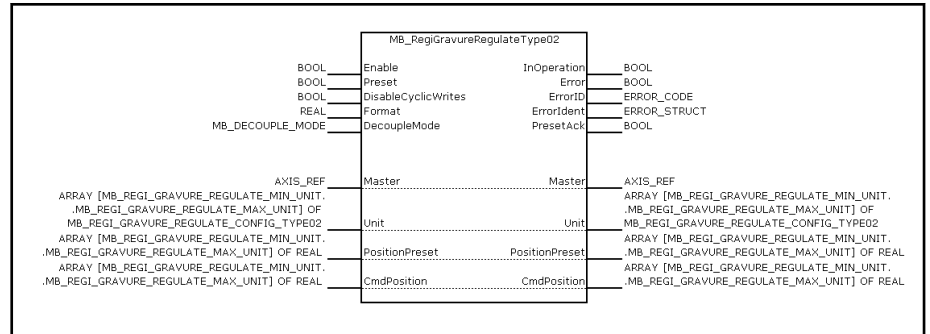


Fig. 4-15: MB_RegiGravureRegulateType02 function block

I/O type	Name	Data type	Explanation
VAR_IN_OUT CONST	Master	AXIS_REF	Reference to the master axis
VAR_IN_OUT	Unit	ARRAY [MB_REGI_GRAVURE_REGULATE_MIN_UNIT .. MB_REGI_GRAVURE_REGULATE_MAX_UNIT] OF MB_REGI_GRAVURE_REGULATE_CONFIG_TYPE02	The array contains a data structure for each printing unit. In this data structure, the axis of the respective printing unit is created as well as all other required parameters of a printing unit.
	PositionPreset	ARRAY [MB_REGI_GRAVURE_REGULATE_MIN_UNIT .. MB_REGI_GRAVURE_REGULATE_MAX_UNIT] OF REAL	Array with the preset values for all printing units in [°].
	CmdPosition	ARRAY [MB_REGI_GRAVURE_REGULATE_MIN_UNIT .. MB_REGI_GRAVURE_REGULATE_MAX_UNIT] OF REAL	Array with the offset values for all printing units in [°]. If DisableCyclicWrites = TRUE, this value must be added manually by the user.

RMB_TechRegiAdvanced

I/O type	Name	Data type	Explanation
VAR_INPUT	Enable	BOOL	Processing of function block enabled
	Preset	BOOL	The array "CmdPosition" is reset and the values from "PositionPreset" are applied. This value will also still be written to the drive in future if DisableCyclicWrites = FALSE.
	DisableCyclicWrites	BOOL	If the input is set, the cyclic data is not written anymore to the axes, but only output in "CmdPosition".
	Format	REAL	Printing cylinder format [mm]
	Decouple-Mode	MB_DECOUPLE_MODE	It indicates the respective algorithm to cascade the controller control variables.
VAR_OUTPUT	InOperation	BOOL	The decoupling function block operates and the outputs are valid.
	Error	BOOL	Processing completed with errors.
	ErrorID	ERROR_CODE	Error group
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics
	PresetAck	BOOL	Values from "PositionPreset" were written in "CmdPosition" or to the axis.

Tab. 4-23: I/O interface of the MB_RegiGravureRegulateType02 function block

Data structures

The MB_REGI_GRAVURE_REGULATE_CONFIG_TYPE02 data structure extends the MB_REGI_GRAVURE_REGULATE_CONFIG data structure:

