rexroth
A Bosch Company

MTX 15VRS

PLC Interface

Project Planning Manual
R911400172

Edition 04



Title MTX 15VRS
PLC Interface

Type of Documentation Project Planning Manual

Document Typecode DOK-MTX***-PLC*INT*V15-PR04-EN-P

Internal File Reference RS-dd6592dad401e6c90a347e864072ef42-4-en-US-3

Revision history Edition 04, 2024-10
Refer to chapter 1.1 "Change Record" on page 1

Copyright © Bosch Rexroth AG 2024
All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

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 Development of Automation Systems - Solution Integration HMI, NC Controls and Support, ToMa (TaDo/PiaSt)

Table of Contents

Page

1 About this documentation ... 1
1.1 Change Record ... 1
1.2 Purpose ... 1
1.3 Overview on target groups and product phases ... 1
1.4 Required and supplementing documentation MTX ... 2
1.4.1 Selection/compilation ... 2
1.4.2 Configuration ... 2
1.4.3 Commissioning ... 3
1.4.4 Operation ... 4
1.4.5 OEM Engineering ... 5
1.4.6 AddOns ... 5
1.5 Information representation ... 5
1.5.1 Using safety instructions ... 5
1.5.2 Names and abbreviations ... 7
1.5.3 Customer feedback ... 7

2 Important instructions on use ... 9
2.1 Intended use ... 9
2.1.1 Introduction ... 9
2.1.2 Areas of use and application ... 9
2.2 Unintended use ... 10

3 Overview ... 11

4 Configurations ... 15
4.1 Configuring PLC-NC bit interface ... 15
4.1.1 General information ... 15
4.1.2 Global interface (general interface) ... 15
4.1.3 Channel interface ... 15
4.1.4 Axis interface ... 16
4.1.5 Spindle interface ... 17

5 Overview on interface signals ... 19
5.1 Global output signals (PLC --> NC) ... 19
5.2 Global input signals (NC --> PLC) ... 19
5.3 Channel-related output signals (PLC --> NC) ... 20
5.4 Channel-related input signals (NC -> PLC) ... 22
5.5 Axis-related output signals (PLC -> NC) ... 24
5.6 Axis-related input signals (NC --> PLC) ... 26
5.7 Spindle-related output signals (PLC -> NC) ... 27
5.8 Spindle-related input signals (NC -> PLC) ... 29

Page

6 Global interface signals ... 31
6.1 Overview on global interface signals ... 31
6.1.1 General information ... 31
6.1.2 Overview on output signals (PLC -->NC) ... 31
6.1.3 Overview on input signals (NC --> PLC) ... 31
6.2 Signal description ... 32
6.2.1 Output signals (PLC --> NC) ... 32
6.2.2 Input signals (NC --> SPS) ... 35

7 Channel-related interface signals ... 37
7.1 Overview on channel-related interface signals ... 37
7.1.1 General information ... 37
7.1.2 Overview on channel-related output signals (PLC --> NC) ... 37
7.1.3 Overview on input signals (NC -> PLC) ... 39
7.2 Signal description ... 41
7.2.1 Channel-related output signals (PLC --> NC) ... 41
7.2.2 Input signals (NC -> PLC) ... 53

8 Axis-related interface signals ... 67
8.1 Overview on axis-related interface signals ... 67
8.1.1 General information ... 67
8.1.2 Overview on axis-related output signals(PLC --> NC) ... 67
8.1.3 Overview on axis-related input signals (NC --> PLC) ... 69
8.2 Signal description ... 70
8.2.1 Axis-related output signals(PLC -> NC) ... 70
8.2.2 Input signals (NC --> PLC) ... 82

9 Spindle-related interface signals ... 93
9.1 Overview on spindle-related interface signals ... 93
9.1.1 General information ... 93
9.1.2 Overview on output signals (PLC -> NC) ... 93
9.1.3 Overview on input signals (NC --> PLC) ... 95
9.2 Signal description ... 96
9.2.1 Output signals (PLC--> NC) ... 96
9.2.2 Input signal (NC--> PLC) ... 106

10 Auxiliary functions ... 117
10.1 General information ... 117
10.2 Bit-coded auxiliary functions ... 118
10.3 Non-bit-coded channel-independent auxiliary functions ... 119
10.4 Non-bit-coded channel-dependent auxiliary functions ... 120

11 Connecting digital I/Os ... 123

Page

12 Program function blocks ... 125
12.1 General structure of program function blocks ... 125
12.2 Reading actual axis values (MT_ScsPos) ... 125
12.2.1 Function ... 125
12.2.2 Configuration example ... 126
12.2.3 Status messages ... 127
12.3 Program selection/deselection (MT_ProgSel) ... 127
12.3.1 Function ... 127
12.3.2 Configuration example ... 129
12.3.3 Status messages ... 130
12.4 NC block specification (MT_NcBlk) ... 131
12.4.1 Function ... 131
12.4.2 Configuration example ... 131
12.4.3 Status messages ... 132
12.5 Extended NC block specification (MT_NcBlkExt) ... 132
12.5.1 Function ... 132
12.5.2 Configuration example ... 134
12.5.3 Status messages ... 134
12.6 Channel restart (MT_ChanRestart) ... 135
12.6.1 Function ... 135
12.6.2 Configuration example ... 135
12.6.3 Status messages ... 136
12.7 Temperature Compensation (MT_TempComp) ... 136
12.7.1 Function ... 136
12.7.2 Configuration example ... 137
12.7.3 Status messages ... 138
12.8 Reading machine parameters (MT_CfgData) ... 138
12.8.1 Function ... 138
12.8.2 Configuration example ... 139
12.8.3 Status messages ... 140
12.9 Handwheel data (MT_Handw) ... 140
12.9.1 Function ... 140
12.9.2 Configuration example ... 141
12.9.3 Status messages ... 141
12.10 Sercos parameters (MT_ScsData) ... 142
12.10.1 Function ... 142
12.10.2 Configuration example ... 143
12.10.3 Status messages ... 146
12.11 Editing Tool Lists (MT_P_DbRecList) ... 147
12.11.1 Function ... 147
12.11.2 Configuration example ... 150
12.11.3 Status messages ... 151
12.12 Editing data blocks (MT_P_DbData) ... 152
12.12.1 Function ... 152
12.12.2 Configuration example ... 153
12.12.3 Status messages ... 155

Page

12.13 Editing data block lists (MT_DbRecList)... 156
12.13.1 Function... 156
12.13.2 Configuration example... 159
12.13.3 Status messages... 161
12.14 Editing data blocks (MT_DbData)... 162
12.14.1 Function... 162
12.14.2 Configuration example... 166
12.14.3 Status messages... 167
12.15 Backing up the database table (MT_DbSave)... 168
12.15.1 Function... 168
12.15.2 Configuration example... 170
12.15.3 Status messages... 171
12.16 Loading the database table (MT_DbLoad)... 172
12.16.1 Function... 172
12.16.2 Configuration example... 173
12.16.3 Status messages... 174
12.17 Tool Correction (MT_TCorr)... 175
12.17.1 Function... 175
12.17.2 Configuration example... 177
12.17.3 Status messages... 179
12.18 Reading/writing permanent CPL variables (MT_CplData)... 179
12.18.1 Function... 179
12.18.2 Configuration example... 181
12.18.3 Status messages... 182
12.19 Reading/writing permanent CPL variables with type conversion (MT_CplDataConv)... 182
12.19.1 Function... 182
12.19.2 Configuration example... 184
12.19.3 Status messages... 185
12.20 Reading System Data (MT_SD_RD)... 185
12.20.1 Function... 185
12.20.2 Configuration example... 187
12.20.3 Status messages... 187
12.21 Writing System Data (MT_SD_WR)... 188
12.21.1 Function... 188
12.21.2 Configuration example... 189
12.21.3 Status messages... 190
12.22 Reading diagnostic data (MT_DiagCode)... 191
12.22.1 General information... 191
12.22.2 Function block parameters... 192
12.22.3 Configuration example... 195
12.22.4 Status messages... 195
12.23 Reading diagnostic texts (MT_DiagText)... 195
12.23.1 General information... 195
12.23.2 Function block parameters... 196
12.23.3 Configuration example... 199
12.23.4 Status messages... 199

Page

12.24 Reading Motion Control Data Services (MT_MCoDS) ... 199
12.24.1 General information ... 199
12.24.2 Function block parameters ... 200
12.24.3 Configuration example ... 200
12.24.4 Status messages ... 200
12.25 Automatic Sercos III I/O exchange (MT_S3AutoAdapt) ... 201
12.25.1 General information ... 201
12.25.2 Function block parameters ... 201
12.25.3 Configuration example ... 201
12.25.4 Status messages ... 201
12.26 Reading out the Sercos ring topology (MT_RingTop) ... 202
12.26.1 General information ... 202
12.26.2 Function block parameters ... 202
12.26.3 Configuration example ... 203
12.26.4 Status messages ... 203
12.27 Log entry in the action recorder (MT_LogData) ... 203
12.27.1 General information ... 203
12.27.2 Function block parameters ... 204
12.27.3 Configuration examples ... 205
12.27.4 Status messages ... 206
12.28 Reading System Data in Segments (MT_SD_Rd_Seg) ... 206
12.28.1 General information ... 206
12.28.2 Function block parameters ... 207
12.28.3 Configuration example ... 207
12.28.4 Status messages ... 208
12.29 Writing System Data in Segments (MT_SD_Wr_Seg) ... 209
12.29.1 General information ... 209
12.29.2 Function block parameters ... 210
12.29.3 Configuration example ... 210
12.29.4 Status messages ... 210
12.30 Reading fast action input data (MT_FaDataRead) ... 211
12.30.1 General information ... 211
12.30.2 Function block parameters ... 212
12.30.3 Configuration example ... 212
12.30.4 Status messages ... 214
12.31 Writing fast action output data (MT_FaDataWrite) ... 215
12.31.1 General information ... 215
12.31.2 Function block parameters ... 215
12.31.3 Status messages ... 215
12.31.4 Configuration example ... 216

13 Configuring PLC-specific data in IndraWorks ... 219
13.1 Configuring machine state display (MZA) ... 219
13.1.1 General information ... 219
13.1.2 Parameterizing the machine state display (MZA) ... 219
13.1.3 Configuring MZA bit interface ... 219

Page

13.1.4 MZA file structure ... 220
13.1.5 Enabling messages ... 221
13.1.6 Supplementing notes ... 221

14 MT_BaseAdditional.library ... 223
14.1 Introduction and overview ... 223
14.2 Standard interfaces at function blocks ... 225
14.2.1 Motivation ... 225
14.2.2 Function block types ... 225
14.2.3 Block types and their interfaces ... 226
14.2.4 Inputs and outputs of state-controlled function blocks, final processing ... 228
14.2.5 Inputs and outputs of edge-controlled function blocks, final processing ... 231
14.2.6 Inputs and outputs of state-controlled function clocks, permanent processing ... 234
14.2.7 Inputs and outputs of edge-triggered function blocks, permanent processing ... 237
14.3 NC channel ... 239
14.3.1 Overview ... 239
14.3.2 MT_ChNumberValidation, Checking validity of channel number ... 240
14.3.3 MT_ChGetState, Determine current channel status ... 242
14.3.4 MT_ChGetActiveProgram, Determining active NC program ... 244
14.3.5 MT_ChNcProgProgress, Determining current program progress ... 246
14.3.6 MT_ChSetReentryBreakPoint, Saving breakpoint for re-entry ... 251
14.3.7 MT_ChSetFeedOverride, Change channel feed ... 254
14.3.8 MT_ChGetFeedOverride, Determining active feed override ... 255
14.3.9 MT_ChGetOPMode, Determine active operation mode ... 256
14.3.10 MT_ChSetMainOPMode, Changing the NC main operation modes ... 259
14.3.11 MT_ChSetOPMode, Changing the NC operation modes ... 263
14.3.12 MT_ChOPMode Changing the NC main operation modes ... 268
14.4 Axes ... 273
14.4.1 Overview ... 273
14.4.2 MT_DrvGetSafeState, Determining the active Safety Engineering operating state ... 274
14.4.3 MT_AxGetChIndex, Determining the channel assignment of the axis ... 278
14.4.4 MT_AxHandwheelAcs, Changing the handwheel ... 278
14.4.5 MT_AxProbeMonitor, Monitoring the probe axis ... 280
14.4.6 MT_AxGetJogMode, Determining Active JogMode ... 283
14.4.7 MT_AxSetJogMode, Changing the JogMode ... 285
14.4.8 MT_AxJogMode, Changing and determining the active jog distance ... 289
14.4.9 MT_AxAsyncGetOPMode, Determining the operation mode of an asynchronous axis ... 294
14.4.10 MT_AxAsyncOPMode, Changing the operation mode of an asynchronous axis ... 295
14.4.11 MT_AxAsyncOPMode, operation mode of an asynchronous axis ... 298
14.4.12 MT_AxAsyncSetOverride, Changing the feed for an asynchronous axis ... 301
14.4.13 MT_AxGetPowerState, Determining the drive status ... 302
14.4.14 MT_AxGetPowerStateExt, drive status incl. Determining errors and warnings ... 303
14.4.15 MT_AxSetPowerOn, Setting the controller enable for an axis ... 305
14.4.16 MT_AxSetPowerOff, Setting the controller enable for an axis ... 306
14.4.17 MT_AxPower, Controller enable and axis status ... 308
14.5 Spindles ... 311

Page

14.5.1 Overview ... 311
14.5.2 MT_SpSetOverride, Changing the feed for a spindle ... 312
14.5.3 MT_SpGetJogSpeed, Determining the active jog velocity ... 313
14.5.4 MT_SpSetJogSpeed, Changing the jog velocity of a spindle ... 314
14.5.5 MT_SpJogSpeed, Changing and determining the active jog velocity ... 317
14.5.6 MT_SpGetPowerState, Determining the drive status ... 320
14.5.7 MT_SpGetPowerStateExt, drive status incl. Determining errors and warnings ... 322
14.5.8 MT_SpSetPowerOff, Setting the controller enable for a spindle ... 323
14.5.9 MT_SpSetPowerOff, Sets controller enable for a spindle ... 325
14.5.10 MT_SpPower, Controller enable and spindle status ... 327
14.6 Tool management ... 330
14.6.1 Overview ... 330
14.6.2 MT_TmToolLife, tool, tool life counting ... 330
14.6.3 MT_TmChGetActiveEdge, Determining active edge of the active tool ... 335
14.7 Miscellaneous ... 336
14.7.1 MT_DistanceAccumulator, Calculates the path distance from the specified position data ... 336
14.7.2 MT_TimeAccumulator, Calculates the lapsed time ... 339
14.7.3 MT_GetRandomNumber, Generates an integer random number ... 341
14.8 Data types ... 343
14.8.1 MT_CHANNEL_OPERATION_MODE, Structure with all channel operation modes ... 343
14.8.2 MT_CHANNEL_STATE, Structure contains all channel states ... 343
14.8.3 MT_SD_SYS_TOOL, Structure of the SD.SysTool system date ... 344
14.8.4 MT_AXIS_ASYNC_OPERATION_MODE, Structure with all axis operation modes ... 344
14.8.5 MT_AXIS_JOG_MODE, Structure with jog distances of an axis ... 345
14.8.6 MT_DRIVE_ERROR_CLASS_1, Structure error details of S-0-0011 of an axis ... 345
14.8.7 MT_DRIVE_ERROR_CLASS_2, Structure warning details of S-0-0012 of an axis ... 346
14.8.8 MT_DRIVE_POWER_STATE, Drive status ... 346
14.8.9 MT_DRIVE_SAFE_OP_MODE, Structure contains the safety operating state of a drive ... 347
14.8.10 MT_SPINDLE_JOG_SPEED, Structure with spindle jog velocities ... 347

15 MT_PLCopen.library ... 349
15.1 Introduction and overview ... 349
15.2 Global_Variables ... 350
15.2.1 Data types ... 350
15.3 POUs ... 351
15.4 MB_Command ... 351
15.5 MB_ReadListParameter ... 356
15.6 MB_ReadMaxRealValue ... 358
15.7 MB_ReadMaxValue ... 360
15.8 MB_ReadMinRealValue ... 362
15.9 MB_ReadMinValue ... 364
15.10 MB_ReadName ... 366
15.11 MB_ReadParameter ... 367
15.12 MB_ReadRealParameter ... 369
15.13 MB_ReadSercosAttribute ... 371
15.14 MB_ReadSercosDataStatus ... 372

Page

15.15 MB_ReadStringParameter ... 374
15.16 MB_ReadUnit ... 376
15.17 MB_WriteParameter ... 378
15.18 MB_WriteRealParameter ... 379
15.19 MB_WriteStringParameter ... 381

16 Service and support ... 385

Index ... 387

1 About this documentation

1.1 Change Record

Editions of this documentation

Edition	Release date	Note
01	2020-07	First edition
02	2020-11	Changes in the chapter "Configurations" Changes in the chapter "Configuring PLC-specific data in IndraWorks" Changes in the chapter "MT_BaseAdditional.library"
03	2023-01	Changes in chapter chapter 12.7.2 "Configuration example" on page 137
04	2024-10	Revision: • chapter 8.2 "Signal description" on page 70 (Defdb00212637, PBI 863405) • chapter 9.2 "Signal description" on page 96 (PBI 863405)

Tab. 1-1: Revision history

1.2 Purpose

This documentation describes the interface signals and the program function blocks for the integrated PLC.

1.3 Overview on target groups and product phases

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

Example: The target group "Programmer" can "program" in the product phase "Engineering" using this documentation.

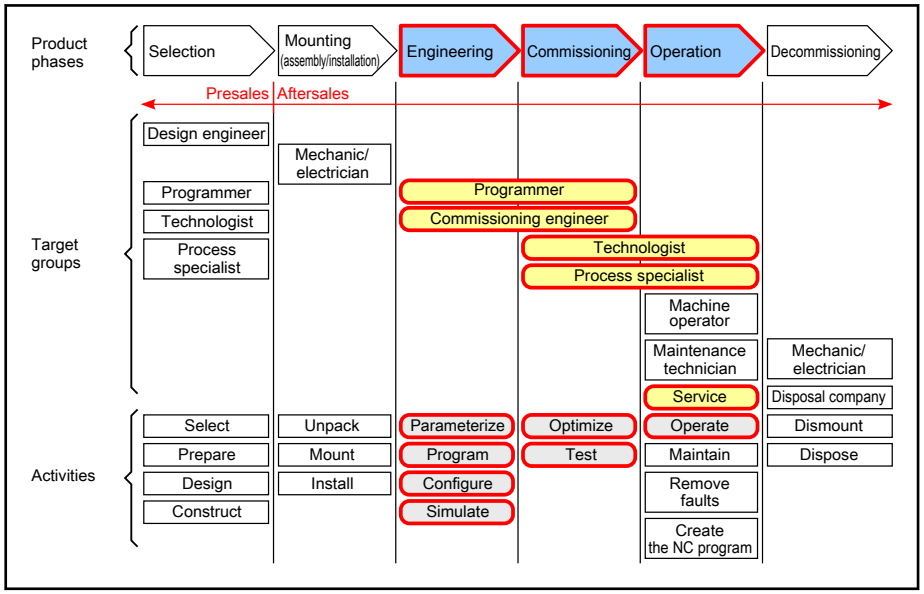


Fig. 1-1: Assigning this documentation to the target groups, product phases and target group activities

1.4 Required and supplementing documentation MTX

1.4.1 Selection/compilation

Documentation titles with type codes and part numbers

MTX 15VRS System Description
DOK-MTX***-SYS*DES*V15-PRRS-EN-P, R911397742
This documentation describes the MTX control. It includes the designs, technical data, interfaces as well as the configuration of the control components.
MTX 15VRS SafeLogic System Overview
DOK-MTX***-SL**SYS*V15-PRRS-EN-P, R911398637
This documentation describes the use of the safety control SafeLogic in the MTX.
IndraControl XM42 Controls
DOK-CONTRL-XM4X*CTRL**-ITRS-EN-P, R911345566
This documentation describes the IndraControl XM42 controls.

RS Corresponding edition

Tab. 1-2: MTX documentation overview - Selecting/composing

1.4.2 Configuration

Documentation titles with type codes and part numbers

MTX 15VRS Machine Parameters
DOK-MTX***-MA*PAR**V15-RERS-EN-P, R911400176
This documentation describes structure and the adjustment of available parameters of the MTX. It also includes the functions of the NC configurator and its operation.
MTX 15VRS PLC Interface
DOK-MTX***-PLC*INT*V15-PRRS-EN-P, R911400172
This documentation describes the interface signals and the program function blocks for the integrated PLC.
MTX 15VRS Functional Description - Basics
DOK-MTX***-NC*F*BA*V15-RERS-EN-P, R911401401
This documentation describes the basic functions of the MTX. The basic commissioning steps and the control functions are provided as description and handling instruction.
MTX 15VRS Functional Description - Extension
DOK-MTX***-NC*F*EX*V15-RERS-EN-P, R911393316
This documentation describes the extended functions of the MTX. The basic commissioning steps and the control functions are provided as description and handling instruction.
MTX 15VRS Functional Description - Special Functions
DOK-MTX***-NC*F*SP*V15-RERS-EN-P, R911393309
This documentation describes the special functions of the MTX. The basic commissioning steps and the control functions are provided as description and handling instruction.

MTX Free Form Surface Milling
DOK-MTX***-FREEFORM***-APRS-EN-P, R911341435
This documentation describes the free form surface milling process wit the MTX control. CNC programs generated by a CAD/CAM system are used as basis for the entire process. Overview over the following topics: Description of the MTX or CNC function for freeform surface milling, strategy during the NC parameterization process, boundary conditions when generating CNC programs with CAM software.
MTX Conversion of MTX Projects
DOK-MTX***-PROCONV****-PRRS-EN-P, R911342484
This documentation provides support during the conversion of MTX 1.x projects to MTX 2G. This project planning manual provides information on how to convert a project and identifies potential difficulties during the conversion.