MB_REGI_GRAVURE_REGULATE_CONFIG_TYPE02		
Slave	AXIS_REF	Axis of the printing unit.
Active	BOOL	Printing unit is active (printing).
Inc	BOOL	Jogging, incrementing the position (Manual mode or external register controller).
Dec	BOOL	Jogging; decrementing the position (Manual mode or external register controller).
RegiGravureControl	BOOL	The register controller MB_RegiControllerAdvType01 is used if the "RegiGravureControl" is enabled. Thus, in Automatic and Manual mode, the RampedControlValueOneStep input is applied indicating the register regulation in [mm].
RampedControlValueOneStep	REAL	The differentiated control variable of the register controller in [mm].
ManualMode	BOOL	It activates Manual mode. Thus, the control variable is not decoupled.
RedSpeed	BOOL	Selection of the regulating velocity.
NormalSpeed	REAL	Regulating velocity [1/s] in the mode RedSpeed = FALSE.
ReducedSpeed	REAL	Regulating velocity [1/s] in the mode RedSpeed = TRUE.
WebLength	REAL	Web length from the active printing unit located upstream of the own printing unit [mm].
UnitPreset	BOOL	A preset of the individual units is executed with the value in UnitPresetValue.
UnitPresetAck	BOOL	Confirmation that a UnitPreset has been executed successfully.

MB_REGI_GRAVURE_REGULATE_CONFIG_TYPE02		
UnitPresetValue	REAL	Value written upon a rising edge at UnitPreset in CmdPosition of the corresponding unit.
NoDecPreviousUnits	BOOL	If the input is enabled, no decoupling values of upstream printing units are taken into consideration.

Tab. 4-24: MB_REGI_GRAVURE_REGULATE_CONFIG_TYPE02 data structures

Enumeration type

The following enumeration type was described in previous function blocks of the RMB_TechRegiAdvanced library:

- MB_DECOUPLE_MODE (tab. 4-20 "Enumeration type MB_DECOUPLE_MODE" on page 150)

Definition of default settings:

MB_REGI_GRAVURE_REGULATE_MIN_UNIT := 1;

MB_REGI_GRAVURE_REGULATE_MAX_UNIT := 16;

Min/max values and default values of the inputs

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
Preset	BOOL			FALSE	Continuous
DisableCyclic-Writes	BOOL			FALSE	At a rising edge at "Enable"
Format	REAL	> 0	n. def	1.0	At a rising edge at "Enable"
DecoupleMode	MB_DECOUPLE_MODE	0	1	COLOR_TO_COLOR := 0	At a rising edge at "Enable"
MB_REGI_GRAVURE_REGULATE_CONFIG_TYPE02					
Slave	AXIS_REF	NoAxis			At a rising edge at "Enable"
Active	BOOL			FALSE	Continuous
Inc	BOOL			FALSE	Continuous
Dec	BOOL			FALSE	Continuous
RegiGravureControl	BOOL			FALSE	Continuous
RampedControlValueOneStep	REAL	n. def	n. def	0.0	Continuous
ManualMode	BOOL			FALSE	Continuous
RedSpeed	BOOL			FALSE	Continuous
NormalSpeed	REAL	0.0	n. def	1	Continuous
ReducedSpeed	REAL	0.0	n. def	0.5	Continuous
WebLength	REAL	1.0	n. def	10000	At a rising edge at "Enable"
UnitPreset	BOOL			FALSE	Continuous
UnitPresetAck	BOOL			FALSE	Continuous

RMB_TechRegiAdvanced

Name	Type	Min. value	Max. value	Default value	Effective
UnitPresetValue	REAL	n. def	n. def	0.0	Rising edge at UnitPreset
NoDecPreviousUnits	BOOL			FALSE	Continuous

Tab. 4-25: Overview of min/max values and initial values of the MB_RegiGravureRegulateType02 inputs

Function description



The function block MB_RegiGravureRegulateType02 contains the complete functionality of the MB_RegiGravureRegulateType01 function block.

For a function description refer to the following documentation:

- MB_RegiGravureRegulateType01 ([chapter 4.3.4 "MB_RegiGravureRegulateType01" on page 146](#))

Additionally, the function block has two more functionalities:

Execute a preset for an individual printing unit

Previously, only a global preset of all printing units can be executed. Application-specifically, this cannot be sufficient, as often only preset values of a printing unit or a unit are written. If Unit.UnitPreset := TRUE (rising edge), the value of Unit.UnitPresetValue is written to CmdPosition for this unit. Unit.UnitPresetAck indicates if writing of a value was successful. If DisableCyclicWrites = FALSE, this value is additionally written to the corresponding slave axis.

Do not consider any decoupling values of upstream printing units

A unit with upstream units that are active and do not operate in manual mode always contains feed forward values of upstream units. Sporadically, a unit can be decoupled from regulations of upstream units. Via the NoDecPreviousUnits = TRUE input, feed forward values of upstream printing units can be ignored. If NoDecPreviousUnits = FALSE, feed forward values of upstream printing units that are not in manual mode are not taken into consideration. Generally, the feed forward values should always be taken into consideration for printing units. These feed forward values should only be disabled in exceptional cases.

Error handling

The function block uses the error table **F_RELATED_TABLE**, 16#0170.

It can generate the following error messages in Additional 1/Additional 2:

ErrorID	Additional 1	Additional 2	Description
STATE_MACHINE_ERROR (16#0005)	16#00000006	16#00000000	Error in the state machine
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000000	Invalid printing cylinder format
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000003A	Invalid Decouple mode
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000038	Invalid NormalSpeed
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#00000039	Invalid ReducedSpeed
INPUT_RANGE_ERROR (16#0006)	16#00001D11	16#0000002B	Invalid web length

ErrorID	Additional 1	Additional 2	Description
INPUT_RANGE_ERROR (16#0006)	16#0000000D	16#00000000	AxisNo of "Axis" is not in the valid range
RESOURCE_ERROR (16#0003)	16#00000001	16#00000000	No power is added to the drive specified in "Slave". A register control is thus not possible.

Tab. 4-26: Error codes of the MS_RegiGravureRegulateType02 function block
Furthermore, errors can occur in the libraries RIL_LoopControl and MS_TechBase.

5 Service and support

Our worldwide service network provides an optimized and efficient support. Our experts offer you advice and assistance should you have any queries. You can contact us **24/7**.

Service Germany Our technology-oriented Competence Center in Lohr, Germany, is responsible for all your service-related queries for electric drive and controls.

Contact the **Service Hotline** and **Service Helpdesk** under:

Phone: **+49 9352 40 5060**
Fax: **+49 9352 18 4941**
E-mail: service.svc@boschrexroth.de
Internet: <http://www.boschrexroth.com/>

Additional information on service, repair (e.g. delivery addresses) and training can be found on our internet sites.

Service worldwide Outside Germany, please contact your local service office first. For hotline numbers, refer to the sales office addresses on the internet.

Preparing information To be able to help you more quickly and efficiently, please have the following information ready:

- Detailed description of malfunction and circumstances
- Type plate specifications of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your e-mail address)

Index

A

About this documentation.....	3
Information representation.....	9
Validity of the documentation.....	3
ANSI Z535.6-2006.....	10

C

Complaints.....	11
Controlling and and optimizing the printing process	
Introduction and overview.....	129
MB_RegiAccelerationCompensation-	
Type01.....	140
MB_RegiGravureRegulateType01.....	146
MB_RegisterControllerAdvType01.....	130
Create defined velocity characteristics	
Introduction and overview.....	121
Criticism.....	11
Customer Feedback.....	11

D

Data structure	
MB_ADAPT_CON.....	143
MB_MODEL_VALUES.....	143
MB_REGI_BARCODE.....	42, 88
MB_REGI_COMMUNICATION_CONFIG.....	42, 88
MB_REGI_FORMAT.....	41, 87
MB_REGI_INTERNAL_PARAMETER.....	43
MB_REGI_INTERNAL_PARAME-	
TER_TYPE02.....	71, 89
MB_REGI_LENGTH_REGISTER_MODE.....	42, 88
MB_REGI_MARK_STREAM.....	42, 88
MB_REGI_MARK_STRUCTURE.....	41, 87
MB_REGI_RECEIVE_DATA.....	43, 89
MB_REGI_SEND_DATA.....	43, 89
MB_REGI_SENSOR01_CFG.....	41
MB_REGI_SENSOR01_CFG_TYPE02.....	87
MB_REGI_SENSOR01_DATA.....	65, 107
MB_REGI_SENSOR01_DIAG.....	111
MB_REGI_SENSOR01_TEACH_DATA.....	118
MB_REGI_SIDE_REGISTER_MODE.....	42, 88
MB_REGI_TOLERANCES.....	43, 89
MB_REGI_WARNING.....	26, 71