RS Corresponding edition

Tab. 1-3: MTX documentation overview - Configuring

1.4.3 Commissioning

Documentation titles with type codes and part numbers

MTX 15VRS Commissioning
DOK-MTX***-STARTUP*V15-CORS-EN-P, R911393281
This documentation describes the commissioning of the MTX control. Apart from a complete overview, commissioning and configuration of the axes and the user interface as well as the PLC data are described.
IndraWorks 15VRS Basic Libraries, IndraLogic 2G
DOK-IL*2G*-BASLIB**V15-LIRS-EN-P, R911398633
This documentation describes the system-comprehensive PLC libraries.
IndraWorks 15VRS Field Buses
DOK-IWORKS-FB******V15-APRS-EN-P, R911393284
This documentation describes the field bus and local periphery connections supported by the MLC and MTX systems. The focus of this documentation lies in the configuration, parameterization, commissioning and diagnostics of different periphery connections. It is the basis for the online help.
IndraWorks 14VRS WinStudio 7.4
DOK-IWORKS-WINSTUD*V14-APRS-EN-P, R911341585
This “User Manual and Technical Reference Book” supports the user in achieving the best results with the “Rexroth WinStudio”™ software. The document provides technical information and step-by-step instructions to create web-enabled HMI/SCADA programs.
IndraWorks 15VRS Software Installation
DOK-IWORKS-SOFTINS*V15-CORS-EN-P, R911393450
This documentation describes the IndraWorks installation.
IndraWorks 15VRS Engineering
DOK-IWORKS-ENGINEE*V15-APRS-EN-P, R911393303
This documentation describes the use 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.
IndraWorks 15VRS PLC Programming System IndraLogic 2G
DOK-IWORKS-IL2GPRO*V15-APRS-EN-P, R911396137
This documentation describes the PLC programming tool IndraLogic 2G and its use. The documentation includes the basic use, first steps, visualization, menu items and editors.

IndraWorks 15VRS HMI
DOK-MLC***-HMI*****V15-APRS-EN-P, R911399270
This documentation describes the functions, configuration and operation of the user interfaces IndraWorks HMI Engineering and IndraWorks HMI Operation.
MTX 15VRS Shape Cutting Technology
DOK-MTX***-TECHCUT*V15-CORS-EN-P, R911399147
This documentation describes the shape cutting technology used by the machines to cut materials using an NC-controlled tool head.
MTX-integrated CAD/CAM System for Shape Cutting
DOK-MTX***-CAD*TECHCUT-CO01-EN-P, R911393320
This documentation describes the mode of operation and the area of application of the CAD/CAM nesting software Lantek Expert Inside.

RS Corresponding edition

Tab. 1-4: MTX documentation overview - Commissioning

1.4.4 Operation

Documentation titles with type codes and part numbers

MTX 12VRS Block Pre-Run
DOK-MTX***-BLK*RUN*V12-APRS-EN-P, R911334379
This documentation explains to the machine manufacturer how to setup the "Block pre-run" function at the machine for the end user.
MTX 15VRS Programming Manual
DOK-MTX***-NC**PRO*V15-RERS-EN-P, R911393318
This documentation describes the standard programming of the MTX control. Apart from the basics of NC programming, the use of NC functions according to DIN 66025 as well as the NC functions with high-level language syntax and CPL functions are described.
MTX 15VRS Standard NC Operation
DOK-MTX***-NC*OP***V15-APRS-EN-P, R911393314
This documentation describes the operation of the standard user interface of the NC control of the MTX. It includes the operation of the interface, the NC program development as well as the tool management.
MTX 15VRS Multitouch
DOK-MTX***-MULTI***V15-APRS-EN-P, R911393311
The Multitouch user interface of the NC control MTX is described in this documentation.
MTX 15VRS Standard NC Cycles
DOK-MTX***-NC*CYC**V15-PRRS-EN-P, R911394940
This documentation describes the application of the standard cycles of the different technologies for the MTX control.
MTX 15VRS NC Simulation
DOK-MTX***-NC*SIM**V15-APRS-EN-P, R911393273
This documentation describes the NC simulation for the MTX control.
MTX 15VRS Measuring Functions
DOK-MTX***-MES*FUN*V15-APRS-EN-P, R911394938
This documentation describes the measuring cycles of the MTX control.

RS Corresponding edition

Tab. 1-5: MTX documentation overview - Operating

1.4.5 OEM Engineering

Documentation titles with type codes and part numbers

MTX 13VRS Automation Interface
DOK-MTX***-AUT*INT*V13-APRS-EN-P, R911337274
This documentation describes the script-based access to IndraWorks project data via the interface of the Automation Interface. Different objects including code examples are described. The Automation Builder is also described in this manual.
MTX 15VRS OPC Communication
DOK-MTX***-OPC*COM*V15-PRRS-EN-P, R911399272
This documentation describes the syntax and the structure of the items for the communication with Bosch Rexroth devices.
IndraWorks OPC UA Communication
DOK-IWORKS-OPC*UA*****-APRS-EN-P, R911379309
This documentation describes the OPC UA communication of the MLC and MTX control systems.

RS Corresponding edition

Tab. 1-6: MTX documentation overview - OEM engineering

1.4.6 AddOns

Documentation titles with type codes and part numbers

MTX 11VRS Action Recorder
DOK-MTX***-ACR*****V11-APxx-EN-P, R911329943
This documentation describes the MTX action recorder. It includes the installation and commissioning as well as interface signals, application and operation.
MTX 15VRS Efficiency Workbench MTX cta, MTX ega
DOK-MTX***-EWB*****V15-APRS-EN-P, R911400178
This documentation describes the mode of operation and the area of application of the analysis tool MTX cta and MTX ega.
MTX Remote Condition Monitoring
DOK-MTX***-RCM*****V01-APRS-EN-P, R911334383
This documentation describes the operation of the Remote Condition Monitoring System.
MTX visIREC User Documentation
DOK-MTX***-VISIREC*V01-APRS-EN-P, R911344242
This documentation describes the analysis tool visIREC used to optimize the free form surface milling process. 2D or 4D display of path-related data. 2D or 4D display of coordinate-related data. Analysis of critical areas (path and orientation deviation). Comparing the programmed NC blocks to the interpolated NC blocks.

xx / RS Corresponding edition

Tab. 1-7: MTX documentation overview - AddOns

1.5 Information representation

1.5.1 Using safety instructions

Structure of the safety instructions

The safety instructions are structured as follows:

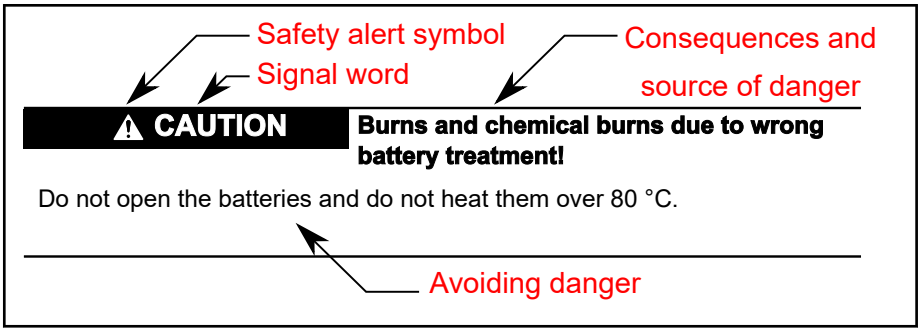


Fig. 1-2: Structure of the safety instructions

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 draws attention to the safety instruction and indicates the risk potential.

The safety alert symbol (triangular safety reflector with exclamation marks), preceding the signal words Danger, Warning, Caution indicates hazards for persons.

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

In case 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.

Explaining the signal alert symbol on the device



If this symbol is on your device, you have to observe the documentation on the device. The respective documentation informs on the type of hazard as well as the steps required to avoid this hazard.

1.5.2 Names and abbreviations

Term	Explanation
IWE	IndraWorks Engineering
IWO	IndraWorks Operation
NC	Numerical Control
OEM	Original Equipment Manufacturer
Profibus	Communication connection
Sercos	Sercos (Serial Realtime Communication System) is a world-wide standardized digital interface used for communication between controls and drives

Tab. 1-8: Terms and abbreviations

1.5.3 Customer feedback

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

2 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 result in damage to property or 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 user 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:

- Anyone handling Bosch Rexroth products in any way is obliged to read and consent to the relevant safety instructions and the intended use.
- Hardware products may not be altered and have to remain in their original state; i.e. no structural changes are permitted. It its 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.

☞ Ensure that the data present in the control or entered or read in by the user is correct before applying it to exclude unwanted axis motion. It can be the following invalid or old data:

- Part programs
- ZO tables
- Compensation tables
- Tool tables
- Permanent CPL variables
- Remanent PLC data
- Permanent system data

2.1.2 Areas of use and application

The MTX control is used to

- program contour and machining technology (path feed, spindle speed, tool change) of a workpiece.
- guide a machining tool along a programmed path.

Feed drives, spindles and auxiliary axes of a machine tool are activated via Sercos interface.



This additionally requires I/O components for the integrated PLC which - together with the actual CNC - control the machining process as a whole and also monitors this process with regard to technical safety.

It may only be operated with the explicitly specified hardware component configurations and combinations and only with the software and firmware specified in the appropriate documentations and functional descriptions.

The CNC system MTX provides the perfect CNC system solution for cutting and forming for the following technologies:

- Rotate
- Milling
- Drilling
- Grinding
- Bending
- Nibbling
- Punching
- Contour cutting
- Handling

2.2 Unintended use

Using the CNC system MTX outside the previously specified areas of application or under operating conditions other than the conditions described in the documentation and the specified technical data, is defined as "unintended use".

The CNC system MTX must not be used if...

- it is subjected to operating conditions not corresponding to the specified ambient conditions. Operation under water, under extreme temperature fluctuations or under extreme maximum temperatures is prohibited.
- Furthermore, the CNC system MTX shall not be used for applications not expressly approved of by Bosch Rexroth. Therefore, please read the information given the general safety instructions!
- The CNC system MTX may not be used in systems or machines connected to the internet via an unsecure network connection. Otherwise, malfunctions or a control failure can result due to unauthorized access.

3 Overview

This manual describes software interfaces and their mode of operation seen from the PLC control integrated into the CNC system MTX.



A detailed description of the integrated PLC is provided in other manuals.

The PLC sequential program is principally capable of communicating with all functional areas and subsystems of the complete system. The manual describes the functions that can be addressed directly by the PLC sequential program using the PLC operands.

PLC interface specifications

Type	Size	Address area
%IB	8 kB	%IB0 … %IB8192
%QB	8 kb	%QB0 … %QB8192
%MB	32 kB	%MB0 … %MB32768

Individual signals

Individual signals that signal a certain state to the PLC sequential program (e.g. axis in position, auxiliary function output) or that activate an NC function (e.g. NC start, stop, feed enable) are refreshed in every PLC cycle or transferred in the corresponding direction when an event occurs.

They are divided into:

- Global Interface Signals, page 31
- Channel-Related Interface Signals, page 37
- Axis-Related Interface Signals, page 67
- Spindle-Related Interface Signals, page 93
- Decoded auxiliary functions, page 117

These interfaces between the PLC and the NC are configured within the IndraLogic using the configurator of the IndraLogic programming interface in the input or output area.

Each interface can be configured individually:

- The node number specifies the physical interface number (channel number from 0, axis and spindle numbers from 1).
- Each interface can be provided with a symbolic identifier to address the interface in the sequential program.
- A separate I/O address can be assigned to each interface. Thus, the physical address of the interface is not affected when inserting or deleting interfaces.

If more interfaces are configured in the system than in the I/O configurator, the unconfigured interfaces are not taken into account during data exchange.

Program function blocks

In addition to the transfer of individual signals, other functions are implemented that are required either rarely or only in special applications. These functions can be called and activated using parameterizable program function blocks.

Each call of a program function block is acknowledged and - according to the called function - the requested date is stored in the operands parameterized by the PLC sequential program.



Available program function blocks, see chapter 12 "Program function blocks" on page 125.

Example:

Communication structure examples

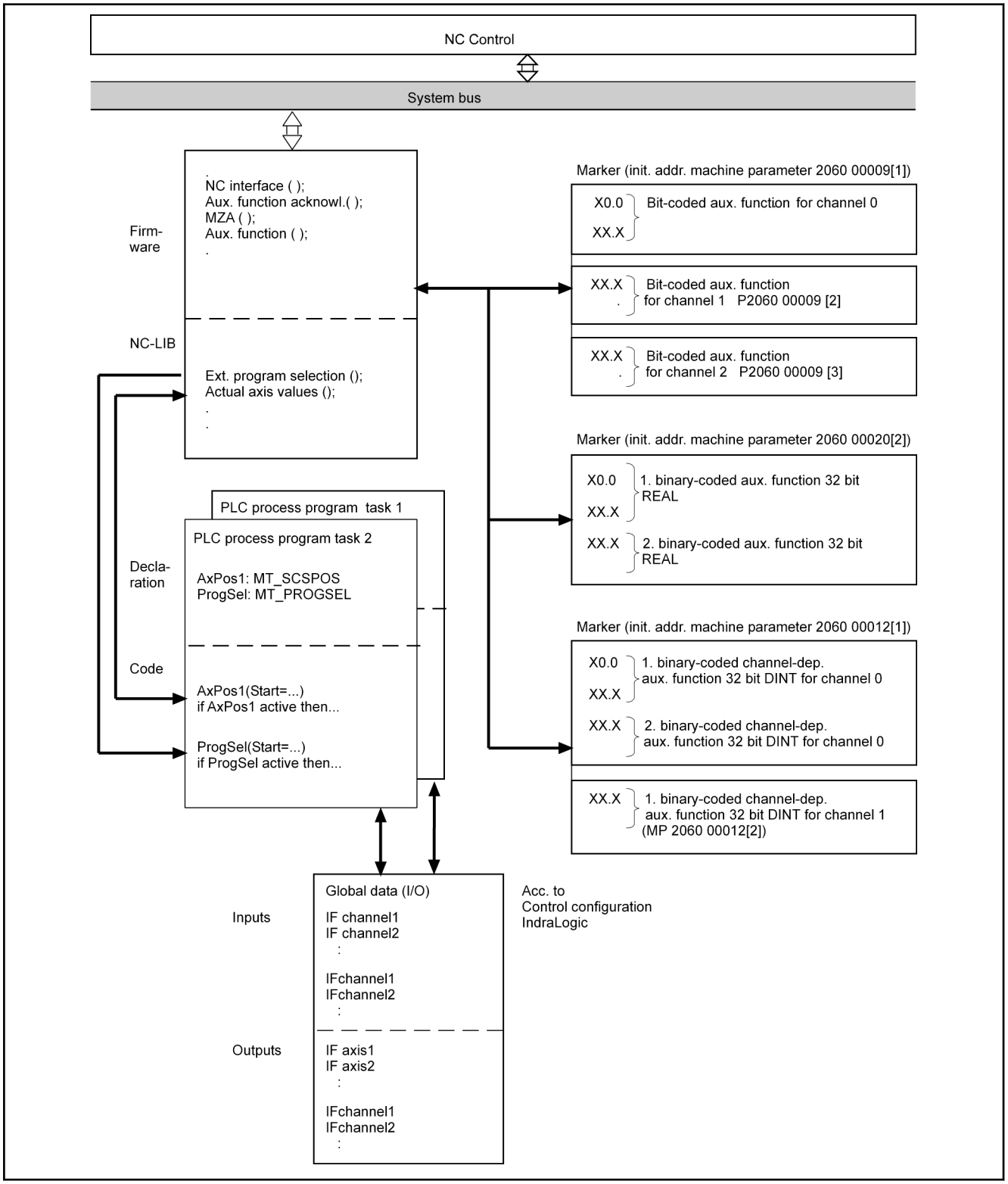


Fig. 3-1: Communication structure examples

Storing PLC-specific files in the control

All files created by the IndraLogic runtime system are stored in the RAM file system under the path **/plc/**. This is the boot project with information on the saved icons (file with the name <Name of application>.app) and some internally used files. If the RAM file system is newly created (for example during firmware exchange), this data is lost and has therefore to be previously saved on **/usrfep/plc** or in an NC archive (tar-file). If the files were backed up in a tar-file, they are saved while reading the tar-files in **/plc** and automatically at next control startup.

If the files were backed up in **/usrfep/plc**, they can be restored by "manual" copying to **/plc** or by a control startup using startup mode 2 after creating the file system.

4 Configurations

4.1 Configuring PLC-NC bit interface

4.1.1 General information

After a new IndraWorks project, a visualization (e.g. PR/VR4x) and the XM4x control has been created, the configuration of the PLC-NC bit interface is a subnode of this control. The corresponding node name is "MTX_NC_PLC_Interface".

The bit interface consists of the global interface (general interface), the channel interface, the axis interface and the spindle interface. The machine state display (MZA) can also be added to the bit interface .

The configuration of the MZA is described in chapter 13.1 Configuring Machine State Display (MZA), page 219.

By default, only the global data interface exists and is pre-assigned.

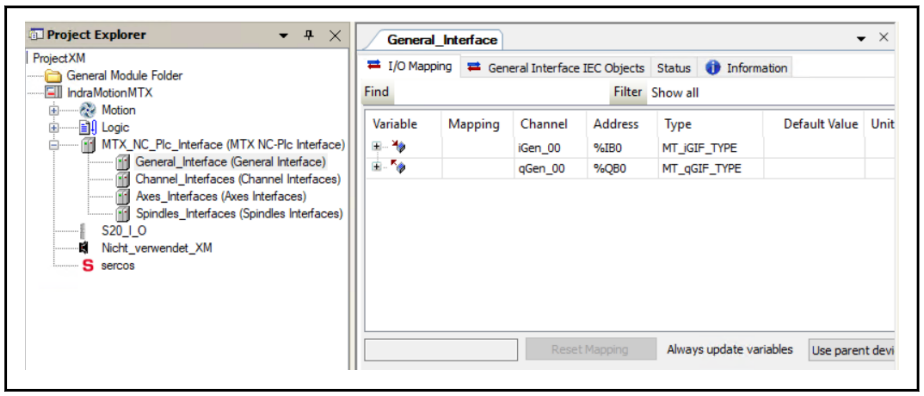


Fig. 4-1: The pre-assigned bit interface with the "Properties" dialog for the global interface

The bit interface signals are always to be seen from the PLC. For example, the signals from "qGen" of the global interface are from PLC to NC.

4.1.2 Global interface (general interface)

The global interface is always present and cannot be deleted. The symbolic name for output signals is "qGen" and the symbolic name for input signals is "iGen". Double-click on the "General Interface" node (or click on "Open" in the context menu) to open the "Properties" dialog. In this "Properties" dialog, pre-assigned names and addresses for the inputs and outputs can be changed and comments can be specified.

4.1.3 Channel interface

The MTX supports a switching function channel (channel 0) and up to twelve machining channels. By default, no channels are defined. New channels are created via the context menu on the "Channel Interfaces" node.

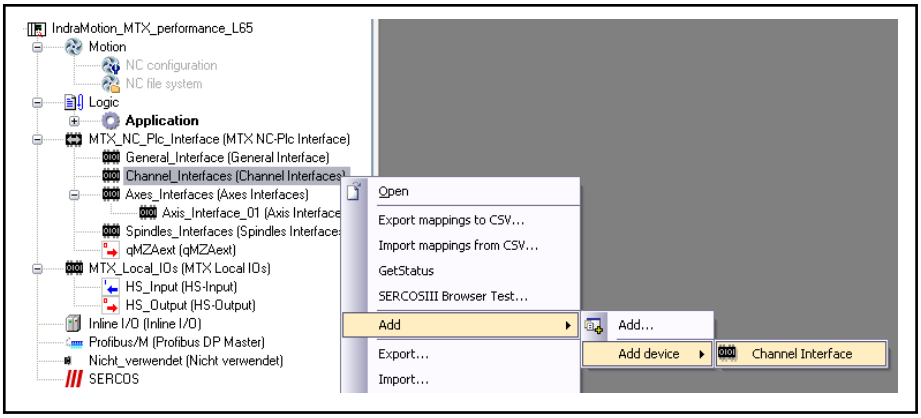


Fig. 4-2: Example of creating channel 1

New channels are continuously created starting with channel 0 (switching function channel). The index is the deciding factor for the significance of a channel. Thus. for example, a channel interface with the index is the interface for the machining channel 2.

The symbolic name for output signals is "qChan_<Node number with leading zero>" and the symbolic name for input signals is "iChan_<Node number with leading zero>".

Double-click on the "Channel Interface <Node number>" node (or click on "Open" in the context menu) to open the "Properties" dialog.

Use the "Properties" dialog to change symbolic names, addresses and comments as well as the corresponding index.

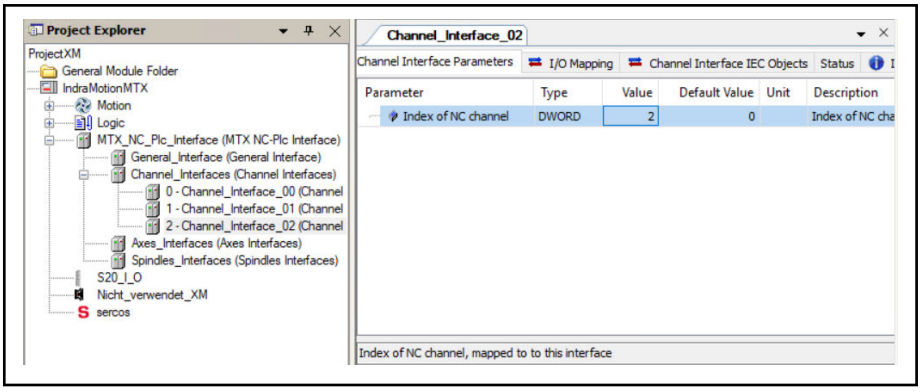


Fig. 4-3: "Properties" dialog for the channel interface



The index for channel interfaces always has to be unique. Therefore, each index may occur only once within the channel interfaces.

4.1.4 Axis interface

The MTX supports up to 64 axes. By default, no axes are yet defined. New axes are created on the "Axis Interfaces" node via the context menu.

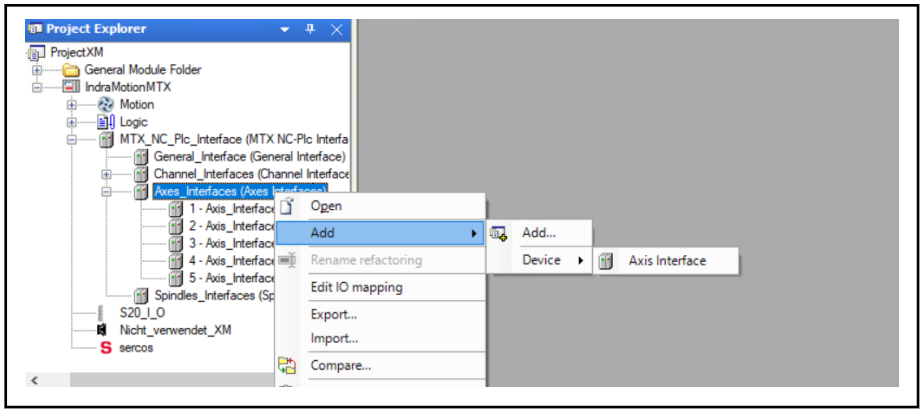


Fig. 4-4: Example: Creating an axis

New axes are continuously created starting with axis 1. The index is a decisive factor for the significance of an axis. Thus, for example, an axis interface with the index 4 is the interface for axis 4.

The symbolic name for output signals is "qAxis_<Node number with leading zero>" and the symbolic name for input signals is "iAxis_<Node number with leading zero>".

Double-click on the "Axis Interface <Node number>" node (or click on "Open" in the context menu) to open the "Properties" dialog.

Use the "Properties" dialog to change symbolic names, addresses and comments as well as the corresponding index.

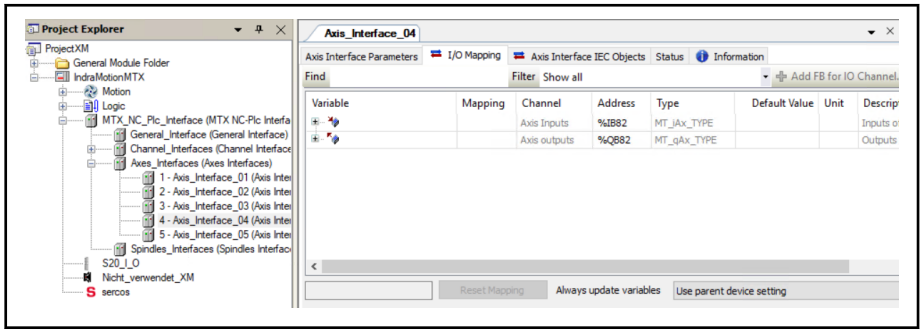


Fig. 4-5: "Properties" dialog for the axis interface

The index for axis interfaces always has to be unique. Therefore, each index may occur only once within the axis interfaces.

4.1.5 Spindle interface

The MTX supports up to 32 spindles. No spindles are defined by default. New spindles are created via the context menu on the "Spindle Interfaces" node.

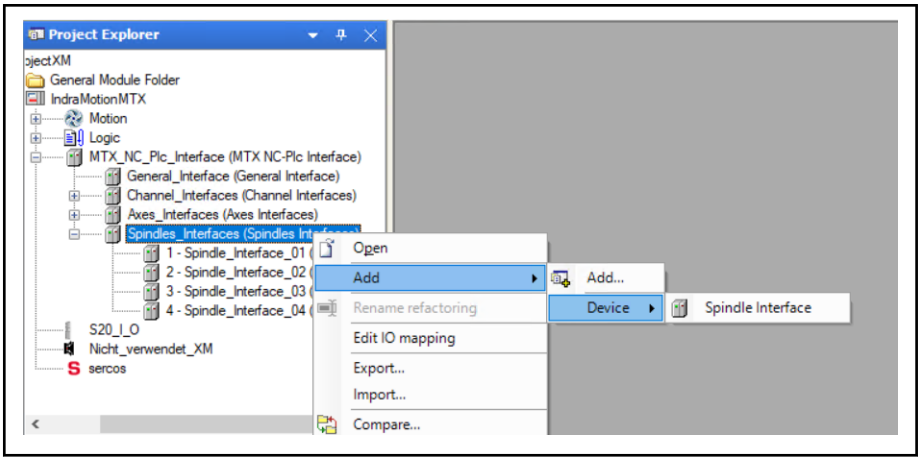


Fig. 4-6: Example: Creating a spindle

New spindles are continuously created starting with spindle 1. The index is the deciding factor for the significance of a spindle. Thus, for example, a spindle interface with the index 6 is the interface for spindle 6.

The symbolic name for output signals is "qSpindle_<Node number with leading zero>" and the symbolic name for input signals is "iSpindle_<Node number with leading zero>".

Double-click on the "Spindle Interface <Node number>" node (or click on "Open" in the context menu) to open the "Properties" dialog.

Use the "Properties" dialog to change symbolic names, addresses and comments as well as the corresponding index.

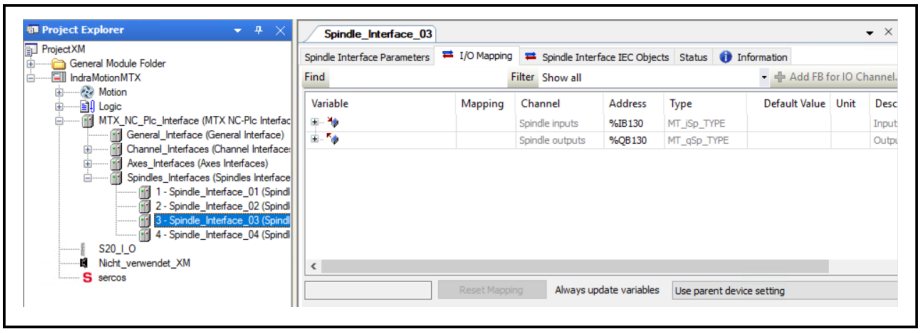


Fig. 4-7: "Properties" dialog for the spindle interface

The index for spindle interfaces always has to be unique. Therefore, each index may occur only once within the spindle interfaces.

5 Overview on interface signals

5.1 Global output signals (PLC --> NC)

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
0.0	qGen_Reset	System reset	1.0	qGen_StrokeInhibit	Stroke inhibit
0.1	-	Reserved	1.1	qGen_StrokeReserv	Stroke reservation
0.2	qGen_DelErrInfo	Clear error info	1.2	qGen_StrokeRel	Stroke on
0.3	qGen_SercosIO_Reset	Clear Sercos I/O device errors	1.3	-	Reserved
0.4	-	Reserved	1.4	-	Reserved
0.5	-	Reserved	1.5	-	Reserved
0.6	-	Reserved	1.6	-	Reserved
0.7	-	Reserved	1.7	-	Reserved
2.0	qGen_Crash	Crash state from PLC to NC (IPO)	3.0	-	Reserved
2.1	qGen_CrashStoreDataAckn	Acknowledgement of crash state from NC to PLC	3.1	-	Reserved
2.2	-	Reserved	3.2	-	Reserved
2.3	-	Reserved	3.3	-	Reserved
2.4	-	Reserved	3.4	-	Reserved
2.5	-	Reserved	3.5	-	Reserved
2.6	-	Reserved	3.6	-	Reserved
2.7	-	Reserved	3.7	-	Reserved

Tab. 5-1: Overview on output signals (PLC --> NC)

5.2 Global input signals (NC --> PLC)

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
0.0	iGen_Reset	Perform system reset	1.0	iGen_StrokeIntend	Stroke intended
0.1	iGen_HardwareState	Hardware warning	1.1	iGen_NoStroke	Stroke not running
0.2	-	Reserved	1.2	-	Reserved
0.3	iGen_SercosIO_Reset	Sercos I/O device errors cleared	1.3	-	Reserved
0.4	-	Reserved	1.4	-	Reserved
0.5	-	Reserved	1.5	-	Reserved
0.6	-	Reserved	1.6	-	Reserved
0.7	-	Reserved	1.7	-	Reserved
2.0	iGen_CrashAckn	Acknowledgement of crash state from PLC to NC (IPO)	3.0	-	Reserved

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
2.1	iGen_CrashStoreData	Crash state from NC to PLC	3.1	-	Reserved
2.2	-	Reserved	3.2	-	Reserved
2.3	-	Reserved	3.3	-	Reserved
2.4	-	Reserved	3.4	-	Reserved
2.5	-	Reserved	3.5	-	Reserved
2.6	-	Reserved	3.6	-	Reserved
2.7	-	Reserved	3.7	-	Reserved

Tab. 5-2: Overview on input signals (NC --> PLC)

5.3 Channel-related output signals (PLC --> NC)

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
0.0	qCh_OpModeSel_00	Sel. operation mode bit 0	1.0	qCh_OpModePlc	PLC operation mode
0.1	qCh_OpModeSel_01	Sel. operation mode bit 1	1.1	qCh_Restart	Automatic restart
0.2	qCh_OpModeSel_02	Sel. operation mode bit 2	1.2	qCh_NCStart	NC start
0.3	qCh_OpModeSel_03	Sel. operation mode bit 3	1.3	qCh_TransferLock	Block transfer inhibit
0.4	-	Reserved	1.4	qCh_FeedHold	Feed hold
0.5	-	Reserved	1.5	qCh_FeedStop	Feed inhibit
0.6	-	Reserved	1.6	qCh_MachineFunction	Machine function
0.7	-	Reserved	1.7	qCh_ReSelOff	Autom. reselection off
2.0	qCh_CtrlReset	Reset	3.0	qCh_ASub1	Asynchr. subroutine 1
2.1	qCh_CancSyntaxError	Clear syntax error	3.1	qCh_ASub2	Asynchr. subroutine 2
2.2	qCh_CancDist	Canceling distance to go	3.2	qCh_ASub3	Asynchr. subroutine 3
2.3	qCh_NextBlk	Switching to next block	3.3	qCh_ASub4	Asynchr. subroutine 4
2.4	-	Reserved	3.4	qCh_ASub5	Asynchr. subroutine 5
2.5	-	Reserved	3.5	qCh_ASub6	Asynchr. subroutine 6
2.6	qCh_RetCont	Return to contour	3.6	qCh_ASub7	Asynchr. subroutine 7
2.7	qCh_Retract	Fast retract	3.7	qCh_ASub8	Asynchr. subroutine 8
4.0	qCh_JogPlusWcs	WCS manual +	5.0	qCh_BlkSlash	Skip block
4.1	qCh_JogMinusWcs	WCS manual -	5.1	qCh_OptStop	Optional stop
4.2	-	Reserved	5.2	qCh_OptJump	Conditional jump
4.3	-	Reserved	5.3	-	Reserved
4.4	qCh_Handw-SelWcs_00	Handwheel selection, Bit0	5.4	qCh_RedRap	Reduced rapid traverse

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
4.5	qCh_Handw-SelWcs_01	Handwheel selection, Bit1	5.5	qCh_Rapid100	Rapid traverse 100%
4.6	qCh_HandwDirWcs	Handwheel direction of rotation	5.6	-	Reserved
4.7	qCh_HandwPosMode	Position handwheel	5.7	qCh_Override100	Override 100%
6.0	qCh_Override_00	Override, bit 0	7.0	qCh_Override_08	Override, bit 8
6.1	qCh_Override_01	Override, bit 1	7.1	qCh_Override_09	Override, bit 9
6.2	qCh_Override_02	Override, bit 2	7.2	qCh_Override_10	Override bit 10
6.3	qCh_Override_03	Override, bit 3	7.3	qCh_Override_11	Override bit 11
6.4	qCh_Override_04	Override, bit 4	7.4	qCh_Override_12	Override bit 12
6.5	qCh_Override_05	Override, bit 5	7.5	qCh_Override_13	Override bit 13
6.6	qCh_Override_06	Override, bit 6	7.6	qCh_Override_14	Override bit 14
6.7	qCh_Override_07	Override, bit 7	7.7	qCh_Override_15	Override bit 15
8.0	qCh_Custom1	Customer input 1	9.0	qCh_OnlCorrWcs	Online correction enabled
8.1	qCh_Custom2	Customer input 2	9.1	qCh_OnlCorrWcsDir	Direction of online correction
8.2	qCh_Custom3	Customer input 3	9.2	qCh_Retrace	Retrace mode
8.3	qCh_Custom4	Customer input 4	9.3	qCh_RetraceMovFwd	Path motion forward
8.4	qCh_Custom5	Customer input 5	9.4	qCh_RetraceMovRev	Path motion backward
8.5	qCh_Custom6	Customer input 6	9.5	-	Reserved
8.6	qCh_Custom7	Customer input 7	9.6	qCh_CoordCoupleOff	Terminate coupling
8.7	qCh_Custom8	Customer input 8	9.7	qCh_TangTRotRel	TTL released
10.0	qCh_RapOverride00	Rapid traverse, override, bit 0	11.0	qCh_RapOverride08	Rapid traverse, override, bit 8
10.1	qCh_RapOverride01	Rapid traverse, override, bit 1	11.1	qCh_RapOverride09	Rapid traverse, override, bit 9
10.2	qCh_RapOverride02	Rapid traverse, override, bit 2	11.2	qCh_RapOverride10	Rapid traverse, override, bit 10
10.3	qCh_RapOverride03	Rapid traverse, override, bit 3	11.3	qCh_RapOverride11	Rapid traverse, override, bit 11
10.4	qCh_RapOverride04	Rapid traverse, override, bit 4	11.4	qCh_RapOverride12	Rapid traverse, override, bit 12
10.5	qCh_RapOverride05	Rapid traverse, override, bit 5	11.5	qCh_RapOverride13	Rapid traverse, override, bit 13
10.6	qCh_RapOverride06	Rapid traverse, override, bit 6	11.6	qCh_RapOverride14	Rapid traverse, override, bit 14
10.7	qCh_RapOverride07	Rapid traverse, override, bit 7	11.7	qCh_RapOverride15	Rapid traverse, override, bit 15
12.0	-	Reserved	13.0	qCH_TestFeed	Test feed
12.1	-	Reserved	13.1	qCH_TestRap	Test rapid traverse

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
12.2	-	Reserved	13.2	qCh_Feedrate2	Second feed
12.3	-	Reserved	13.3	-	Reserved
12.4	-	Reserved	13.4	-	Reserved
12.5	-	Reserved	13.5	-	Reserved
12.6	-	Reserved	13.6	-	Reserved
12.7	-	Reserved	13.7	-	Reserved

Tab. 5-3: Overview on output signals (PLC --> NC)

5.4 Channel-related input signals (NC -> PLC)

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
0.0	iCh_OpMode_00	Sel. operation mode bit 0	1.0	iCh_DryRun	Test mode (dry run)
0.1	iCh_OpMode_01	Sel. operation mode bit 1	1.1	iCh_NCReady	NC ready
0.2	iCh_OpMode_02	Sel. operation mode bit 2	1.2	iCh_ProgRun	Program running
0.3	iCh_OpMode_03	Sel. operation mode bit 3	1.3	iCh_TransferLockAct	Block transfer inhibit active
0.4		Reserved	1.4	iCh_FeedHoldAct	Feed hold active
0.5		Reserved	1.5	-	Reserved
0.6		Reserved	1.6	iCh_ProgStopM0	Program stop M0
0.7		Reserved	1.7	iCh_ProgStopM30	Program stop M30
2.0	iCh_Reset	Channel reset	3.0	iCh_ASub1	Asynchr. subroutine 1
2.1	-	Reserved	3.1	iCh_ASub2	Asynchr. subroutine 2
2.2	iCh_RetpaAct	Return to starting point active	3.2	iCh_ASub3	Asynchr. subroutine 3
2.3	iCh_RetContPath	Return to contour to jog path	3.3	iCh_ASub4	Asynchr. subroutine 4
2.4	iCh_RemoveFinish	Leaving completed	3.4	iCh_ASub5	Asynchr. subroutine 5
2.5	iCh_ReadyReEnter	Ready to return to contour	3.5	iCh_ASub6	Asynchr. subroutine 6
2.6	iCh_ReEnterAct	Re-entry active	3.6	iCh_ASub7	Asynchr. subroutine 7
2.7	-	Reserved	3.7	iCh_ASub8	Asynchr. subroutine 8
4.0	iCh_State_00	Channel state, bit 0	5.0	iCh_BlkSlash	Activate "skip block"
4.1	iCh_State_01	Channel state, bit 1	5.1	iCh_OptStop	Activate optional stop
4.2	iCh_State_02	Channel state, bit 2	5.2	-	Reserved
4.3	iCh_State_03	Channel state, bit 3	5.3	iCh_SRunAct	Block pre-run active
4.4	iCh_State_04	Channel state, bit 4	5.4	iCh_SRunReEnter	Re-entry active
4.5	-	Reserved	5.5	iCh_SRunRepos	Return to contour active

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
4.6	-	Reserved	5.6	iCh_Override0	Override 0%
4.7	-	Reserved	5.7	iCh_Override100	Override 100%
6.0	iCh_Cpl01	CPL customer output 1	7.0	iCh_Cpl09	CPL customer output 9
6.1	iCh_Cpl02	CPL customer output 2	7.1	iCh_Cpl10	CPL customer output 10
6.2	iCh_Cpl03	CPL customer output 3	7.2	iCh_Cpl11	CPL customer output 11
6.3	iCh_Cpl04	CPL customer output 4	7.3	iCh_Cpl12	CPL customer output 12
6.4	iCh_Cpl05	CPL customer output 5	7.4	iCh_Cpl13	CPL customer output 13
6.5	iCh_Cpl06	CPL customer output 6	7.5	iCh_Cpl14	CPL customer output 14
6.6	iCh_Cpl07	CPL customer output 7	7.6	iCh_Cpl15	CPL customer output 15
6.7	iCh_Cpl08	CPL customer output 8	7.7	iCh_Cpl16	CPL customer output 16
8.0	iCh_Custom1	Customer output 1	9.0	iCh_G0Act	Rapid traverse active
8.1	iCh_Custom2	Customer output 2	9.1	iCh_InPosAct	Inpos range 2 active
8.2	iCh_Custom3	Customer output 3	9.2	iCh_G41G141Act	G41/G141 active
8.3	iCh_Custom4	Customer output 4	9.3	iCh_G42G142Act	G42/G142 active
8.4	iCh_Custom5	Customer output 5	9.4	iCh_RetraceAct	Retrace mode active
8.5	iCh_Custom6	Customer output 6	9.5	-	Reserved
8.6	iCh_Custom7	Customer output 7	9.6	iCh_CoordCoupleAct	Coord. Coupling active
8.7	iCh_Custom8	Customer output 8	9.7	iCh_TangTRotCmd	Tool rotation (Tang-Tool)
10.0	iCh_ActFunc01	G70 active	11.0	iCh_ActFunc09	Tool correction active, bit 0
10.1	iCh_ActFunc02	Feed 100% active	11.1	iCh_ActFunc10	Tool correction active, bit 1
10.2	iCh_ActFunc03	G95 active	11.2	iCh_ActFunc11	Tool correction active, bit 2
10.3	iCh_ActFunc04	Axis transformation 2 inactive	11.3	iCh_ActFunc12	Tool correction active, bit 3
10.4	iCh_ActFunc05	Program position active	11.4	iCh_ActFunc13	Tool correction active, bit 4
10.5	iCh_ActFunc06	Thread cycle active	11.5	iCh_ActFunc14	Not assigned
10.6	iCh_ActFunc07	Tapping active	11.6	iCh_ActFunc15	Not assigned
10.7	iCh_ActFunc08	Thread cutting active	11.7	iCh_ActFunc16	Not assigned
12.0	iCh_ActFunc17	Not assigned	13.0	iCh_TestFeed	Test feed active
12.1	iCh_ActFunc18	G96 active	13.1	iCh_TestRap	Test rapid traverse active

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
12.2	iCh_ActFunc19	Not assigned	13.2	iCh_RapOverride0	Rapid traverse override 0%
12.3	iCh_ActFunc20	Not assigned	13.3	iCh_RapOverride100	Rapid traverse override 100%
12.4	iCh_ActFunc21	Not assigned	13.4	iCh_Feedrate2	Second feed
12.5	iCh_ActFunc22	Not assigned	13.5	-	Reserved
12.6	iCh_ActFunc23	Not assigned	13.6	-	Reserved
12.7	iCh_ActFunc24	Not assigned	13.7	-	Reserved

Tab. 5-4: Overview on input signals (NC -> PLC)

5.5 Axis-related output signals (PLC -> NC)

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
0.0	qAx_OpModeSel_00	Axis mode, bit 0	1.0	qAx_TrvLim_00	Limit switch range, bit 0
0.1	qAx_OpModeSel_01	Axis mode, bit 1	1.1	qAx_TrvLim_01	Limit switch range, bit 1
0.2	qAx_JogPlus	Manual +	1.2	qAx_SwLimOff	Hide limit switch
0.3	qAx_JogMinus	Manual-	1.3	-	Reserved
0.4	qAx_JogInch	Incremental step in inch	1.4	-	Reserved
0.5	qAx_JogDia	Incremental step diam-eter	1.5	-	Reserved
0.6	qAx_NextNotch	Next grid position	1.6	-	Reserved
0.7	qAx_Reset	Axis Reset	1.7	qAx_FxStopRel	Cancel fixed stop
2.0	qAx_ SafOpModeSwitch	Mode selection (BA)	3.0	qAx_HandwSel_00	Handwheel selection, bit 0
2.1	qAX_SafDrvLock	Drive inhibit	3.1	qAx_HandwSel_01	Handwheel selection, Bit1
2.2	qAx_SafEnablCtrl	Enabling button	3.2	qAx_HandwDir	Handwheel direction of rotation
2.3	qAx_SafSwitch1	Safety switch 1 (S1)	3.3	qAx_HandwPosMode	Position handwheel
2.4	qAx_SafSwitch2	Safety switch 2 (S2)	3.4	-	Reserved
2.5	qAx_ SafCheckInputState	Check input, safety status	3.5	-	Reserved
2.6	qAx_SafTechState	Status of safety signals	3.6	-	Reserved
2.7	qAx_SafRedTransTime	Shorten transfer time	3.7	-	Reserved
4.0	qAx_ManFeed_00	Manual feed, bit 0	5.0	-	Reserved
4.1	qAx_ManFeed_01	Manual feed, bit 1	5.1	-	Reserved
4.2	qAx_ManFeed_02	Manual feed, bit 2	5.2	qAx_HomeSwitch	Reference switch