Data structures

MB_REGI_GRAVURE_REGU-	
LATE_CONFIG.....	150
MB_REGI_GRAVURE_REGU-	
LATE_CONFIG_TYPE02.....	155

E

Enumeration type	
MB_COMP_MODE.....	143
MB_DECOUPLE_MODE.....	150
MB_REGI_ADV_CONTROL_MODE.....	135
MB_REGI_ADV_RESPONSE_MODE.....	135
MB_REGI_MARK_STATE.....	66, 108

MB_REGI_MEASURE_HANDLING.....	44, 90
MB_REGI_READ_WRITE_TEACH_DATA..	119
MB_REGI_SENSOR_STATE.....	66, 108
MB_REGI_SENSOR01_COMM_AC-	
TIONS.....	27, 72
MB_REGI_SENSOR01_COMM_STA-	
TUS.....	26, 71

Error codes

MB_ReachVelocityType01.....	125
MB_RegiAccelerationCompensation-	
Type01.....	146
MB_RegiGravureRegulateType01.....	152
MB_RegiSensor01CommType01.....	39
MB_RegiSensor01CommType02.....	85
MB_RegiSensor01ConfigType01.....	56
MB_RegiSensor01ConfigType02.....	102
MB_RegiSensor01CycComType01.....	63
MB_RegiSensor01CycComType02.....	67
MB_RegiSensor01CycComType04.....	109
MB_RegiSensor01ReadDiagDataType01...	112
MB_RegiSensor01SendRecvDataType01....	59
MB_RegiSensor01SendRecvDataType02...	106
MB_RegiSensor01SendSpeedAccType01..	117
MB_RegiSensor01SetIPAddressType01....	121
MB_RegiSensor01TeachDataType01.....	119
MB_RegisterControllerAdvType01.....	140
MS_RegiGravureRegulateType02.....	157

F

Feedback.....	11
---------------	----

H

Hazard warnings.....	9
Helpdesk.....	159
Hotline.....	159

I

Intended use	
Introduction.....	13
Use cases.....	13

M

MB_ReachVelocityType01	
Brief description.....	122
Function description.....	123, 124
Interface description.....	122
Min/max values and default values of the	
inputs.....	123
MB_RegiAccelerationCompensationType01	
Data structures.....	143
Enumeration type.....	143
Error handling.....	145
Function description.....	143
Interface description.....	141