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
4.3	qAx_ManFeed_03	Manual feed, bit 3	5.3	-	Reserved
4.4	-	Reserved	5.4	-	Reserved
4.5	-	Reserved	5.5	-	Reserved
4.6	-	Reserved	5.6	-	Reserved
4.7	-	Reserved	5.7	qAx_Override100	Axis override 100 %
6.0	qAx_Override_00	Override, bit 0	7.0	qAx_Override_08	Override, bit 8
6.1	qAx_Override_01	Override, bit 1	7.1	qAx_Override_09	Override, bit 9
6.2	qAx_Override_02	Override, bit 2	7.2	qAx_Override_10	Override bit 10
6.3	qAx_Override_03	Override, bit 3	7.3	qAx_Override_11	Override bit 11
6.4	qAx_Override_04	Override, bit 4	7.4	qAx_Override_12	Override bit 12
6.5	qAx_Override_05	Override, bit 5	7.5	qAx_Override_13	Override bit 13
6.6	qAx_Override_06	Override, bit 6	7.6	qAx_Override_14	Override bit 14
6.7	qAx_Override_07	Override, bit 7	7.7	qAx_Override_15	Override bit 15
8.0	qAx_Custom1	Customer input 1	9.0	-	Reserved
8.1	qAx_Custom2	Customer input 2	9.1	-	Reserved
8.2	qAx_Custom3	Customer input 3	9.2	-	Reserved
8.3	qAx_Custom4	Customer input 4	9.3	-	Reserved
8.4	qAx_Custom5	Customer input 5	9.4	-	Reserved
8.5	qAx_Custom6	Customer input 6	9.5	-	Reserved
8.6	qAx_Custom7	Customer input 7	9.6	-	Reserved
8.7	qAx_Custom8	Customer input 8	9.7	-	Reserved
10.0	qAx_TrqErrOff	Hide standstill error	11.0	qAx_Discharge	Axis discharged
10.1	qAx_LagErrOff	Hide coupling error	11.1	qAx_FrzIpoPos	Keep command position
10.2	qAx_MasterPos	Gantry on master position	11.2	-	Reserved
10.3	qAx_TakeActOffs	Apply actual value offset	11.3	qAx_TrqLim	Torque reduction
10.4	-	Reserved	11.4	-	Reserved
10.5	-	Reserved	11.5	-	Reserved
10.6	-	Reserved	11.6	qAx_DrvOn	Drive on
10.7	qAx_RedDelayErrClass1	Delayed Reduce error reaction	11.7	qAx_DrvLock	Feed inhibit

Tab. 5-5: Overview on output signals (PLC --> NC)

5.6 Axis-related input signals (NC --> PLC)

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
0.0	iAx_RefKnown	Reference point known	1.0	iAx_DistCtrl	Axis before end point
0.1	iAx_RefReached	Reference point reached	1.1	iAx_ProgSpReach	Axis velocity reached
0.2	iAx_TrvCmd	Travel command	1.2	-	Reserved
0.3	iAx_TrvDirNeg	Negative traversing direction	1.3	-	Reserved
0.4	iAx_Run	Axis running	1.4	-	Reserved
0.5	iAx_InPos	Axis in position	1.5	-	Reserved
0.6	iAx_NotchPos	Axis on grid position	1.6	iAx_FxStopReached	Fixed stop reached
0.7	iAx_Reset	Axis is reset	1.7	iAx_FxStopAct	Fixed stop active
2.0	iAx_SafOpMode_00	Safety mode, bit 0	3.0	iAx_PosSwitch1	Position switching point 1
2.1	iAx_SafOpMode_01	Safety mode, bit 1	3.1	iAx_PosSwitch2	Position switching point 2
2.2	iAx_SafOpMode_02	Safety mode, bit 2	3.2	iAx_PosSwitch3	Position switching point 3
2.3	iAx_SafOpMode_03	Safety mode, bit 3	3.3	iAx_PosSwitch4	Position switching point 4
2.4	iAx_SafStatePos	Safe position status	3.4	iAx_PosSwitch5	Position switching point 5
2.5	iAx_ SafCtrlOutputState	Status output of controller	3.5	iAx_PosSwitch6	Position switching point 6
2.6	-	Reserved	3.6	iAx_PosSwitch7	Position switching point 7
2.7	-	Reserved	3.7	iAx_PosSwitch8	Position switching point 8
4.0	iAx_ChIndex_00	Channel number, bit 0	5.0	-	Reserved
4.1	iAx_ChIndex_01	Channel number, bit 1	5.1	-	Reserved
4.2	iAx_ChIndex_02	Channel number, bit 2	5.2	-	Reserved
4.3	iAx_ChIndex_03	Channel number, bit 3	5.3	-	Reserved
4.4	iAx_ChIndex_04	Channel number, bit 4	5.4	-	Reserved
4.5	iAx_ChIndex_05	Channel number, bit 5	5.5	-	Reserved
4.6	-	Reserved	5.6	iAx_Override0	Axis override 0 %
4.7	-	Reserved	5.7	iAx_Override100	Axis override 100 %
6.0	iAx_ScsState00	SCS signal status 0	7.0	iAx_ScsState08	SCS signal status 8
6.1	iAx_ScsState01	SCS signal status 1	7.1	iAx_ScsState09	SCS signal status 9
6.2	iAx_ScsState02	SCS signal status 2	7.2	iAx_ScsState10	SCS signal status 10
6.3	iAx_ScsState03	SCS signal status 3	7.3	iAx_ScsState11	SCS signal status 11

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
6.4	iAx_ScsState04	SCS signal status 4	7.4	iAx_ScsState12	SCS signal status 12
6.5	iAx_ScsState05	SCS signal status 5	7.5	iAx_ScsState13	SCS signal status 13
6.6	iAx_ScsState06	SCS signal status 6	7.6	iAx_ScsState14	SCS signal status 14
6.7	iAx_ScsState07	SCS signal status 7	7.7	iAx_ScsState15	SCS signal status 15
8.0	iAx_Custom1	Customer output 1	9.0	iAx_MasterAxIndex_00	Index of master axis, bit 0
8.1	iAx_Custom2	Customer output 2	9.1	iAx_MasterAxIndex_01	Index of master axis, bit 1
8.2	iAx_Custom3	Customer output 3	9.2	iAx_MasterAxIndex_02	Index of master axis, bit 2
8.3	iAx_Custom4	Customer output 4	9.3	iAx_MasterAxIndex_03	Index of master axis, bit 3
8.4	iAx_Custom5	Customer output 5	9.4	iAx_MasterAxIndex_04	Index of master axis, bit 4
8.5	iAx_Custom6	Customer output 6	9.5	iAx_MasterAxIndex_05	Index of master axis, bit 5
8.6	iAx_Custom7	Customer output 7	9.6	iAx_MasterAxIndex_06	Index of master axis, bit 6
8.7	iAx_Custom8	Customer output 8	9.7	iAx_MasterAxIndex_07	Index of master axis, bit 7
10.0	iAx_TrqExceed	Standstill error	11.0	iAx_DrvErrClass1	Error of state class 1
10.1	iAx_CoupleLag	Lagging error	11.1	iAx_DrvChangeClass2	Modification of state class 2
10.2	iAx_CmdOffsExst	Gantry command value offset active	11.2	iAx_DrvChangeClass3	Modification of state class 3
10.3	iAx_CmdOffsExceed	Compensable gantry command value offset exceeded	11.3	iAx_TrqLim	Torque reduced
10.4	-	Reserved	11.4	iAx_DryRun	Test mode (dry run)
10.5	-	Reserved	11.5	iAx_DrvPower	Enabled for power activation
10.6	-	Reserved	11.6	iAx_DrvReady	Drive ready
10.7	iAx_DelayErrClass1	Delayed error reaction	11.7	iAx_DrvAct	Drive in operation

Tab. 5-6: Overview on input signals (NC -> PLC)

5.7 Spindle-related output signals (PLC -> NC)

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
0.0	qSp_CAxOn	C-axis on	1.0	qSp_TurnCW	Spindle M3 manual
0.1	qSp_CAxOff	C-axis off	1.1	qSp_TurnCCW	Spindle M4 manual
0.2	qSp_JogPlus	Jog spindle M3	1.2	qSp_Stop	Spindle M5 manual
0.3	qSp_JogMinus	Jog spindle M4	1.3	qSp_Orientate	Spindle M19 manual

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
0.4	-	Reserved	1.4	-	Reserved
0.5	-	Reserved	1.5	-	Reserved
0.6	-	Reserved	1.6	-	Reserved
0.7	qSp_Reset	Spindle reset	1.7	-	Reserved
2.0	qSp_SafOpModeS-witch	Mode selection (BA)	3.0	qSp_Gear1Act	Acknowledgement GTS 1
2.1	qSp_SafDrvLock	Drive inhibit	3.1	qSp_Gear2Act	Acknowledgement GTS 2
2.2	qSp_SafEnablCtrl	Enabling button	3.2	qSp_Gear3Act	Acknowledgement GTS 3
2.3	qSp_SafSwitch1	Safety switch 1 (S1)	3.3	qSp_Gear4Act	Acknowledgement GTS 4
2.4	qSp_SafSwitch2	Safety switch 2 (S2)	3.4	-	Reserved
2.5	qSp_ SafCheckInputState	Check input, safety status	3.5	-	Reserved
2.6	qSp_SafTechState	Status of safety signals	3.6	-	Reserved
2.7	qSp_SafRedTran-sTime	Shorten transfer time	3.7	qSp_GearIdleAct	Acknowledgement in idle
4.0	qSp_ManSpeed_00	Jog speed, bit 0	5.0	qSp_SpeedLimit	Speed limit
4.1	qSp_ManSpeed_01	Jog speed, bit 1	5.1	qSp_SValueSD	S-value specified via SD
4.2	qSp_ManSpeed_02	Jog speed, bit 2	5.2	qSp_HomeSwitch	Reference switch
4.3	-	Reserved	5.3	-	Reserved
4.4	-	Reserved	5.4	-	Reserved
4.5	-	Reserved	5.5	-	Reserved
4.6	-	Reserved	5.6	-	Reserved
4.7	-	Reserved	5.7	qSp_Override100	Spindle override 100 %
6.0	qSp_Override_00	Override, bit 0	7.0	qSp_Override_08	Override, bit 8
6.1	qSp_Override_01	Override, bit 1	7.1	qSp_Override_09	Override, bit 9
6.2	qSp_Override_02	Override, bit 2	7.2	qSp_Override_10	Override bit 10
6.3	qSp_Override_03	Override, bit 3	7.3	qSp_Override_11	Override bit 11
6.4	qSp_Override_04	Override, bit 4	7.4	qSp_Override_12	Override bit 12
6.5	qSp_Override_05	Override, bit 5	7.5	qSp_Override_13	Override bit 13
6.6	qSp_Override_06	Override, bit 6	7.6	qSp_Override_14	Override bit 14
6.7	qSp_Override_07	Override, bit 7	7.7	qSp_Override_15	Override bit 15
8.0	qSp_Custom1	Customer input 1	9.0	-	Reserved
8.1	qSp_Custom2	Customer input 2	9.1	-	Reserved
8.2	qSp_Custom3	Customer input 3	9.2	-	Reserved

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
8.3	qSp_Custom4	Customer input 4	9.3	-	Reserved
8.4	qSp_Custom5	Customer input 5	9.4	-	Reserved
8.5	qSp_Custom6	Customer input 6	9.5	-	Reserved
8.6	qSp_Custom7	Customer input 7	9.6	-	Reserved
8.7	qSp_Custom8	Customer input 8	9.7	-	Reserved
10.0	-	Reserved	11.0	-	Reserved
10.1	-	Reserved	11.1	-	Reserved
10.2	-	Reserved	11.2	-	Reserved
10.3	-	Reserved	11.3	-	Reserved
10.4	-	Reserved	11.4	-	Reserved
10.5	-	Reserved	11.5	-	Reserved
10.6	-	Reserved	11.6	qSp_DrvOn	Drive on
10.7	qSp_RedDelayErr-Class1	Delayed Reduce error reaction	11.7	qSp_DrvLock	Spindle inhibit

Tab. 5-7: Overview on output signals (PLC --> NC)

5.8 Spindle-related input signals (NC -> PLC)

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
0.0	iSp_CAxAct	C-axis is active	1.0	iSp_ProgSpReach	Speed reached
0.1	iSp_CAxSwitch	C-axis switching	1.1	iSp_SpLim	Speed limited
0.2	iSp_TurnCmd	Rotation command	1.2	iSp_Stop	Spindle stop
0.3	iSp_TurnDirM4	Direction of rotation M4	1.3	iSp_OrientateFinish	Spindle oriented
0.4	-	Reserved	1.4	iSp_OrientateAct	Orient spindle active
0.5	iSp_InPos	Spindle in position	1.5	-	Reserved
0.6	iSp_PosCtrl	Position control active	1.6	-	Reserved
0.7	iSp_Reset	Spindle reset	1.7	-	Reserved
2.0	iSp_SafOpMode_00	Safety mode, bit 0	3.0	iSp_Gear1Sel	GTS 1 selection
2.1	iSp_SafOpMode_01	Safety mode, bit 1	3.1	iSp_Gear2Sel	GTS 2 selection
2.2	iSp_SafOpMode_02	Safety mode, bit 2	3.2	iSp_Gear3Sel	GTS 3 selection
2.3	iSp_SafOpMode_03	Safety mode, bit 3	3.3	iSp_Gear4Sel	GTS 4 selection
2.4	iSp_SafStatePos	Safe position status	3.4	iSp_GearChange	GTS change
2.5	iSp_ SafCtrlOutputState	Status output of controller	3.5	iSp_IdleSpeed	Idling speed reached
2.6	-	Reserved	3.6	-	Reserved
2.7	-	Reserved	3.7	iSp_GearIdleSel	Idle gear selection
4.0	-	Reserved	5.0	-	Reserved
4.1	-	Reserved	5.1	-	Reserved

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
4.2	-	Reserved	5.2	-	Reserved
4.3	-	Reserved	5.3	-	Reserved
4.4	-	Reserved	5.4	-	Reserved
4.5	-	Reserved	5.5	-	Reserved
4.6	-	Reserved	5.6	iSp_Override0	Spindle override 0 %
4.7	iSp_ToolLimAct	Tool limitations active	5.7	iSp_Override100	Spindle override 100 %
6.0	iSp_ScsState00	SCS signal status 0	7.0	iSp_ScsState08	SCS signal status 8
6.1	iSp_ScsState01	SCS signal status 1	7.1	iSp_ScsState09	SCS signal status 9
6.2	iSp_ScsState02	SCS signal status 2	7.2	iSp_ScsState10	SCS signal status 10
6.3	iSp_ScsState03	SCS signal status 3	7.3	iSp_ScsState11	SCS signal status 11
6.4	iSp_ScsState04	SCS signal status 4	7.4	iSp_ScsState12	SCS signal status 12
6.5	iSp_ScsState05	SCS signal status 5	7.5	iSp_ScsState13	SCS signal status 13
6.6	iSp_ScsState06	SCS signal status 6	7.6	iSp_ScsState14	SCS signal status 14
6.7	iSp_ScsState07	SCS signal status 7	7.7	iSp_ScsState15	SCS signal status 15
8.0	iSp_Custom1	Customer output 1	9.0	iSp_CoupleIndex_00	No. of coupling, bit 0
8.1	iSp_Custom2	Customer output 2	9.1	iSp_CoupleIndex_01	No. of coupling, bit 1
8.2	iSp_Custom3	Customer output 3	9.2	iSp_CoupleIndex_02	No. of coupling, bit 2
8.3	iSp_Custom4	Customer output 4	9.3	-	Reserved
8.4	iSp_Custom5	Customer output 5	9.4	-	Reserved
8.5	iSp_Custom6	Customer output 6	9.5	-	Reserved
8.6	iSp_Custom7	Customer output 7	9.6	-	Reserved
8.7	iSp_Custom8	Customer output 8	9.7	-	Reserved
10.0	iSp_Master	Spindle is master	11.0	iSp_DrvErrClass1	Error of state class 1
10.1	iSp_CoupleErr	Coupling error	11.1	iSp_DrvChangeClass2	Modification of state class 2
10.2	iSp_Synchr1	Synchronous run 1	11.2	iSp_DrvChangeClass3	Modification of state class 3
10.3	iSp_Synchr2	Synchronous run 2	11.3	-	Reserved
10.4	-	Reserved	11.4	iSp_DryRun	Test mode (dry run)
10.5	-	Reserved	11.5	iSp_DrvPower	Enabled for power activation
10.6	-	Reserved	11.6	iSp_DrvReady	Drive ready
10.7	iSp_DelayErrClass1	Delayed error reaction	11.7	iSp_DrvAct	Drive in operation

Tab. 5-8: Overview on input signals (NC -> PLC)

6 Global interface signals

6.1 Overview on global interface signals

6.1.1 General information

Global interface signals are higher level signals used with functions relating to the complete NC.

6.1.2 Overview on output signals (PLC -->NC)

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
0.0	qGen_Reset	System reset	1.0	qGen_StrokeInhibit	Stroke inhibit
0.1	-	Reserved	1.1	qGen_StrokeReserv	Stroke reservation
0.2	qGen_DelErrInfo	Clear error info	1.2	qGen_StrokeRel	Stroke on
0.3	qGen_SercosIO_Reset	Clear Sercos I/O device errors	1.3	-	Reserved
0.4	-	Reserved	1.4	-	Reserved
0.5	-	Reserved	1.5	-	Reserved
0.6	-	Reserved	1.6	-	Reserved
0.7	-	Reserved	1.7	-	Reserved
2.0	qGen_Crash	Crash state from PLC to NC (IPO)	3.0	-	Reserved
2.1	qGen_CrashStoreDataAckn	Acknowledgement of crash state from NC to PLC	3.1	-	Reserved
2.2	-	Reserved	3.2	-	Reserved
2.3	-	Reserved	3.3	-	Reserved
2.4	-	Reserved	3.4	-	Reserved
2.5	-	Reserved	3.5	-	Reserved
2.6	-	Reserved	3.6	-	Reserved
2.7	-	Reserved	3.7	-	Reserved

Tab. 6-1: Overview on output signals (PLC --> NC)

6.1.3 Overview on input signals (NC --> PLC)

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
0.0	iGen_Reset	Perform system reset	1.0	iGen_StrokeIntend	Stroke intended
0.1	iGen_HardwareState	Hardware warning	1.1	iGen_NoStroke	Stroke not running
0.2	-	Reserved	1.2	-	Reserved
0.3	iGen_SercosIO_Reset	Sercos I/O device errors cleared	1.3	-	Reserved
0.4	-	Reserved	1.4	-	Reserved
0.5	-	Reserved	1.5	-	Reserved

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
0.6	-	Reserved	1.6	-	Reserved
0.7	-	Reserved	1.7	-	Reserved
2.0	iGen_CrashAckn	Acknowledgement of crash state from PLC to NC (IPO)	3.0	-	Reserved
2.1	iGen_CrashStoreData	Crash state from NC to PLC	3.1	-	Reserved
2.2	-	Reserved	3.2	-	Reserved
2.3	-	Reserved	3.3	-	Reserved
2.4	-	Reserved	3.4	-	Reserved
2.5	-	Reserved	3.5	-	Reserved
2.6	-	Reserved	3.6	-	Reserved
2.7	-	Reserved	3.7	-	Reserved

Tab. 6-2: Overview on input signals (NC --> PLC)

6.2 Signal description

6.2.1 Output signals (PLC --> NC)

System reset qGen_Reset

This signal is applied to all NC channels. Its effect on the individual channels is identical with a control reset of each single channel.

System reset affects:

- Axis-related signals of the NC to the PLC
- *Channel-related signals of the NC:*
  - Control reset executed
  - NC ready
  - Program running

Signal level

1: The system reset is triggered in **all** channels.

0: No effect.

Delete error information qGen_DelErrInfo

The signal "Clear error info" allows to delete all errors, warnings and message entries in the diagnostic memory of diagnostic overview ("Info" dialog) of MTX standard user interface via PLC. The interface signal has the same effect as the F-key "Delete" if the F-key is set to the "All" state for the filter.

Signal level

1: Deleting all entries of errors, warnings and messages in a diagnostic memory.

0: No effect.

Clearing Sercos I/O device errors qGen_SercosIO_Reset

The "Clear Sercos I/O device errors" signal allows for the PLC to clear all C1D currently pending at the Sercos I/O devices.

Signal level

1: Clear Sercos I/O device errors.

0: No effect.

Stroke inhibit function qGen_StrokeInhibit

With this signal, the PLC inhibits releasing a punching stroke, i.e. the corresponding high-speed output on the high-speed I/O plug-in card is disabled.

Signal level

1: The high-speed output HSO is disabled.

0: No effect.

Stroke reservation qGen_StrokeReserv

With this signal, the PLC reserves a high-speed output (HSO) on the high-speed I/O plug-in card for its own stroke release (see chapter "Stroke On qGen_StrokeRel" on page 33).

Signal level

1: The high-speed output HSO is reserved for the PLC.

0: No effect.

Stroke On qGen_StrokeRel

With the signal **Stroke on** , the PLC instructs the NC to release a punching stroke via the high-speed output HSO on the high-speed I/O plug-in card.

Signal edge/level

0 →1: Releasing punching stroke via the high-speed output HSO using the PLC.

0: No effect.

Punching HS logic

Individual strokes can be released by the PLC.

The PLC can release a stroke by instructing the NC via interface signals.

The bit signals used in the communication among the NC, the punching control and the PLC are shown in the following figure:

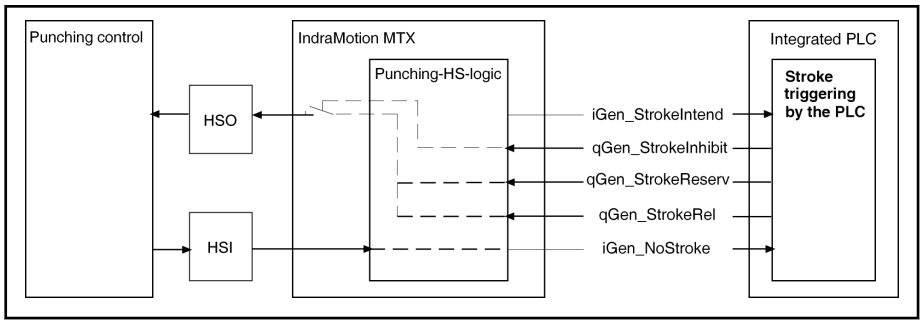


Fig. 6-1: Punching HS logic

The punching HS logic can process five interface signals:

Interface signal	Description
iGen_StrokeIntend	"Stroke intended" The NC informs the PLC that it wants to release a stroke.
iGen_NoStroke	"Stroke not running" The stroke HS logic transmits the high-speed HSI input to the PLC.
qGen_StrokeInhibit	"Stroke inhibit" The PLC inhibits to set the HSO.

Interface signal	Description
qGen_StrokeReserv	"Stroke reservation" The PLC reserves the high-speed output (HSO) for its own stroke release.
qGen_StrokeRel	"Stroke on" The PLC instructs the NC to release a stroke.

Tab. 6-3: Interface signals

A punching stroke can be released from NC and PLC:

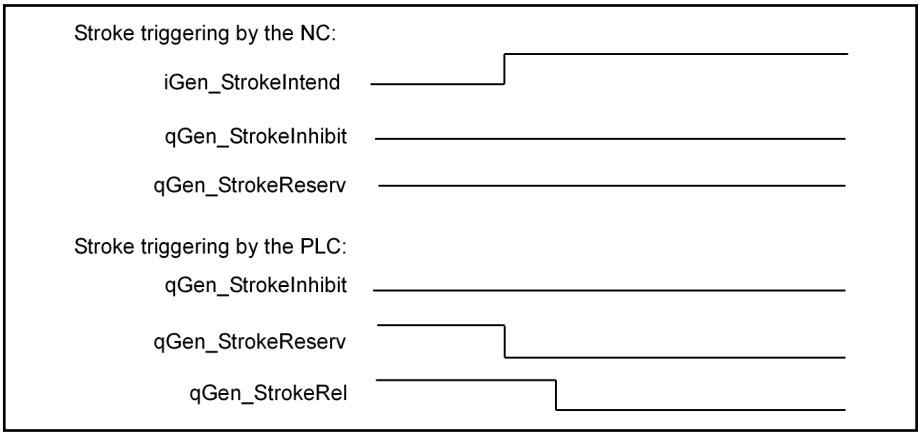


Fig. 6-2: Stroke release by NC

Crash state from PLC to NC (IPO) qGen_Crash

This signal is set by the PLC if it identifies a self-detected crash. The crash trigger "PLC error message" is triggered at the action recorder. The signal can be reset from the PLC after setting the acknowledgement signal "iGen_CrashAckn".

☞ A crash signal from the PLC is processed in the action recorder only if the "PLC error message " entry is active.

Signal edge

0 →1: Crash signaled by PLC
1 → 0: Process completed

Crash state from NC to PLC qGen_CrashStoreDataAckn

Acknowledgement signal of the PLC to the NC that the task was processed. After this signal has been detected, the action recorder resets the signal "iGenCrashStoreData".

☞ A new crash can be identified by the action recorder as soon as the current crash has been processed. A new crash is only indicated to the PLC by an 0 -> 1 edge if this edge has acknowledged the previous error.

Signal level

1: NC task completely successfully.
0: Process completed.

6.2.2 Input signals (NC --> SPS)

System reset executed

This signal is effective together with the system reset signal and the system reset function initiated via the NC user interface (refer also to the signal diagram for system reset on chapter "System reset qGen_Reset" on page 32).

Signal level

1: The reset function has been executed control-internally and all channels are in the switched-on state.

The system reset signal has to be reset by this signal.

Signal level 1 is maintained as long as the channels are in the switched-on state and no program has yet been selected.

0: At least one channel is no longer in the switched-on state, an NC part program is selected or already active.

Hardware warning iGen_HardwareState

*The signal **Hardware Warning** is set if one or more of the following conditions are met:*

- NC module reports "Overtemperature".
- NC module reports "Battery advance warning".

The PLC program can analyze the following system data and a following adequate response (refer to the Functional Description):

- /SysHwStates/BattFail
- /SysHwStates/OverTemp
- /SysStates/BdTemp/CpuTemp

Sercos I/O device errors cleared

*The **iGen_SercosIO_Reset** signal is set if the following two conditions are met:*

- The PLC signal "qGen_SercosIO_Reset" is set.
- The C1D errors reported at the Sercos I/O devices are cleared.

Stroke intended iGen_StrokeIntend

With the **Stroke intended** signal, the NC communicates to the PLC that it wants to release a punching stroke. If the PLC permits the stroke to be released (qGen_StrokeInhibit = 0 and qGen_StrokeReserv = 0), the high-speed output (HSO) on the high-speed I/O plug-in card is set by the punching HS logic (chapter 6.2 "Signal description" on page 32).

Signal level

1: The NC communicates to the PLC that it wants to release a stroke.

0: No effect.

'Stroke not running' iGen_NoStroke

The response signal **stroke not running** from the punching control (high-speed input HSI on the high-speed I/O plug-in card) is forwarded to the PLC.

Signal level

1: Stroke not running = Response signal from punching control to PLC.

0: Stroke running.

Acknowledging the crash state from PLC to NC (IPO) iGen_CrashAckn

The acknowledgement confirms to the PLC that the action recorder received the crash signal. It is reset by the action recorder after it has backed up all data required and if the normal operation is running again

Signal edge

0 →1: Crash was detected and is processed.
1 → 0: Action recorder processed crash.

Crash state from NC to PLC iGen_CrashStoreData

This signal is set by the action recorder if the crash trigger "Sercos drive error (state class 1)" was activated in the interpolator. This signals the PLC that it executes further operations if necessary.

After these operations, the PLC signals hat processing has been completed using the "qGen_CrashStoreDataAckn" signal.



A new crash can be identified by the action recorder as soon as the current crash has been processed. A new crash in only indicated to the PLC by an 0 → 1 edge if the edge has acknowledged the previous crash.

Signal edge

0 →1: A crash occurred. Perform PLC operations.
0: Process completed.

7 Channel-related interface signals

7.1 Overview on channel-related interface signals

7.1.1 General information

The MTX is provided with a **channel structure** that allows processing several tasks at a time.

The channel-related interface signals have mainly an effect on the NC functions related to the processing of the NC part program.

Every channel has its own interface; i.e. with n channels, the interface described in the sections chapter 7.1.2 "Overview on channel-related output signals (PLC --> NC) " on page 37 and chapter 7.1.3 "Overview on input signals (NC -> PLC)" on page 39 is mapped n-times on the respective flags.

The axis-related interface is also significant for the machining axes combined in one channel as well as for the auxiliary axes.

Two channels have to be entered at least:

- Channel 0 for switching functions
- Channel 1 for a machining channel

The following description shows the interface signals for a channel.

7.1.2 Overview on channel-related output signals (PLC --> NC)

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
0.0	qCh_OpModeSel_00	Sel. operation mode bit 0	1.0	qCh_OpModePlc	PLC operation mode
0.1	qCh_OpModeSel_01	Sel. operation mode bit 1	1.1	qCh_Restart	Automatic restart
0.2	qCh_OpModeSel_02	Sel. operation mode bit 2	1.2	qCh_NCStart	NC start
0.3	qCh_OpModeSel_03	Sel. operation mode bit 3	1.3	qCh_TransferLock	Block transfer inhibit
0.4	-	Reserved	1.4	qCh_FeedHold	Feed hold
0.5	-	Reserved	1.5	qCh_FeedStop	Feed inhibit
0.6	-	Reserved	1.6	qCh_MachineFunction	Machine function
0.7	-	Reserved	1.7	qCh_ReSelOff	Autom. reselection off
2.0	qCh_CtrlReset	Reset	3.0	qCh_ASub1	Asynchr. subroutine 1
2.1	qCh_CancSyntaxError	Clear syntax error	3.1	qCh_ASub2	Asynchr. subroutine 2
2.2	qCh_CancDist	Canceling distance to go	3.2	qCh_ASub3	Asynchr. subroutine 3
2.3	qCh_NextBlk	Switching to next block	3.3	qCh_ASub4	Asynchr. subroutine 4
2.4	-	Reserved	3.4	qCh_ASub5	Asynchr. subroutine 5
2.5	-	Reserved	3.5	qCh_ASub6	Asynchr. subroutine 6
2.6	qCh_RetCont	Return to contour	3.6	qCh_ASub7	Asynchr. subroutine 7
2.7	qCh_Retract	Fast retract	3.7	qCh_ASub8	Asynchr. subroutine 8
4.0	qCh_JogPlusWcs	WCS manual +	5.0	qCh_BlkSlash	Skip block

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
4.1	qCh_JogMinusWcs	WCS manual -	5.1	qCh_OptStop	Optional stop
4.2	-	Reserved	5.2	qCh_OptJump	Conditional jump
4.3	-	Reserved	5.3	-	Reserved
4.4	qCh_Handw-SelWcs_00	Handwheel selection, Bit0	5.4	qCh_RedRap	Reduced rapid traverse
4.5	qCh_Handw-SelWcs_01	Handwheel selection, Bit1	5.5	qCh_Rapid100	Rapid traverse 100%
4.6	qCh_HandwDirWcs	Handwheel direction of rotation	5.6	-	Reserved
4.7	qCh_HandwPosMode	Position handwheel	5.7	qCh_Override100	Override 100%
6.0	qCh_Override_00	Override, bit 0	7.0	qCh_Override_08	Override, bit 8
6.1	qCh_Override_01	Override, bit 1	7.1	qCh_Override_09	Override, bit 9
6.2	qCh_Override_02	Override, bit 2	7.2	qCh_Override_10	Override bit 10
6.3	qCh_Override_03	Override, bit 3	7.3	qCh_Override_11	Override bit 11
6.4	qCh_Override_04	Override, bit 4	7.4	qCh_Override_12	Override bit 12
6.5	qCh_Override_05	Override, bit 5	7.5	qCh_Override_13	Override bit 13
6.6	qCh_Override_06	Override, bit 6	7.6	qCh_Override_14	Override bit 14
6.7	qCh_Override_07	Override, bit 7	7.7	qCh_Override_15	Override bit 15
8.0	qCh_Custom1	Customer input 1	9.0	qCh_OnlCorrWcs	Online correction enabled
8.1	qCh_Custom2	Customer input 2	9.1	qCh_OnlCorrWcsDir	Direction of online correction
8.2	qCh_Custom3	Customer input 3	9.2	qCh_Retrace	Retrace mode
8.3	qCh_Custom4	Customer input 4	9.3	qCh_RetraceMovFwd	Path motion forward
8.4	qCh_Custom5	Customer input 5	9.4	qCh_RetraceMovRev	Path motion backward
8.5	qCh_Custom6	Customer input 6	9.5	-	Reserved
8.6	qCh_Custom7	Customer input 7	9.6	qCh_CoordCoupleOff	Terminate coupling
8.7	qCh_Custom8	Customer input 8	9.7	qCh_TangTRotRel	TTL released
10.0	qCh_RapOverride00	Rapid traverse, override, bit 0	11.0	qCh_RapOverride08	Rapid traverse, override, bit 8
10.1	qCh_RapOverride01	Rapid traverse, override, bit 1	11.1	qCh_RapOverride09	Rapid traverse, override, bit 9
10.2	qCh_RapOverride02	Rapid traverse, override, bit 2	11.2	qCh_RapOverride10	Rapid traverse, override, bit 10
10.3	qCh_RapOverride03	Rapid traverse, override, bit 3	11.3	qCh_RapOverride11	Rapid traverse, override, bit 11
10.4	qCh_RapOverride04	Rapid traverse, override, bit 4	11.4	qCh_RapOverride12	Rapid traverse, override, bit 12
10.5	qCh_RapOverride05	Rapid traverse, override, bit 5	11.5	qCh_RapOverride13	Rapid traverse, override, bit 13

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
10.6	qCh_RapOverride06	Rapid traverse, override, bit 6	11.6	qCh_RapOverride14	Rapid traverse, override, bit 14
10.7	qCh_RapOverride07	Rapid traverse, override, bit 7	11.7	qCh_RapOverride15	Rapid traverse, override, bit 15
12.0	-	Reserved	13.0	qCH_TestFeed	Test feed
12.1	-	Reserved	13.1	qCH_TestRap	Test rapid traverse
12.2	-	Reserved	13.2	qCh_Feedrate2	Second feed
12.3	-	Reserved	13.3	-	Reserved
12.4	-	Reserved	13.4	-	Reserved
12.5	-	Reserved	13.5	-	Reserved
12.6	-	Reserved	13.6	-	Reserved
12.7	-	Reserved	13.7	-	Reserved

Tab. 7-1: Overview on output signals (PLC --> NC)

7.1.3 Overview on input signals (NC -> PLC)

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
0.0	iCh_OpMode_00	Sel. operation mode bit 0	1.0	iCh_DryRun	Test mode (dry run)
0.1	iCh_OpMode_01	Sel. operation mode bit 1	1.1	iCh_NCReady	NC ready
0.2	iCh_OpMode_02	Sel. operation mode bit 2	1.2	iCh_ProgRun	Program running
0.3	iCh_OpMode_03	Sel. operation mode bit 3	1.3	iCh_TransferLockAct	Block transfer inhibit active
0.4		Reserved	1.4	iCh_FeedHoldAct	Feed hold active
0.5		Reserved	1.5	-	Reserved
0.6		Reserved	1.6	iCh_ProgStopM0	Program stop M0
0.7		Reserved	1.7	iCh_ProgStopM30	Program stop M30
2.0	iCh_Reset	Channel reset	3.0	iCh_ASub1	Asynchr. subroutine 1
2.1	-	Reserved	3.1	iCh_ASub2	Asynchr. subroutine 2
2.2	iCh_RetpaAct	Return to starting point active	3.2	iCh_ASub3	Asynchr. subroutine 3
2.3	iCh_RetContPath	Return to contour to jog path	3.3	iCh_ASub4	Asynchr. subroutine 4
2.4	iCh_RemoveFinish	Leaving completed	3.4	iCh_ASub5	Asynchr. subroutine 5
2.5	iCh_ReadyReEnter	Ready to return to contour	3.5	iCh_ASub6	Asynchr. subroutine 6
2.6	iCh_ReEnterAct	Re-entry active	3.6	iCh_ASub7	Asynchr. subroutine 7
2.7	-	Reserved	3.7	iCh_ASub8	Asynchr. subroutine 8
4.0	iCh_State_00	Channel state, bit 0	5.0	iCh_BlkSlash	Activate "skip block"
4.1	iCh_State_01	Channel state, bit 1	5.1	iCh_OptStop	Activate optional stop

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
4.2	iCh_State_02	Channel state, bit 2	5.2	-	Reserved
4.3	iCh_State_03	Channel state, bit 3	5.3	iCh_SRunAct	Block pre-run active
4.4	iCh_State_04	Channel state, bit 4	5.4	iCh_SRunReEnter	Re-entry active
4.5	-	Reserved	5.5	iCh_SRunRepos	Return to contour active
4.6	-	Reserved	5.6	iCh_Override0	Override 0%
4.7	-	Reserved	5.7	iCh_Override100	Override 100%
6.0	iCh_Cpl01	CPL customer output 1	7.0	iCh_Cpl09	CPL customer output 9
6.1	iCh_Cpl02	CPL customer output 2	7.1	iCh_Cpl10	CPL customer output 10
6.2	iCh_Cpl03	CPL customer output 3	7.2	iCh_Cpl11	CPL customer output 11
6.3	iCh_Cpl04	CPL customer output 4	7.3	iCh_Cpl12	CPL customer output 12
6.4	iCh_Cpl05	CPL customer output 5	7.4	iCh_Cpl13	CPL customer output 13
6.5	iCh_Cpl06	CPL customer output 6	7.5	iCh_Cpl14	CPL customer output 14
6.6	iCh_Cpl07	CPL customer output 7	7.6	iCh_Cpl15	CPL customer output 15
6.7	iCh_Cpl08	CPL customer output 8	7.7	iCh_Cpl16	CPL customer output 16
8.0	iCh_Custom1	Customer output 1	9.0	iCh_G0Act	Rapid traverse active
8.1	iCh_Custom2	Customer output 2	9.1	iCh_InPosAct	Inpos range 2 active
8.2	iCh_Custom3	Customer output 3	9.2	iCh_G41G141Act	G41/G141 active
8.3	iCh_Custom4	Customer output 4	9.3	iCh_G42G142Act	G42/G142 active
8.4	iCh_Custom5	Customer output 5	9.4	iCh_RetraceAct	Retrace mode active
8.5	iCh_Custom6	Customer output 6	9.5	-	Reserved
8.6	iCh_Custom7	Customer output 7	9.6	iCh_CoordCoupleAct	Coord. Coupling active
8.7	iCh_Custom8	Customer output 8	9.7	iCh_TangTRotCmd	Tool rotation (Tang-Tool)
10.0	iCh_ActFunc01	G70 active	11.0	iCh_ActFunc09	Tool correction active, bit 0
10.1	iCh_ActFunc02	Feed 100% active	11.1	iCh_ActFunc10	Tool correction active, bit 1
10.2	iCh_ActFunc03	G95 active	11.2	iCh_ActFunc11	Tool correction active, bit 2
10.3	iCh_ActFunc04	Axis transformation 2 inactive	11.3	iCh_ActFunc12	Tool correction active, bit 3
10.4	iCh_ActFunc05	Program position active	11.4	iCh_ActFunc13	Tool correction active, bit 4
10.5	iCh_ActFunc06	Thread cycle active	11.5	iCh_ActFunc14	Not assigned

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
10.6	iCh_ActFunc07	Tapping active	11.6	iCh_ActFunc15	Not assigned
10.7	iCh_ActFunc08	Thread cutting active	11.7	iCh_ActFunc16	Not assigned
12.0	iCh_ActFunc17	Not assigned	13.0	iCh_TestFeed	Test feed active
12.1	iCh_ActFunc18	G96 active	13.1	iCh_TestRap	Test rapid traverse active
12.2	iCh_ActFunc19	Not assigned	13.2	iCh_RapOverride0	Rapid traverse override 0%
12.3	iCh_ActFunc20	Not assigned	13.3	iCh_RapOverride100	Rapid traverse override 100%
12.4	iCh_ActFunc21	Not assigned	13.4	iCh_Feedrate2	Second feed
12.5	iCh_ActFunc22	Not assigned	13.5	-	Reserved
12.6	iCh_ActFunc23	Not assigned	13.6	-	Reserved
12.7	iCh_ActFunc24	Not assigned	13.7	-	Reserved

Tab. 7-2: Overview on input signals (NC -> PLC)

☞ Please note that the default assignment of the NC function-specific bit interface signals iCH_ActFunc<nn> can have a different configuration, depending on the machine manufacturer (configuration parameter: /PLC/NcFuncBitIf/NcFunc [1**/PLC/NcFuncBitIf/NcFunc [1]** "iChActFunc [1..24] //NC function assignment \n (max. 8 characters)" (3020 00001) or **/PLC/NcFuncBitIf/NcFunc [1]** "iChActFunc [1..24] //NC function assignment \n (supplement max. 8 characters)" (3020 00002)).

☞ The signal iCh_ActFunc02 was assigned twice in V01 with "Feed 100%" / "G95 active". The dual assignment was deleted.

The following applies in the presetting for V02 or higher:

- iCh_ActFunc02 Feed 100%
- iCh_ActFunc03G95 active

G196 is no longer displayed in iCh_ActFunc18 and has to be configured if necessary.

7.2 Signal description

7.2.1 Channel-related output signals (PLC --> NC)

Operation mode selection qCh_OpModeSel_00 ... 03

The operation mode can be entered using either the PLC or the NC user interface (see qCh_OpModePlc, chapter "Operation mode of PLC qCh_OpModePlc" on page 43).

The following operation modes are possible:

Operation mode	Operation mode selection			
	Bit 3 (qCh_ OpMode-Sel_03)	Bit 2 (qCh_ OpMode-Sel_02)	Bit 1 (qCh_ OpMode-Sel_01)	Bit 0 (qCh_ OpMode-Sel_00)
No operation mode	0	0	0	0
Manual setup	0	0	0	1
Setup approach reference point	0	0	1	0
Reserved	0	0	1	1
MDI: NC block	0	1	0	0
Continuous block	0	1	0	1
Program block	0	1	1	0
Single step	0	1	1	1
Teach	1	0	0	0
Reserved	1	0	0	1
Single block	1	0	1	0
Return to contour	1	0	1	1
CPL debugger program block	1	1	0	0
CPL debugger continuous block	1	1	0	1
Manual setup: workpiece coordinates	1	1	1	0
Reserved	1	1	1	1

Tab. 7-3: Operation mode selection

Switching from one operation mode to another is possible under the following circumstances:

Operation mode		Change in operation mode, channel active		Change in operation mode, channel inactive (iCh_Reset = 1, channel reset)
Continuous block	Automatic mode, group 1	Yes, in group 1	Yes, within group 1 and group 2: Is channel state ready for program start (at beginning or end of program)	Yes
Single step				Yes
Single block				Yes
Program block	Automatic mode, group 2	Yes, in group 2		Yes
CPL debugger program block				Yes
CPL debugger continuous block				Yes

Operation mode	Change in operation mode, channel active	Change in operation mode, channel inactive (iCh_Reset = 1, channel reset)
MDI: NC block	Yes, triggers channel reset executed before changing	Yes
Return to contour		Yes
Manual setup		Yes
Setup approach reference point		Yes
Manual setup: workpiece coordinates		Yes
Teach		Yes

Tab. 7-4: Change in operation mode

Operation mode of PLC qCh_OpModePlc

The operation modes can be selected manually via the NC user interface or via the PLC sequential program.

Signal level

1: The operation mode is specified by the PLC via **qCh_OpModeSel_00...03**.

As long as signal level = 1 is set, no operation mode can be preselected via the NC user interface.

0: The operation mode is specified via the **NC user interface**.

As long as signal level = 0 is set, it is not possible to specify the operation mode via the PLC.

Automatic restart qCh_Restart

Signal level

1: After program end, a program is restarted automatically.

0: No automatic selection.

NC start qCh_NCStart

Applies to the operation modes **Continuous block**, **Single block**, **Single step**, **Program block** and **MDI NC block**.

In connection with NC start, the input signals are NC ready, the feed hold is active and program running are significant.

Signal edge

0 →1: Starts the execution of a part program (continuous block) or an NC block. **Feed hold active** is reset and the axis motion is initiated again.

1 → 0: No effect.

Block transfer inhibit qCh_TransferLock

Applies to the operation modes **Continuous block**, **Single block** and **MDI NC block**.

Signal level

1: The execution of the next NC block is inhibited. However, the active block is executed. Upon completion of the active block, the signal **Block transfer inhibit active** is output.

As long as block transfer inhibit is set, NC start cannot execute the part program or the entered NC block.

The block transfer inhibit does not stop block preparation. It continues until the configured memory for the block preparation is assigned.

0: The processing of an NC - part program interrupted by the block transfer inhibit is continued.

Feed hold qCh_FeedHold

Applies to **all machining axes** of a channel.

Feed hold has no effect on NC blocks programmed without motion (the execution of the NC program is continued until a machining axis is about to perform a traversing motion).

Feed hold is self-sustaining. The input signal **Feed hold active** remains set as long as feed hold is active.

Self-sustainment is canceled by the channel-related interface signal **NC start**.

Signal level

1: Axis motions are stopped with negative path acceleration. Before canceling the self-sustainment, no traversing motions can be started.

Feed hold has **no effect** on thread cutting (G33). In this case, the feed rate depends on the spindle speed and the thread pitch.

0: As soon as "feed hold" is reset, its self-sustainment can be canceled with **NC start** and the next traversing motion can be started.