Index

Min/max values and default values of the inputs.....	142	Error handling.....	62
Short description.....	140	Function description.....	61
MB_RegiGravureRegulateType01		Interface description.....	60
Data structures.....	149	Min/max values and default values of the inputs.....	61
Enumeration type.....	150	MB_RegiSensor01CycComType02	
Error handling.....	152	Brief description.....	63
Function description.....	150	Data structures.....	65
Interface description.....	147	Enumeration type.....	65
Min/max values and default values of the inputs.....	149	Error handling.....	67
Short description.....	146	Function description.....	66
MB_RegiGravureRegulateType02		Interface description.....	64
Data structures.....	154	Min/max values and default values of the inputs.....	65
Enumeration type.....	155	MB_RegiSensor01CycComType04	
Error handling.....	156	Brief description.....	106
Function description.....	156	Data structures.....	107
Interface description.....	153	Enumeration type.....	107
Min/max values and default values of the inputs.....	155	Error handling.....	109
Short description.....	152	Function description.....	108
MB_RegiSensor01CommType01		Interface description.....	106
Brief description.....	22	Min/max values and default values of the inputs.....	107
Data structures.....	25	MB_RegiSensor01ReadDiagType01	
Enumeration type.....	26	Brief description.....	110
Error handling.....	38	Data structures.....	111
Function description.....	29	Function description.....	112
Interface description.....	23	Interface description.....	110
Min/max values and default values of the inputs.....	27	Min/max values and default values of the inputs.....	111
MB_RegiSensor01CommType02		MB_RegiSensor01SendRecvDataType01	
Brief description.....	67	Brief description.....	56
Data structures.....	70	Data structures.....	58
Enumeration type.....	71	Error handling.....	59
Error handling.....	84	Function description.....	58
Function description.....	75	Interface description.....	56
Interface description.....	69	Min/max values and default values of the inputs.....	57
Min/max values and default values of the inputs.....	72	MB_RegiSensor01SendRecvDataType02	
MB_RegiSensor01ConfigType01		Brief description.....	102
Brief description.....	39, 41	Data structures.....	104
Enumeration type.....	43	Error handling.....	105
Error handling.....	55	Function description.....	104
Function description.....	46	Interface description.....	103
Interface description.....	40	InterfMin/max values and default values of the input description.....	104
Min/max values and default values of the inputs.....	44	MB_RegiSensor01SendSpeedAccType01	
MB_RegiSensor01ConfigType02		Brief description.....	112
Brief description.....	85, 87	Data structures.....	115
Enumeration type.....	89	Error handling.....	116
Error handling.....	101	Function description.....	115
Function description.....	92	Interface description.....	113
Interface description.....	86	Min/max values and default values of the inputs.....	114
Min/max values and default values of the inputs.....	90	MB_RegiSensor01SetIPAddressType01	
MB_RegiSensor01CycComType01		Brief description.....	119
Brief description.....	59	Error handling.....	121
Data structures.....	61	Function description.....	120

Index

Interface description.....	120
Min/max values and default values of the inputs.....	120
MB_RegiSensor01TeachDataType01	
Brief description.....	117
Data structures.....	118
Enumeration type.....	119
Error handling.....	119
Function description.....	119
Interface description.....	117
Min/max values and default values of the inputs.....	118
MB_RegisterControllerAdvType01	
Enumeration type.....	135
Function description.....	136
Interface description.....	132
Min/Max values and default values of the inputs.....	134
Short description.....	130

R

RMB_TechRegiAdvanced	127
Controlling and and optimizing the printing process.....	129
General information.....	129
Library subject to license.....	127
MB_RegiGravureRegulateType02	152
RMB_TechRegiExtended	15
Communication with the mark stream sensor	18
Create defined velocity characteristics.....	121
General information.....	17
Introduction and overview.....	18
Library subject to license.....	15
MB_ReachVelocityType01	122
MB_RegiSensor01ConfigType01	22, 39
MB_RegiSensor01ConfigType02	67, 85
MB_RegiSensor01CycComType01	59
MB_RegiSensor01CycComType02	63
MB_RegiSensor01CycComType04	106
MB_RegiSensor01ReadDiagType01	110
MB_RegiSensor01SendRecvDataType01	56
MB_RegiSensor01SendRecvDataType02 ...	102
MB_RegiSensor01SendSpeedAccType01 ..	112
MB_RegiSensor01SetIPAddressType01	119
MB_RegiSensor01TeachDataType01	117

S

Safety alert symbol.....	10
Safety instructions.....	9
Service hotline.....	159
Signal words.....	10
Suggestions.....	11
Support.....	159
Symbols.....	11

U

Unintended use.....	14
---------------------	----

Unintended Use	
Consequences, non-liability.....	13

W

Warnings.....	9
---------------	---

Notes

Notes

Bosch Rexroth AG

Electric Drives and Controls

P.O. Box 13 57

97803 Lohr, Germany

Bgm.-Dr.-Nebel-Str. 2

97816 Lohr, Germany

Tel. +49 9352 18 0

Fax +49 9352 18 8400

www.boschrexroth.com/electrics



R911341509