Feed inhibit qCh_FeedStop

Applies simultaneously to all machining axes assigned to a channel in the operation modes **Continuous block**, **Single block**, **Single step** and **MDI NC block**.

This signal has no effect on auxiliary axes since auxiliary axes are not assigned to any particular channel.

Axis- interface signal feed inhibit has a higher priority than the channel-related signal.

Signal level

1: The axes do not traverse when the signal is set. If the signal is set while the axes are moving, all axes of the channel stop with a negative path acceleration.

Feed inhibit has **no effect** on thread cutting (G33). The feed rate depends on the spindle speed and the thread pitch.

0: If feed inhibit is reset, the axes move again. They accelerate to the current path feed.

Machine function qCh_MachineFunction

Function not yet implemented.

Automatic reselection off qCh_ReSelOff

This signal affects the automatic reselection of a part program.

Signal level

1: Automatic program reselection is suppressed.

0: Automatic reselection permitted.

Control reset qCh_CtrlReset

This signal affects the axes interpolated in **one** channel as well as the part program running on this channel. It has the same effect as the control reset function triggered via the NC user interface.

In connection with the control reset, the channel-related signals **Channel reset**, **NC ready** and **Program running** are significant. Furthermore, control re-

set effects the axis-related signals **Axis in position, axis running** and **Travel command**.

Signal level

1: With, control reset, the NC part program is deselected and the NC functions to the switched-on state defined in the machine parameter **/CHAN/Ch[1]/Ini/ChResetState** "switched-on state after reset" (7060 00020) are reset for example. Furthermore, the machining axes are decelerated up to a standstill.

If the control reset function is ended control-internally, the NC sets signal **Control reset executed**. The PLC resets the signal level to 0.

If the signal edges of all channels are simultaneously set from 0 to 1, a complete control reset is executed. In addition to individual control resets, also auxiliary axes and spindles are decelerated up to standstill and channel-comprehensive machine parameters are applied.

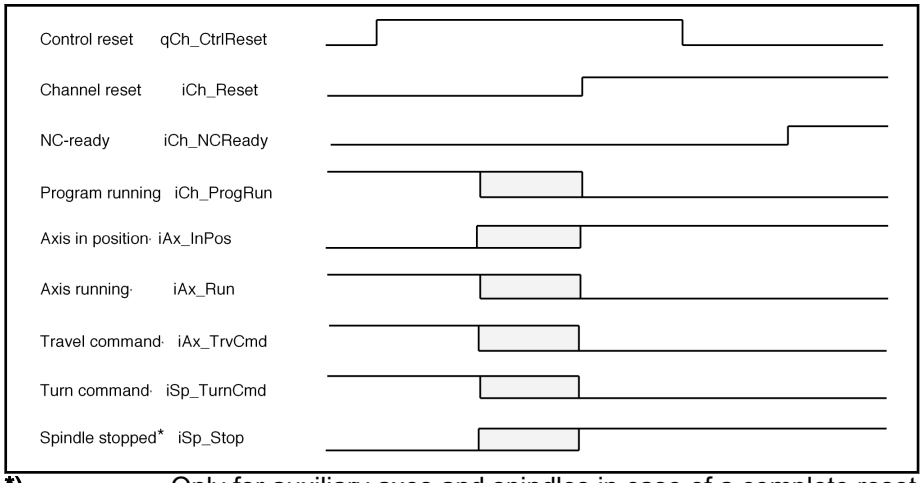
In case of thread cutting (G33), **Control reset** triggers a fast retract (also see chapter "Fast retract qCh_Retract" on page 47).

0: No effect.

Example:

Reset

Signal **Control reset** is set via the channel-related interface to abort the processing of a part program.



*) Only for auxiliary axes and spindles in case of a complete reset

Fig. 7-1: Reset

☞ The "NC ready" signal is not set until an NC part program has been selected.

Deleting syntax errors qCh_CancSyntaxError

Function not yet implemented.

Canceling distance to go qCh_CancDist

Affects the operation modes **Continuous block**, **Single block**, **Single step**, **Program block** with the active program and the **MDI NC block**.

First, the program is stopped using feed hold (qCh_FeedHold; this is not compulsory). The display shows the distance to go the axes have to traverse until block end.

After triggering "Cancel distance to go", all prepared NC blocks including the remaining current block are cancelled and prepared again.

☞ *Note:*

- CPL blocks or CPL parts are not considered:

  Example: The CPL variable POS had the value 10 for preparation: The X[POS] NC word is interpreted as X10 following "Cancel distance to go", even though POS can have a totally different value at this time.

- Any changed correction values are taken into account.

  In the display, the end point is set to the current position and the indicated distance to go is deleted at the same time. The channel then go to the state "NC ready" (iCh_NCReady).

  Following the NC start (iCh_NCStart), the program is continued **at the point of interruption** taking into account the new correction values.

Signal edge

0 →1: Triggers "Cancel distance to go" at active program.

1 → 0: No effect.

☞ This signal must not be set when thread cutting is active.

Switching to next block qCh_NextBlk

When the part program is active, this signal affects the operation modes **Continuous block**, **Single block**, **Single step** and **Program block**.

Signal edge

0 →1: The synchronous axis motion of the channel is stopped with the admissible path acceleration. Subsequently, all prepared NC blocks are cancelled. The channel switches to the **NC ready** state (iCh_NCReady).

After the NC start (qCh_NCStart), the subsequent blocks are prepared again and executed. The part of the interrupted NC block not yet executed is ignored.

To activate exactly this function, stop the program at a defined position first using feed hold.

1 → 0: No effect.

☞ This function only works if the aborted block is followed by a linear block (due to the changed starting position, a consistent circle is no longer achieved using programmed parameters for example).

If the milling cutter path correction is active, the function usually leads to undesired traversing motions and thus to the destruction of the part to be machined.

Returning to contour qCh_RetCont

Signal level/edge

1: Following the NC start, the tool returns to contour on the stored leaving motion.

0: Machining is immediately continued after the NC start.

0 →1: Recording the leaving motion is stopped.

Fast retract qCh_Retract

If thread cutting (G33) is active, the retract motion starts if the cutting motion has been started.

Prerequisite:

The retract motion is enabled via **/THRD/Ch[1]/Retr/DrIndRetrAx, /THRD/Ch[1]/Retr/DrIndSeconAx** "thread cutting: Fast retract, system axis numbers" (7050 00645) - **/THRD/Ch[1]/Retr/RetrVectRetrAx, /THRD/Ch[1]/Retr/RetrVectSeconAx** "thread cutting: Fast retract, distance and direction" (7050 00650) or configured using the function ThreadSet(RD(,)) and **/THRD/Ch[1]/Retr/EnablRetrThrd** "thread cutting: Fast retract, activate" (7050 00640) or enabled using function ThreadSet(RON1).

Signal level

1: Retract from contour is initiated.

A linear retract motion is superimposed on the "secondary cutting axis".This motion moves the cutting tool away from the machined part.

Secondary cutting axis:

- Secondary axis of the plane in case of longitudinal thread
- Main axis of the plane in case of face thread

0: No effect.

Asynchronous subroutine 1 ... 8 qCh_ASub1 ... 8

Each of these signals triggers one of a maximum of eight asynchronous subroutines. The processing of an NC part program is interrupted and the asynchronous subroutine is started immediately. Processing the NC part program can be continued after the asynchronous subroutine is completed.

WCS manual + qCh_JogPlusWcs and WCS manual - qCh_JogMinusWcs

These signals affect the **Manual setup workpiece coordinates** operation mode. For a detailed description of this operation mode, refer to the "Functional Description" manual.

Signal edge

0 →1: If a coordinate is selected in the "Manual setup workpiece coordinates" operation mode, the coordinate starts to move using Manual + or Manual -.

1 → 0: The continuous motion "Manual +" or "Manual -" is stopped.

Handwheel selection bit 0 qCh_HandwSelWcs_00 and handwheel selection bit 1 qCh_HandwSelWcs_01

These signals affect the **Manual setup workpiece coordinates** operation mode. For a detailed description of this operation mode, refer to the "Functional Description" manual.

Two handwheels can be connected to the NC control. As soon as one of the two signals is set, the selected coordinate can be traversed using the selected handwheel. The signals WCS manual + and WCS manual have no effect.

Signal edge

0 →1: If the operation mode and a coordinate is selected, the handwheel mode is activated via "handwheel selection, bit 0" or "handwheel selection, bit 1".

1 → 0: The handwheel mode is deselected. A possible active motion is stopped.

Handwheel direction of rotation qCh_HandwDirWcs

This signal can be used to specify the direction of the coordinate motion irrespective of the handwheel direction of rotation.

Signal level

1: The direction of coordinate motion is reversed.

0: No effect.

Position handwheel qCh_HandwPosMode

This signal is used to specify the behavior of handwheel mode when exceeding the currently allowed maximum velocity.

☞ The signal level is applied by NC when setting qCh_HandwSelWcs_0x.

Signal level

1: The distance specified by the handwheel motions is also processed if the maximum velocity is exceeded.

Attention: Axis motions can also occur after completing the handwheel motion.

0: When exceeding the maximum velocity, the distance to be traversed is reduced accordingly.

Skipping the block qCh_BlkSlash

Applies to the operation modes **Continuous block**, **Program block**, **Single block** and **Single step**.

Signal level

1: At the time of **block preparation**, the NC blocks marked with a slash **"/"** in the NC part program are skipped during processing.

The block preparation can be ahead of the active block by several NC blocks. To ensure that skipping takes place, interrupt the block preparation at an appropriate point (prior to the NC block with the slash /) with regard to the application. The block preparation is interrupted in the part program using the **WAIT** command.

0: The NC block with the slash **"/"** is not skipped.

Optional stop qCh_OptStop

Effective with the function **M1** programmed in the NC part program.

Signal level

1: Processing the NC part program is stopped after the NC block in which function **M1** has been programmed. It can be restarted with **NC start**.

0: Function **M1** is not effective. The NC part program is not stopped.

Conditional jump qCh_OptJump

Effective with the function **GoCond (GOC)** programmed in the NC part program.

Signal level

1: In the NC part program, a jump to the NC block executed. The jump is defined as jump target.

For this purpose, the signal level "logic 1" had already to be detected **at the time of block preparation**.

The block preparation can be ahead of the active block by several NC blocks.

To ensure that the jump is actually performed, interrupt block preparation at an appropriate point (prior to the NC block with the "GoCond" function) with regard to the application. The interruption is executed in the part program using the **WAIT** command.

0: There is no jump in the part program.

Reduced rapid traverse qCh_RedRap

In the operation modes **MDI NC block**, **Continuous block**, **Program block**, **Single step**, rapid traverse G0 is restricted by the setting this signal to a maximum value specified in the machine parameter**/CHAN/Ch[1]/Path/Vel/RedChVel** "Value for restricted rapid mode velocity" (7030 00110).

If the reduction is set after program start, it is only applied after all blocks already prepared at this time have been executed.

If this is not the desired, stop machining with "Feed hold" and cancel the prepared blocks with "Cancel distance to go" before machining can be continued using NC start.

A channel override is also effective in case of a reduced rapid traverse, i.e. 100% override is exactly equivalent to the reduced velocity.

The function is effective for all NC functions traversing in rapid traverse.

The "Rapid traverse" function does not affect the spindle speed or the programmed path feed.

This function has only an effect on the synchronous axes assigned to the channel.

Signal level

1: All rapid traverse blocks are prepared with reduced rapid traverse velocity.

0: The reduced rapid traverse is not active and all rapid traverse blocks are prepared for execution at maximum velocity

Rapid traverse 100% qCh_Rapid100

This signal is only active together with the **Rapid traverse override** function.

Is effective for all axes defined in the channel.

Signal level

1: Irrespective of the actual position of the rapid traverse override, rapid traverse motions travel at maximum possible velocity (if the rapid traverse override function was applied to the channel).

0: The rapid traverse override is active (if the rapid traverse override function was applied to the channel).

Override 100% qCh_Override100

Is effective for all axes defined in the channel.

Signal level

1: Cancels the **Override function**.The axes traverse with the programmed/preselected velocity.

0: The **Override function** is effective.

Override bit 0 ... bit 15 qCh_Override_00 ... 15

The traversing velocities of all machining axes of a channel can be controlled with the **Override function**. The "Override" function affects the velocities programmed in the NC part program and the preselected velocities in setup mode.

qCh_Override_00 ... 15 can be used to continuously specify 16 bits. The MTX interprets the value directly as override value in 0.01%.

☞ Channel 0 defines the values for the auxiliary axes for which the values cannot be set directly.

Customer input 1 ... 8 qCh_Custom1 ... 8

NC interface that can be configured by the user (OEM).

Use cases:

- For OEM-specific firmware
- Reporting an active online correction in the MCS (machine parameters ONL/McsOnlCorr/Ch[k]/Base/ChIfSigMcsCorr)

Online correction enabled qCh_OnlCorrWcs

This signal enables the "Online correction in workpiece coordinates" NC function for a selected coordinate.

In case of an online correction in workpiece coordinates, a handwheel is used to correct the position and orientation in the workpiece coordinate system (WCS) of a channel. This correction is possible with an active and inactive part program. For a detailed description, refer to the "Functional Description" manual.

☞ The online correction cannot be enabled in the "Manual setup", "Manual setup workpiece coordinates" and "Setup approach reference point" operation modes.

Signal level

1: The selected online correction processes the handwheel data.

0 The selected online correction does not respond to the handwheel data.

Direction of online correction qCh_OnlCorrWcsDir

This signal can be used to change the direction of correction.

Signal level

1: A positive direction of rotation of the handwheel results in a **negative** online correction.

0: A positive direction of rotation of the handwheel results in a **positive** online correction.

Retrace mode qCh_Retrace

Activates the retrace mode.

The user can move backward or forward on the drawn contour.

Signal level

1: At edge change, 0 -> 1 the running program is interrupted (internal feed hold). There is no further motion in the part program. Path motions can be backward or forward.

0: At edge change 1 -> 0, the retrace mode is deactivated. Machining can be continued with NC start.

Path motion forward qCh_RetraceMovFwd

Motion in retrace memory forward up to the beginning of the sets stored in the retrace memory.

Condition:

- Retrace mode activated (qCh_Retrace=1)
- Path motion backward deactivated (qCh_RetraceMovRev=0)
- Recorded sets available in retrace memory

Signal level

1: Motion forward

0: Motion is paused

Path motion backward qCh_RetraceMovRev

Motion in retrace memory backward up to the end of the sets stored in the retrace memory.

Condition:

- Retrace mode activated (qCh_Retrace=1)
- Path motion forward deactivated (qCh_RetraceMovFwd=0)
- Recorded sets available in retrace memory

Signal level

1: Motion backward

0: Motion is paused

Coupling off qCh_CoordCoupleOff

This signal ends the coordinate coupling for a slave channel.

Since the master channel has to be at a standstill when the slave is coupled, it is recommended that the master is stopped during coupling with **Block transfer inhibit** and **Feed inhibit**. Both signals can be actively reset by the slave when outputting the signal "coordinate coupling".

When coordinate coupling is completed, the axes of the master channel are to be at standstill.

Signal edge

0 --> 1: The coordinate coupling is ended for a slave channel. The slave channel switches to the next block and the **Coordinate coupling active** signal is set to logic 0.

0: No effect.

TTL released qCh_TangTRotRel

This signal allows the control to execute an intermediate block if the contour knee angle for the tool rotation between two blocks is too large (also see chapter "Tool rotation (TangTool (TTL)) iCh_TangTRotCmd" on page 61).

Signal level

1: Enabled for the control to execute the intermediate block.

0: Enabled to execute intermediate block is terminated. The control can execute additional blocks.

Rapid traverse override bit 0 ... bit 15 qCh_Override_00 ... 15

The traversing velocities of all machining axes of a channel in the rapid traverse can be affected with the **rapid traverse override** function. The "rapid traverse override" function affects the velocities programmed in the NC part program and the preselectable velocities in setup mode.

Above qCh_RapOverride_00 ... 15, a continuous 16 bits specification is possible. The MTX interprets the value directly as override value in 0.01%.

The signal is only evaluated if the machine parameter **/CHAN/Ch[]/Path/RapTravOvrd/EnableRapTravIOvrd** "Enable rapid traverse override" (7030 00014) is set to "Yes".

Test feed qCh_TestFeed

The control uses this signal to ignore the programmed velocity information and moves instead with the velocity value stored in the system date /SysCh-TestMode/TestFeedrate.

The "test feed" function can be activated by the NCS function or using the bit interface of the corresponding channel.

It is switched immediately, since the signals are directly analyzed in the interpolator logic influencing the command feed.

☞ Note, that a certain time passes till the command feed is reached, since the decelerations or accelerations are executed with the parameterized/programmed acceleration.

An increase of the programmed feed using the test feed is only possible if the system date /SysChTestMode/Enable was set at the time of block preparation.

The parameterized dynamic limit values are complied with during active test feed.

The NC reports the currently active state to the NC output interface.

The "test feed" function is effective under G93 (Time programming), G94 (Direct feed programming) and G95 (Feed/rotation). However, the behavior does not depend on the state of G93/G94/G95. If the functionality is active, "test feed" is always scaled with the corresponding feed override. In case of G95, the spindle speed has no effect.

Signal level

1: The control ignores the programmed velocity information and moves with the velocity value stored in the/SysChTestMode/TestFeedrate system date.

0: Test feed is no longer active and the control traversed again with the programmed velocity.

Test rapid traverse qCh_TestRap

The control uses this signal to ignore the programmed velocity information and moves instead with the velocity value stored in the system date /SysChTestMode/TestRapid.

The "Test rapid traverse" function can be activated by the NCS function or via the bit interface of the corresponding channel.

It is immediately switched, since the signals are directly analyzed in the interpolator logic influencing the command rapid traverse.

☞ Note that certain time passes till the command rapid traverse is reached, since decelerations or accelerations are executed with the parameterized/programmed acceleration.

The NC reports the currently active state to the NC output interface.

The "Test rapid traverse" functionality is effective under G93 (Time programming), G94 (Direct feed programming) and G95 (Feed/rotation). However, the behavior does not depend on the state of G93/G94/G95. If the functionality is active, "test rapid traverse" is always scaled with the corresponding feed override. In case of G95, the spindle speed has no effect.

Signal level

1: The control ignores the rapid traverse velocity and moves with the velocity value stored in the/SysChTestMode/TestRapid system date.

0: Test rapid traverse is no longer active. The control traverses again with rapid traverse.

Second feed qCh_Feedrate2

This signal is used to activate the "Feed2" value (the control moves with the velocity value programmed with F2). It is switched immediately, since the signal is directly analyzed in the interpolator logic influencing the command feed.

The NC reports the currently active state to the NC output interface (iCh_Feedrate2).

The functionality "Feed2" is effective under G94 (direct feed programming). Feed2 is scaled with the default feed override and additionally with an own feed override value resulting from the system date SD.SysChCmd[...].Feedrate2Ovr.

Signal level

1: The control traverses with the programmed velocity F2 (Feed2)

0: The control traverses with the programmed velocity F (default feed)

7.2.2 Input signals (NC -> PLC)

Active operation mode iCh_OpMode_00 ... 03

The active operation mode output is coded:

Operation mode	Active operation mode			
	Bit 3 (iCh_ OpMode_03)	Bit 2 (iCh_ OpMode_02)	Bit 1 (iCh_ OpMode_01)	Bit 0 (iCh_ OpMode_00)
No operation mode	0	0	0	0
Manual setup	0	0	0	1
Setup approach reference point	0	0	1	0
Reserved	0	0	1	1
MDI: NC block	0	1	0	0
Continuous block	0	1	0	1
Program block	0	1	1	0
Single step	0	1	1	1
Teach	1	0	0	0
Reserved	1	0	0	1
Single block	1	0	1	0
Return to contour	1	0	1	1
CPL debugger program block	1	1	0	0
CPL debugger continuous block	1	1	0	1
Manual setup: workpiece coordinates	1	1	1	0
Reserved	1	1	1	1

Tab. 7-5: Active operation mode

Test mode (dry run) iCh_DryRun

This signal is set when all axes of the channel are in test mode.

In test mode, axes are still interpolated internally, but the calculated command values are only shown on the display and not transmitted to the drive. The axis is held in the same position as it was when test mode was activated.

The test mode can be automatically activated for unconnected drives (no Sercos ring present) via a softkey or via the machine parameter 1001 00010 **/MAIN/Dr[1]/EnablVirtMode** "Virtual drive".

NC ready iCh_NCReady

The "NC ready" signal is a prerequisite to set the **NC start** signal.

Signal level

1: *It is set if:*

- a program has been selected or, while in MDI mode, a block is activated but not yet started.
- a block is executed in the operation modes "Program block", "Single block" or "Single step" and the next block has not yet started
- a program is completed in the operation modes "Continuous block", "Program block", "Single block" or "Single step"
- the NC functions "Program stop" **M0** or **M1** were executed
- Cancel distance to go (qCh_CancDist) is executed

0: *It is reset if:*

- no NC program is selected
- the NC block is entered
- processing has already been initiated with NC start and the "program running" signal is set.

NC start may not be set while NC ready is reset.

There is only one exception in combination with **Feed hold**. If "Feed hold" was set, NC ready remains 0, but the machining program can nevertheless be started with NC start.

Program running iCh_ProgRun

This is only effective in the operation modes **"Continuous block", "Program block"** and **"Single block"** if a program or in **MDI** mode an NC block has been selected before

Signal level

1: The signal is set when an NC part program has been started with **NC start** in the operation modes "Continuous block", "Program block", "Single block", and "Single step" or when an execution of a single NC block has been initiated with NC start in the **MDI** mode.

The output signals **Feed inhibit**, **Block transfer inhibit** and **Drive off** have no effect on this signal. If **Override = 0** is specified, the signal level remains at logic 1.

0: *It is reset if:*

- a program is selected in the operation modes "Continuous block", "Program block", "Single block" or "Single step"
- a program is selected or, while in MDI mode, a block is activated but not yet started
- a block is executed in the operation modes "Program block", "Single block" or "Single step" and the next block has not yet started
- the NC functions **M0**, **M1**, **M2** or **M30** were executed.
- feed hold (qCh_FeedHold) is present

Block transfer inhibit active iCh_TransferLockAct

This signal indicates that the channel waits - due to the present block transfer inhibit - until the block transfer inhibit is reset.

Signal level

1: After a block transfer inhibit has been activated and the current NC block has been processed at activation, the channel waits for the block transfer inhibit to be reset and sets the "Block transfer inhibit active" signal.

This signal can be used if, for example, **Cancel distance to go** is executed for the transfer of the external correction at block end.

As soon as this signal is set after the block transfer inhibit has been activated, **Cancel distance to go** can be used to delete the prepared blocks so that the transferred correction values are applied from the next block onwards.

0: No block transfer inhibit active.

This signal is not an acknowledgement signal indicating that the block transfer inhibit has been detected by the NC. It indicates only that the block transfer inhibit became active.

Feed hold active iCh_FeedHoldAct

This signal remains set as long as the signal **Feed hold** is self-sustaining.

Signal level

1: The self-sustainment of the "Feed hold" signal is active and no traversing motions of machining axes can be executed until the "Feed hold" signal is reset by **NC start**.

0: "Feed hold" is not active. The traversing motions of machining axes can be executed.

Program stop M0 iCh_ProgStopM0

This signal is output when the NC function **Program stop** (conditional and unconditional) is executed. At the same time, NC ready (iCh_NCReady) is set.

If no NC start (qCh_NCStart) is set before the block is completed, the program stops and has to be restarted using qCh_NCStart.

Signal level

1: *It is set if:*

- an NC block becomes active with "Program stop" (M0).
- an NC block becomes active with a conditional program stop (M1) and the signal "Optional stop" (qCh_OptStop) is set at the same time.

0: *It is reset if:*

- a program is restarted with NC start after "program stop" (M0, M1)
- a program is deselected

Program end M30 iCh_ProgStopM30

This signal is output when the main program is terminated.

Signal level

1: Is set if the NC block is completed with M30 in the main program or if the main program has been fully processed (that is M30 is completed).

0: The signal is reset at NC start (qCh_NCStart) at the beginning of a program or when a program is deselected.

Channel reset iCh_Reset

This signal is active together with the system reset signal and the system reset function initiated via the NC user interface (refer also to the signal diagram for system reset on chapter "Control reset qCh_CtrlReset" on page 44).

Signal level

1: The "control reset" function is carried out internally by the control and the channel is in the switched-on state.

The " control reset" signal has to be reset using this signal.

Signal level 1 remains as long as the channel is in the switched-on state and no program is yet selected.

0: The channel is no longer in the switched-on state. An NC part program is selected or already active.

Returning to approach point active iCh_RetpaAct

Signal level

1: The return to contour motion to the approach point after leaving the contour and returning to the contour, ASUP and block pre-run was started and the approach point is not yet reached.

The starting or end point or the point of interruption can be the selected as starting point of the interrupted block.

0: Upon completion of the return to contour motion, the signal is reset.

Returning to jog path iCh_RetContPath

Signal level

1: The return to contour path "Jog path" is the default setting. The return to contour motion in approach mode "Automatic" and "Single block" traverses backwards along the jog path to the starting point. The starting point corresponds to the end point of the first jogging motion. From the approach point, NC start can be used to traverse to the approach point (starting or end point or point of interruption).

In approach mode "Manual", the selection "Jog path" is ignored.

0: The return to contour path "Direct" is the default setting. The return to contour motion directly leads to the approach point using NC start.

Leaving contour completed iCh_RemoveFinish

Signal level

1: The return to contour motion was completed with a rising edge (qCh_RetCont) or by pressing the softkey "Return to contour". Subsequent traversing motions are not recorded.

0: Leaving motions are recorded up to a maximum number of blocks (specified in the machine parameter **/OPF/ReturnPath/Ch[1]/NofJogSteps** "Minimum shape order for path jerk limitation" (7050 00330).

Ready to return to contour iCh_ReadyReEnter

Signal level

1: In the "Automatic approach" return to contour mode and the "Return to contour" (qCh_RetCont) signal set, return to the stored contour using NC start. In all other cases, machining can be continued via NC start.

0: No automatic approach or continuation of machining is possible.

Re-entry active iCh_ReEnterAct

Signal level

1: Automatic approach to the saved contour (jog path) was started with NC start and the point of return to contour has not yet been reached. The starting point corresponds to the end point of the first jogging motion.

0: Automatic approach has not yet been started or is already completed.

Asynchronous subroutine 1 ... 8 iCh_ASub1 ... 8

Each of these signals is assigned to one of a maximum of eight asynchronous subroutines.

Signal level

1: Asynchronous subroutine No. x active

0: Asynchronous subroutine No. x not active

Channel state bit 0 ... 4 iCh_State_00 .. 04

Depending on the operation mode, the following channel states are output in binary code:

Operation mode	Channel state	Bit 4 (iCh_ State_04)	Bit 3 (iCh_ State_03)	Bit 2 (iCh_ State_02)	Bit 1 (iCh_ State_01)	Bit 0 (iCh_ State_00)
Inactive	Operation mode is not active and a process can be selected.	0	0	0	0	1
Ready	Operation mode is ready and the process can be started.	0	0	0	1	0
Running	Operation mode is active and executes a program or an NC block.	0	0	0	1	1
Internal/reserved	Reserved	0	0	1	0	0
Internal/reserved	Reserved	0	0	1	0	1
Errors	An error occurred in the operation mode. This error can be eliminated only with "Control reset" or "Program deselection".	0	0	1	1	0
Internal/reserved	Reserved	0	0	1	1	1
Reset Active	Control reset is being executed by the channel. No tasks may be accepted anymore until the state switches to "inactive".	0	1	0	0	0
Program selection Running	A program is selected and prepared (e.g. linked).	0	1	0	0	1
Canceling distance to go Preparation	"Cancel distance to go" was triggered. Cleanup continues.	0	1	0	1	0
Canceling distance to go Running	Operation mode is active and processes existing buffers again.	0	1	0	1	1

Operation mode	Channel state	Bit 4 (iCh_ State_04)	Bit 3 (iCh_ State_03)	Bit 2 (iCh_ State_02)	Bit 1 (iCh_ State_01)	Bit 0 (iCh_ State_00)
Ready for Program start	Operation mode is ready and the process is at program beginning and can be started.	0	1	1	0	0
Ready for next block	All blocks at the buffered NC block specification are processed. Waiting for next input.	0	1	1	0	1

Tab. 7-6: Binary-coded active channel states

Activating skip block iCh_BlkSlash

Is part of the character for **Block skip (/)** programmed in the NC part program and the interface signal **Block skip**.

Signal level

1: This is set when "Block skip" was pressed on the NC user interface. The signal remains set until the "Block skip" softkey is pressed again.

The NC internal function **Block skip** is not yet activated by pressing the softkey. To activate the function, the input signal "Block skip" (qCh_BlkSlash) must be set. This can be easily achieved by returning **Activate block skip** to the **Block skip** signal.

All NC blocks in the part program identified with a slash "/" are then skipped.

0: The **Block skip** softkey was not activated.

Activating optional stop iCh_OptStop

Is related to the **M1** function programmable in the NC part program and the signal **Optional stop**.

Signal level

1: Is set when the optional stop softkey is pressed on the NC user interface. The signal remains set until the "Optional stop" softkey is pressed again.

Pressing the softkey only does not activate the NC internal function **Optional stop**. To activate the function, the input signal "Optional stop" (qCh_OptStop) has to be set. This can be easily achieved by relating the **Activate optional stop** to the **Optional stop** signal.

While processing the NC part program, the program execution is interrupted in the block in which the function M1 is programmed. Program execution can be continued by setting NC start.

0: The "Optional stop" softkey was not activated.

iCh_OptJump

Currently, this signal is not supported. The corresponding functionality has not yet been implemented in the MTX interface.

Overview on interface signals for block pre-run

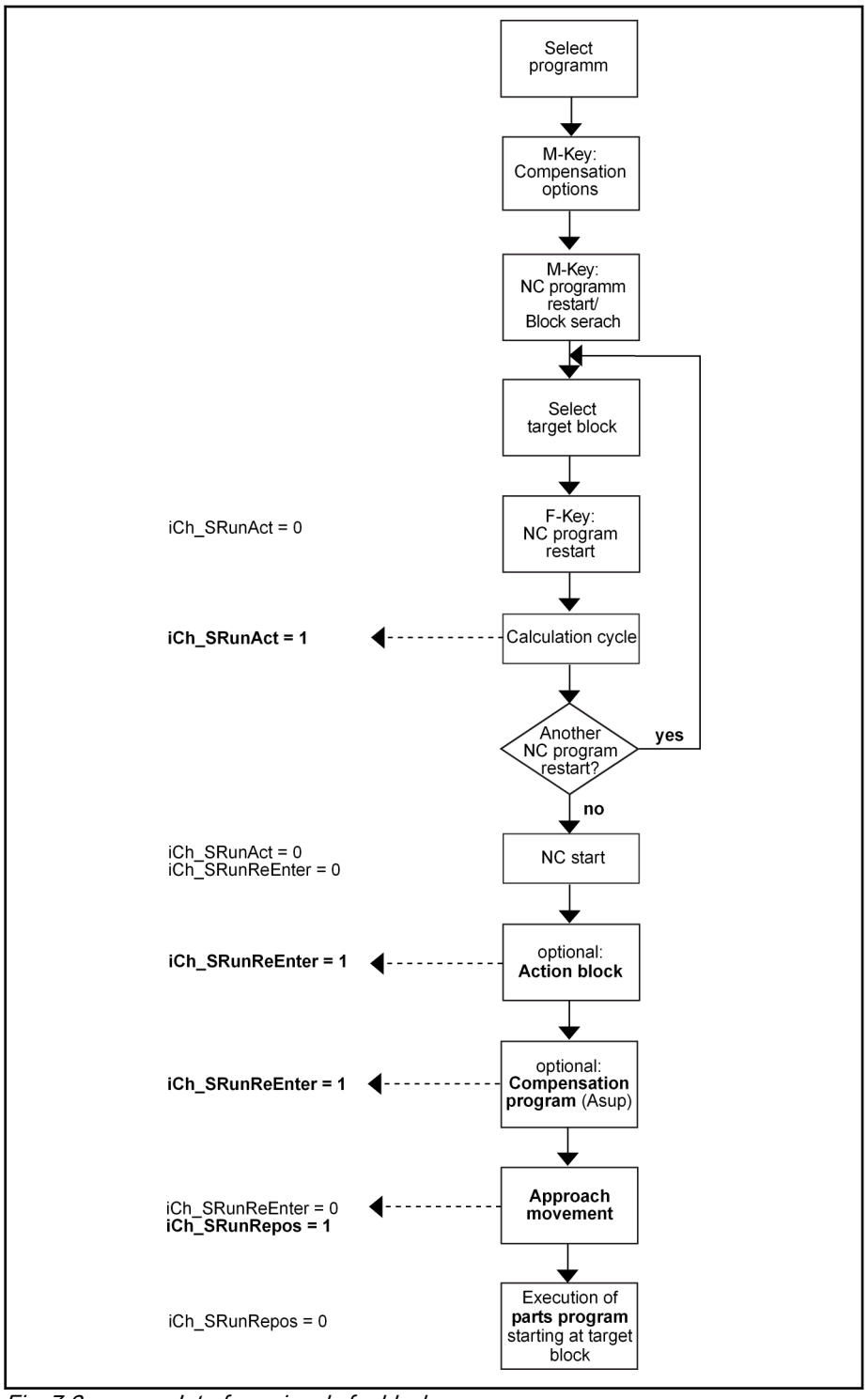


Fig. 7-2: Interface signals for block pre-run

Block pre-run active iCh_SRunAct

The signal has an effect together with the block pre-run functionality.

Signal level

1: During "Computation ", the program is processed until the aborted block (= target block-1).

0: No computation active.

Re-entry active iCh_SRunReEnter

The signal has an effect together with the block pre-run functionality.

Signal level

1: The action block or the adjustment program is processed.

0: No action block and adjustment program are processed.

Return to contour active iCh_SRunRepos

The signal has an effect together with the block pre-run functionality.

Signal level

1: The approaching block to the target block is active.

0: The approaching block is not active.

Override 0% iCh_Override0

This signal acts together with the **Override** function.

The steps of the feed influencing can be scaled with **/CHAN/Ch[1]/Path/Ovrd/ChOvrdStep[1], /AX/AsynchrAxOvrdIf/AxOvrdStep[1]** "Feed override for channel" (7030 00010).

Signal level

1: Signal override 100% (qCh_Override100) is **not** set for the channel.

A possible 100 % feed programmed with "OvrDis"(OVD) has no effect on this signal.

0: Override 100% has been selected using the interface.

Override 100% iCh_Override100

This signal acts together with the **Override** function.

The steps of the feed influencing can be scaled with machine parameter **/CHAN/Ch[1]/Path/Ovrd/ChOvrdStep[1], or /AX/AsynchrAxOvrdIf/AxOvrd-Step[1]** "Feed override for channel" (7030 00010).

Signal level

1: Signal override 100% (qCh_Override100) is set for the channel.

A possible 100 % feed programmed with "OvrDis"(OVD) has no effect on this signal.

0: No Override 100% has been selected using the interface.

CPL customer output 1 ... 16 iCh_Cpl01 ... 16

These 16 signals can be read and written directly from the CPL part program by CPL command "**BITIF(..)**" (see the "Programming Manual").

Customer output 1 ... 8 iCh_Custom1 ... 8

NC configurable interface that can be assigned as required.

(This option is currently not yet active)

Rapid traverse active iCh_G0Act

This signal is set if "Rapid traverse" is modally active (G0 variants).

Signal level

1: The signal is output as long as rapid traversing variants are active.

0: No rapid traverse effective.

Inpos range 2 active iCh_InPosAct

This signal indicates that the "Accurate stop" mode is active in **In-position window 2 (rough)** (e.g. G0(IPS2)) for all of the axes assigned to the channel. Inposition window 2 is specified for each axis with Sercos parameter S-00-261.

If "Accurate stop" modes ..(IPS1) or ..(IPS3) are active, this signal is not set. The positioning window of Sercos parameter S-0-0057, used to form the axis interface signal "Axis in position" (iAx_InPos) is used for the corresponding channel axes.

Signal level
1: **Inposition window 2 (rough)** is active for all axes assigned to the channel.
0: The "normal" positioning window is active for all axes assigned to the channel.

G41/141 active iCh_G41G141Act, G42/142 active iCh_G42G142Act

These signals indicate that the respective function is modally active in the channel. G41/141 and G42/142 mutually exclude each other in programming, i.e. only one of these signals can be active at any time. If neither of the signals is set, no tool radius correction is selected, i.e. G40 and G140 are active.

Signal level
1: G41/141 or G42/142 is active.
0: G41/141 or G42/142 is not active.

Retrace mode Active iCh_RetraceAct

Retrace mode is activated. Backward and forward motions can be carried out.

The signal is deleted after the retrace mode was completed that means qCh_Retrace=0 **and** the machining was continued with NC start.

Signal level
1: Retrace mode active
0: Retrace mode deactivated

Coordinate coupling active iCh_CoordCoupleAct

This signal indicates that the respective channel is coupled to a master channel.

Signal level
1: Coordinate coupling is active.
0: Coordinate coupling is not active.

Tool rotation (TangTool (TTL)) iCh_TangTRotCmd

This signal indicates that the angle between 2 traversing blocks has exceeded the current contour knee angle that has been programmed with TangTool (TTL) (tangential tool leading).

Before executing the internally generated intermediate block, the NC waits until the PLC has set the acknowledgement **TangTool (TTL) enabled (qCh_TangTRotRel)**.

The iCh_TangTRotCmd signal remains set until the execution of the intermediate block is completed.

Signal level
1: The angle between two blocks exceeds the programmed contour knee angle.
The NC is waiting for the PLC to set enabled or executes the intermediate block.
0: The tool rotation does not exceed the contour knee angle.
There is currently no intermediate block that waits for being enables in the NC.

G70 active iCh_ActFunc01

This signal is set if G70 has been selected in the respective channel. The values programmed for the linear synchronous axes are then interpreted as being set in inches.

Signal level

1: G70 is active (inch programming).

0: G71 is active (metric programming).

Feed 100% active iCh_ActFunc02

This signal indicates that the **Set override 100%** function has been programmed for the channel and is applied modally. The Override 100% interface signal has no effect on the **Feed 100% active signal**. If this signal is not set, the OvrEna function (OVR) (Feed 100% off) is active in the channel, which in turn can be superimposed by the Override 100% interface signal.

Signal level

1: The OvrDis (OVD) function is active.

0: The OvrDis (OVD) function is not active.

G95 active iCh_ActFunc03

This signal shows that the speed definition in channel is interpreted in F-word as **feed per rotation**. G95 is selected via basic setting or programming G95.

Signal level

1: Feed per rotation is effective.

0: Feed per rotation is not active.

Axis transformation Inactive iCh_ActFunc04

This signal shows if the axis transformation 2 is switched off.

Signal level

1: Axis transformation 2 is inactive.

0: Axis transformation 2 is active (e.g. lateral cylinder surface or transmit transformation).

Program position active iCh_ActFunc05

This signal indicates that a **SetPos offset** is active in the channel. The SetPos offset is selected by control reset or by programming SetPos (PLC) without any axis addresses.

Signal level

1: A SetPos offset is active.

0: A SetPos offset is not active.

Thread cycle active iCh_ActFunc06

This signal can be set from a part program using the ThreadSet (TST) TCIx function.

☞ The end of the main program (M30) or a control reset cancel the signal.

Signal level

1: ThreadSet (TST) TCI1 is programmed.

0: ThreadSet (TST) TCI0 is programmed or no program is active.

Tapping active iCh_ActFunc07

This signal indicates that the tapping function is active.

Signal level

1: Tapping (G63) is active

0: Tapping is not active.

Example:

The sequence of the most important signals for tapping can be illustrated in an example.

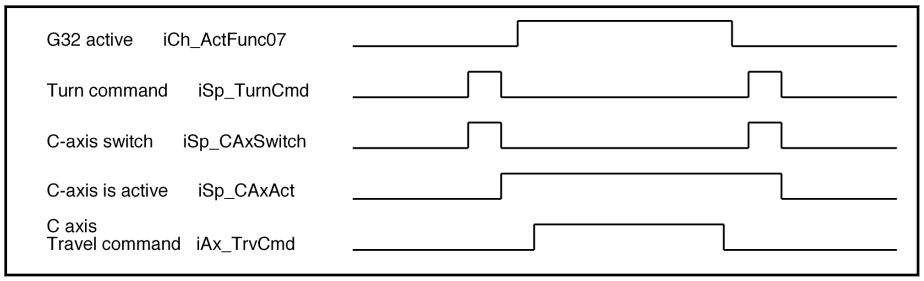


Fig. 7-3: Signals at tapping

Thread cutting active iCh_ActFunc08

This signal is set in case of active thread cutting (G33).

Signal level

1: G33 active. The feed rate depends on the spindle speed and the thread pitch

*The following signal reacts **differently**:*

- qCh_CtrlReset (control reset):

  If configured and enabled, this triggers a fast retract (see qCh_Retract, Fast retract).

*The following signals have **no effect**:*

- qCh_Override100 (Override 100%)
- qCh_Override_00-15 (Override, bit 0...15)
- qCh_FeedStop (feed inhibit)
- qCh_FeedHold (feed hold).

*The following signal must **not** be set:*

- qCh_CancDist (Cancel distance to go)

0: G33 is not active.

Tool correction active bit 0 ... Bit 4 iCh_ActFunc09 ... 13

These signals indicate which tool correction function is active in the channel.

Tool correction	Output signal				
	Bit 4 iCh_ActFunc13	Bit 3 iCh_ActFunc12	Bit 2 iCh_ActFunc11	Bit 1 iCh_ ActFunc10	Bit 0 iCh_ ActFunc09
ED0	0	0	0	0	0
ED1	0	0	0	0	1
ED2	0	0	0	1	0
ED3	0	0	0	1	1
ED4	0	0	1	0	0
ED5	0	0	1	0	1
ED6	0	0	1	1	0
ED7	0	0	1	1	1

Tool correction	Output signal				
	Bit 4 iCh_ActFunc13	Bit 3 iCh_ActFunc12	Bit 2 iCh_ActFunc11	Bit 1 iCh_ ActFunc10	Bit 0 iCh_ ActFunc09
ED8	0	1	0	0	0
ED9	0	1	0	0	1
ED10	0	1	0	1	0
ED11	0	1	0	1	1
ED12	0	1	1	0	0
ED13	0	1	1	0	1
ED14	0	1	1	1	0
ED15	0	1	1	1	1
ED16	1	0	0	0	0

Tab. 7-7: Tool correction active

G96 active iCh_ActFunc18

This signal indicates whether the spindle travels constant cutting velocity (G96) in current the channel.

Signal level

1: Constant cutting velocity is active.

0: Direct speed programming for spindle is active.

Rapid traverse override 0% iCh_RapOverride0

This signal is only active together with the **Rapid traverse override** function.

The steps of the feed influencing can be scaled with **/CHAN/Ch[]/Path/Ovrd/ChOvrdStep[]** "Override value for steps" (7030 00010).

Signal level

1: Signal rapid traverse override 100% (qCh_Rap100) is **not** set for the channel.

A possible 100 % feed programmed with "OvrDis"(OVD) has no effect on this signal.

0: Rapid traverse override 100% was selected via the interface.

Rapid traverse override 100% iCh_RapOverride100

This signal is only active together with the **Rapid traverse override** function.

The stages of the feed influence can be scaled using the machine parameter **/CHAN/Ch[]/Path/Ovrd/ChOvrdStep[]** "Override value for stages" (7030 00010).

Signal level

1: Signal "rapid traverse override 100%" (qCh_Rap100) is not set for the channel.

A possible 100 % feed programmed with "OvrDis"(OVD) has no effect on this signal.

0: No rapid traverse override 100% was selected via the interface.

Second feed iCh_Feedrate2

The NC uses this signal to report whether the Feed2 value is enabled. That means it indicates, whether the control travels with the default feed F or the second feed F2 programmed as velocity value.

Signal level	**1:**	The control traverses with the programmed velocity F2 (Feed2)
	0:	The control traverses with the programmed velocity F (default feed)

8 Axis-related interface signals

8.1 Overview on axis-related interface signals

8.1.1 General information

The axis-related interface signals have an effect on the NC functions associated with controlling a machining or auxiliary axis.

The number of the existing axis interfaces corresponds to the index of the last axis or C-axis/spindle (identical with the physical axis or drive index) entered in the machine parameter **/MAIN/NofDr, /MAIN/Dr[1]/EnablDr, /MAIN/Dr[1]/AxFunc, /MAIN/Dr[1]/SpFunc, /MAIN/Dr[1]/SpFunc/SpInd** "Drive function type" (1001 00001). The axis interface of a C-axis is evaluated or written by the NC only while the C-axis is operated.

If axes are hidden or "simple" spindles are configured before the last set axis or C-axis/spindle, an axis-related interface is mapped to the corresponding flags for them. However, these interfaces are not operated by the NC!

The following description refers to the interface signals for one axis.

8.1.2 Overview on axis-related output signals(PLC --> NC)

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
0.0	qAx_OpModeSel_00	Axis mode, bit 0	1.0	qAx_TrvLim_00	Limit switch range, bit 0
0.1	qAx_OpModeSel_01	Axis mode, bit 1	1.1	qAx_TrvLim_01	Limit switch range, bit 1
0.2	qAx_JogPlus	Manual +	1.2	qAx_SwLimOff	Hide limit switch
0.3	qAx_JogMinus	Manual-	1.3	-	Reserved
0.4	qAx_JogInch	Incremental step in inch	1.4	-	Reserved
0.5	qAx_JogDia	Incremental step diameter	1.5	-	Reserved
0.6	qAx_NextNotch	Next grid position	1.6	-	Reserved
0.7	qAx_Reset	Axis Reset	1.7	qAx_FxStopRel	Cancel fixed stop
2.0	qAx_ SafOpModeSwitch	Mode selection (BA)	3.0	qAx_HandwSel_00	Handwheel selection, bit 0
2.1	qAX_SafDrvLock	Drive inhibit	3.1	qAx_HandwSel_01	Handwheel selection, Bit1
2.2	qAx_SafEnablCtrl	Enabling button	3.2	qAx_HandwDir	Handwheel direction of rotation
2.3	qAx_SafSwitch1	Safety switch 1 (S1)	3.3	qAx_HandwPosMode	Position handwheel
2.4	qAx_SafSwitch2	Safety switch 2 (S2)	3.4	-	Reserved
2.5	qAx_ SafCheckInputState	Check input, safety status	3.5	-	Reserved
2.6	qAx_SafTechState	Status of safety signals	3.6	-	Reserved
2.7	qAx_SafRedTransTime	Shorten transfer time	3.7	-	Reserved

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
4.0	qAx_ManFeed_00	Manual feed, bit 0	5.0	-	Reserved
4.1	qAx_ManFeed_01	Manual feed, bit 1	5.1	-	Reserved
4.2	qAx_ManFeed_02	Manual feed, bit 2	5.2	qAx_HomeSwitch	Reference switch
4.3	qAx_ManFeed_03	Manual feed, bit 3	5.3	-	Reserved
4.4	-	Reserved	5.4	-	Reserved
4.5	-	Reserved	5.5	-	Reserved
4.6	-	Reserved	5.6	-	Reserved
4.7	-	Reserved	5.7	qAx_Override100	Axis override 100 %
6.0	qAx_Override_00	Override, bit 0	7.0	qAx_Override_08	Override, bit 8
6.1	qAx_Override_01	Override, bit 1	7.1	qAx_Override_09	Override, bit 9
6.2	qAx_Override_02	Override, bit 2	7.2	qAx_Override_10	Override bit 10
6.3	qAx_Override_03	Override, bit 3	7.3	qAx_Override_11	Override bit 11
6.4	qAx_Override_04	Override, bit 4	7.4	qAx_Override_12	Override bit 12
6.5	qAx_Override_05	Override, bit 5	7.5	qAx_Override_13	Override bit 13
6.6	qAx_Override_06	Override, bit 6	7.6	qAx_Override_14	Override bit 14
6.7	qAx_Override_07	Override, bit 7	7.7	qAx_Override_15	Override bit 15
8.0	qAx_Custom1	Customer input 1	9.0	-	Reserved
8.1	qAx_Custom2	Customer input 2	9.1	-	Reserved
8.2	qAx_Custom3	Customer input 3	9.2	-	Reserved
8.3	qAx_Custom4	Customer input 4	9.3	-	Reserved
8.4	qAx_Custom5	Customer input 5	9.4	-	Reserved
8.5	qAx_Custom6	Customer input 6	9.5	-	Reserved
8.6	qAx_Custom7	Customer input 7	9.6	-	Reserved
8.7	qAx_Custom8	Customer input 8	9.7	-	Reserved
10.0	qAx_TrqErrOff	Hide standstill error	11.0	qAx_Discharge	Axis discharged
10.1	qAx_LagErrOff	Hide coupling error	11.1	qAx_FrzIpoPos	Keep command position
10.2	qAx_MasterPos	Gantry on master position	11.2	-	Reserved
10.3	qAx_TakeActOffs	Apply actual value offset	11.3	qAx_TrqLim	Torque reduction
10.4	-	Reserved	11.4	-	Reserved
10.5	-	Reserved	11.5	-	Reserved
10.6	-	Reserved	11.6	qAx_DrvOn	Drive on
10.7	qAx_RedDelayErrClass1	Delayed Reduce error reaction	11.7	qAx_DrvLock	Feed inhibit

Tab. 8-1: Overview on output signals (PLC --> NC)

8.1.3 Overview on axis-related input signals (NC --> PLC)

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
0.0	iAx_RefKnown	Reference point known	1.0	iAx_DistCtrl	Axis before end point
0.1	iAx_RefReached	Reference point reached	1.1	iAx_ProgSpReach	Axis velocity reached
0.2	iAx_TrvCmd	Travel command	1.2	-	Reserved
0.3	iAx_TrvDirNeg	Negative traversing direction	1.3	-	Reserved
0.4	iAx_Run	Axis running	1.4	-	Reserved
0.5	iAx_InPos	Axis in position	1.5	-	Reserved
0.6	iAx_NotchPos	Axis on grid position	1.6	iAx_FxStopReached	Fixed stop reached
0.7	iAx_Reset	Axis is reset	1.7	iAx_FxStopAct	Fixed stop active
2.0	iAx_SafOpMode_00	Safety mode, bit 0	3.0	iAx_PosSwitch1	Position switching point 1
2.1	iAx_SafOpMode_01	Safety mode, bit 1	3.1	iAx_PosSwitch2	Position switching point 2
2.2	iAx_SafOpMode_02	Safety mode, bit 2	3.2	iAx_PosSwitch3	Position switching point 3
2.3	iAx_SafOpMode_03	Safety mode, bit 3	3.3	iAx_PosSwitch4	Position switching point 4
2.4	iAx_SafStatePos	Safe position status	3.4	iAx_PosSwitch5	Position switching point 5
2.5	iAx_ SafCtrlOutputState	Status output of controller	3.5	iAx_PosSwitch6	Position switching point 6
2.6	-	Reserved	3.6	iAx_PosSwitch7	Position switching point 7
2.7	-	Reserved	3.7	iAx_PosSwitch8	Position switching point 8
4.0	iAx_ChIndex_00	Channel number, bit 0	5.0	-	Reserved
4.1	iAx_ChIndex_01	Channel number, bit 1	5.1	-	Reserved
4.2	iAx_ChIndex_02	Channel number, bit 2	5.2	-	Reserved
4.3	iAx_ChIndex_03	Channel number, bit 3	5.3	-	Reserved
4.4	iAx_ChIndex_04	Channel number, bit 4	5.4	-	Reserved
4.5	iAx_ChIndex_05	Channel number, bit 5	5.5	-	Reserved
4.6	-	Reserved	5.6	iAx_Override0	Axis override 0 %
4.7	-	Reserved	5.7	iAx_Override100	Axis override 100 %
6.0	iAx_ScsState00	SCS signal status 0	7.0	iAx_ScsState08	SCS signal status 8
6.1	iAx_ScsState01	SCS signal status 1	7.1	iAx_ScsState09	SCS signal status 9
6.2	iAx_ScsState02	SCS signal status 2	7.2	iAx_ScsState10	SCS signal status 10
6.3	iAx_ScsState03	SCS signal status 3	7.3	iAx_ScsState11	SCS signal status 11

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
6.4	iAx_ScsState04	SCS signal status 4	7.4	iAx_ScsState12	SCS signal status 12
6.5	iAx_ScsState05	SCS signal status 5	7.5	iAx_ScsState13	SCS signal status 13
6.6	iAx_ScsState06	SCS signal status 6	7.6	iAx_ScsState14	SCS signal status 14
6.7	iAx_ScsState07	SCS signal status 7	7.7	iAx_ScsState15	SCS signal status 15
8.0	iAx_Custom1	Customer output 1	9.0	iAx_MasterAxIndex_00	Index of master axis, bit 0
8.1	iAx_Custom2	Customer output 2	9.1	iAx_MasterAxIndex_01	Index of master axis, bit 1
8.2	iAx_Custom3	Customer output 3	9.2	iAx_MasterAxIndex_02	Index of master axis, bit 2
8.3	iAx_Custom4	Customer output 4	9.3	iAx_MasterAxIndex_03	Index of master axis, bit 3
8.4	iAx_Custom5	Customer output 5	9.4	iAx_MasterAxIndex_04	Index of master axis, bit 4
8.5	iAx_Custom6	Customer output 6	9.5	iAx_MasterAxIndex_05	Index of master axis, bit 5
8.6	iAx_Custom7	Customer output 7	9.6	iAx_MasterAxIndex_06	Index of master axis, bit 6
8.7	iAx_Custom8	Customer output 8	9.7	iAx_MasterAxIndex_07	Index of master axis, bit 7
10.0	iAx_TrqExceed	Standstill error	11.0	iAx_DrvErrClass1	Error of state class 1
10.1	iAx_CoupleLag	Lagging error	11.1	iAx_DrvChangeClass2	Modification of state class 2
10.2	iAx_CmdOffsExst	Gantry command value offset active	11.2	iAx_DrvChangeClass3	Modification of state class 3
10.3	iAx_CmdOffsExceed	Compensable gantry command value offset exceeded	11.3	iAx_TrqLim	Torque reduced
10.4	-	Reserved	11.4	iAx_DryRun	Test mode (dry run)
10.5	-	Reserved	11.5	iAx_DrvPower	Enabled for power activation
10.6	-	Reserved	11.6	iAx_DrvReady	Drive ready
10.7	iAx_DelayErrClass1	Delayed error reaction	11.7	iAx_DrvAct	Drive in operation

Tab. 8-2: Overview on input signals (NC -> PLC)

8.2 Signal description

8.2.1 Axis-related output signals(PLC -> NC)

General information

Depending on their function, the MTX evaluates the signals either as **static signals** with the signal levels logic 0 and logic 1 or as **edge-controlled signals** with the signal edges 0 --> 1 or 1 --> 0.

Axis operation mode bit 0 and 1 qAx_OpModeSel_00 ... 01

With this signal, one can define for the auxiliary axes whether they shall travel in the operation modes **Manual setup** or **Setup approach reference point** when activating Manual +/- (provided that no channel of the axis is not busy with programming, i.e. it must not move).

The selected operation modes of the channels are not significant, since asynchronous axes are not assigned to any channel.

The operation modes are coded as follows:

Operation modes	Mode selection	
	Bit 1 (qAx_OpModeSel_01)	Bit 0 (qAx_OpModeSel_00)
Programming mode	0	0
Manual setup	0	1
Setup approach reference point	1	0
Reserved	1	1

Tab. 8-3: Coding operation modes

Manual+ qAx_JogPlus, Manual- qAx_JogMinus

The Manual+ and Manual- signals are effective for **channel axes** (synchronous axes) and **auxiliary axes** (asynchronous axes) in the **Manual setup** and **Setup reference point**.

In addition, the signals are used to jog synchronous axes manually when **leaving the contour**. The active part program is stopped with feed hold

Signal level

1:

Continuous jogging:

If the channel/axis mode **Manual setup** is set and all other prerequisites are fulfilled (feed hold, feed inhibit, etc. are not set), the axis is traversed to the appropriate direction using Manual+ or Manual-.

Feed is selected via "Manual feed" (qAx_ManFeed_00 ... 03), the acceleration is parameterized in machine parameter **/AX/Dr[1]/Jog/VarDistAxJog** "Variable incremental step" (1015 00002).

The signals "Travel command" (iAx_TrvCmd), "Axis running" (iAx_Run) and, depending on the travel direction, "Negative travel direction" (iAx_TrvDirNeg) are set on the axis interface. The "Axis in position" (iAx_InPos) signal is cancelled at the same time.

Incremental jogging:

The same as above, but the axis travels the **incremental step** selected using "Manual feed" (qAx_ManFeed_00 ... 03). The axis continues to move even if Manual+ or Manual- is no longer active and the motion along the preselected path has not been completed. The distance will be travelled at the speed entered under **Manual feed Bit 0 ... Bit 3** defined steps and/or variable steps.

A new motion can only be initiated if Manual+ or Manual- had the signal level logic 0 before (edge triggering).

Approach reference point:

If the channel or axis mode **Setup approach reference point** is set and all other prerequisites are met, drive-controlled referencing is triggered for the axis with Manual+ or Manual-. The behavior of the drive is determined in the Sercos reference travel parameter S-0-0147.

0: **Continuous jogging:**

If either the signal Manual+ or Manual- is cancelled, the traversing motion is stopped with the acceleration set in the machine parameters. The travel command is reset. As soon as the axis is located in the inpos window, the "Axis running" signal is reset and "Axis in position" is output.

Example:

Traversing an axis (synchronous or asynchronous) in the positive direction in "Setup" mode.

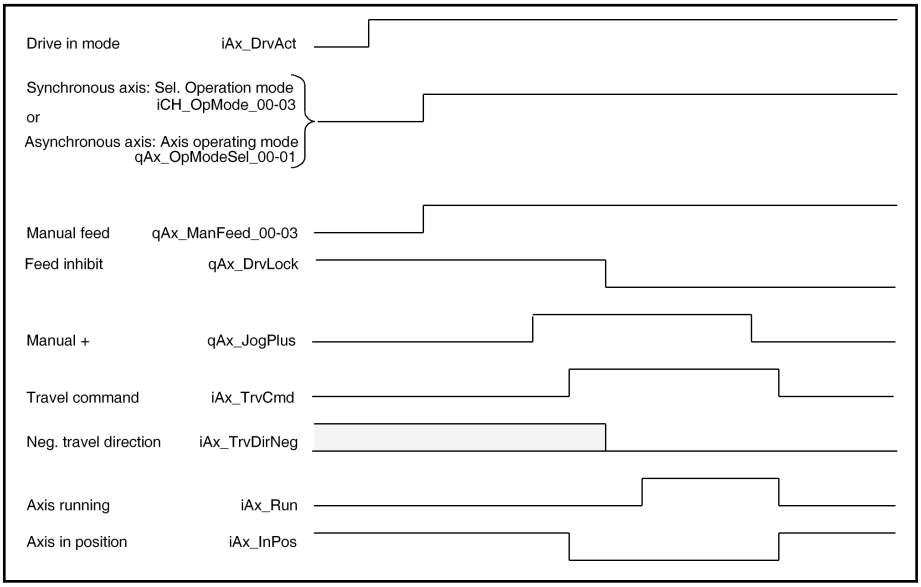


Fig. 8-1: Traversing in positive direction

The control is switched on, the control loop is already closed and the drive generates a torque. **Drive in operation** (iAx_DrvAct) is set.

Setup mode has to be active on the respective channel to traverse an axis **manually** in the positive or negative direction. **Setup** mode is output encoded in the channel-related at the signals iCH_OpMode_00-03.

For auxiliary axes, the mode selection qAx_OpModeSel_00...01 has to be set accordingly. **Manual feed** or **Incremental step** is specified encoded via signals qAx_ManFeed_00-03.

When the **Manual+** signal is set, the **travel command** and **traversing direction** signals are output.

When the **Travel command** is output, the **Axis in position** signal is reset.

As long as **Feed inhibit** is active, there is no traversing motion. When resetting the feed inhibit signal, the axis traverses as long as Manual+ is pressed or until the specified **incremental step** was traversed. While the axis is traversing, the **Axis running** signal is output. If feed inhibit was reset at an earlier point in time, the axis traverse immediately after specifying Manual+.

The **Travel command** signal remains set until the traversing motion is completed. If the axis is at standstill, the **Axis running** signal is reset and the **Axis in position** signal is output if the axis is in the inpos window. The **Traversing**

direction signal remains until the next travel command in opposite direction is output.

Incremental step in inch qAx_JogInch

Setting this signal allows to jog "Incremental steps in inches" or traverse them with the handwheel in setup mode. The incremental step specified via the interface is also interpreted in inches.

The inch increment of an axis results from:

0.0001 * AX/Dr[x]/Jog/IncrAxJog in inches.

☞ The "Incremental step in inch" interface signal is insignificant for rotary and endless axes.

Example:

Machine parameters AX/Dr[x]/Jog/IncrAxJog = 1,000 µm

qAx_JogInch = 1: 1 Incr = 0.0001 Inch

qAx_JogInch = 0: 1 Incr = 0.001 mm

Signal level

1: In jog or handwheel mode, the incremental step specified is interpreted in "inches".

0: In jog or handwheel mode, the incremental step specified is interpreted in the "metric" unit.

Incremental step as diameter qAx_JogDia

Setting this signal allows to jog "incremental steps in diameter" or traverse them with the handwheel in setup mode. The incremental step also specified via the interface is interpreted as diameter difference.

The measuring unit of increments is specified by the signal "Incremental step in inch".

Signal level

1: In jog or handwheel mode, the specified incremental step is interpreted as diameter difference.

0: In jog or handwheel mode, the specified incremental step is interpreted as radius difference.

Next grid position qAx_NextNotch

The control can use this signal to move a Hirth axis to the next grid position. For this purpose, the NC has to be in **manual setup** mode and the Hirth axis must not currently be controlled by **jogging** (iAx_NotchPos set).

The latest active traversing direction (see iAx_TrvDirNeg) determines the direction in which the Hirth axis is moved.

The signal can be used to move a Hirth axis which has come to a standstill between 2 grid positions (e.g. in case of an EMERGENCY STOP) back to a grid position in direction of its current travel direction.

Axis reset qAx_Reset

This signal is used to cancel the motion of an asynchronous axis. After the axis has come to a standstill, the signal **axis is reset** (iAx_Reset).

Signal edge

0 --> 1: The signal **axis reset** (iAx_Reset) is set.

1 --> 0: No effect.

Limit switch range 0 ... 1 qAx_TrvLim_00 ... 01

Selects one of four possible limit switch ranges:

Bit combination 0:	1. Limit switch range active
Bit combination 1:	2. Limit switch range active
Bit combination 2:	3. Limit switch range active (not yet available)
Bit combination 3:	4. Limit switch range active (not yet available)

☞ Signal "Hide limit switches" (qAx_SwLimOff) affects the selected limit switch range.

Hiding limit switch qAx_SwLimOff

This signal hides the positive and negative limit switches axis-by-axis. In the NC, no position limit monitoring is active for this axis.

As long as the reference point of the axis is not known (see iAx_RefKnown), this signal is not relevant. In this case, the NC-internal checks are always disabled.

Monitoring in the drive are not affected by this signal (e.g. S-0-0049 Position limit value positive; S-0-0050 Position limit value negative). Other safety-relevant functions have to be applied for hidden limit switches from the machine and PLC.

The **Hide limit switches** signal is active in all operation modes.

Signal level

1: The software limit switches are hidden.

0: The software limit switches are not hidden. The NC monitors the position limits as defined in the machine parameters according to the selected limit switch range.

Release fixed stop qAx_FxStopRel

Setting this signal releases an active fixed stop for an asynchronous axis.

Operation mode selection qAx_SafOpModeSwitch

This signal is part of the "Integrated Safety engineering according to IEC 61508" function of the IndraDrive series and does not apply to drives with SMO.

The operation mode selection of the PLC is transferred as channel 1 to the drive via the Sercos interface in real time.

Signal level

0: Normal mode NO

1: Special mode SO

(Special mode at standstill/in motion)

Drive inhibit qAx_SafDrvLock

This signal is part of the "Integrated Safety engineering according to IEC 61508" function of the IndraDrive series and does not apply to drives with SMO.

The "Drive inhibit" signal is transferred as channel 1 to the drive using the PLC via the Sercos interface in real time. The drive releases itself from torque when drive inhibit is selected.

Signal level

1: Drive inhibit on

0: Drive inhibit off

Enabling button qAx_SafEnablCtrl

This signal is part of the "Integrated Safety engineering according to IEC 61508" function of the IndraDrive series and does not apply to drives with SMO.

Switching to "Special mode with motion" via the PLC is transferred as channel 1 to the drive via the Sercos interface in real time.

Signal level

1: Special mode with motion

0: Special mode at standstill

Si-switch 1 (S1) qAx_SafSwitch1, Si-switch 2 (S2) qAx_SafSwitch2

These signals are part of the "Integrated Safety engineering according to IEC 61508" function of the IndraDrive series and does not apply to drives with SMO.

Up to four different parameter sets can be stored in the drive for the "special mode with motion". The parameter set selected via PLC is transferred as channel 1 to the drive via Sercos in real time.

Up to MTX14V06, the MTX supports a maximum of 2 save motions. From MTX14V08, a maximum of 4 safe motions can be used in connection with Safety engineering according to IEC 61508.

Please refer to the notes in the drive documentation about "State machine of integrated Safety technology" and the Sercos parameter "P-0-3212 (SI control word, channel 1) bit 3 and bit 4".

☞ If the SI switch 2 information "P-0-3212 (SI control word, channel 1) bit 4" is cyclically transmitted to the drive, MTX four expects configured safe motions!

When using a maximum of 2 safe motions, only the SI switch 1 information "P-0-3212 (SI control word, channel 1) bit 3" is cyclically transmitted to the drive.

	Bit 1 (qAx_SafSwitch2)	**Bit 0 (qAx_SafSwitch1)**
SMM1	0	0
SMM2	0	1
SMM3	1	0
SMM4	1	1

Tab. 8-4: Selecting safe motions

Check input: Safety status qAx_SafCheckInputState

This signal is part of the "Integrated Safety engineering according to IEC 61508" function of the IndraDrive series and does not apply to drives with SMO.

The safe status signal set via PLC is transferred as channel 1 to the drive via the Sercos interface in real time.

Status of safety signals qAx_SafTechState

This signal is part of the "Integrated Safety engineering according to IEC 61508" function of the IndraDrive series and does not apply to drives with SMO.

To perform checks, the control signals of the PLC (qAx_Saf...) transferred via channel 1 to the drive have to be adjusted dynamically. As a result, the PLC sets the signals to "0" for one cycle. At the same time, the PLC sets the safety signal status from "valid" to "made dynamic" and thus informing the drive that the control signals are invalid.

The signals are then reset.

Signal level

1: The control signals of the safety technology are valid.

0: The control signals of the safety technology are made dynamic.

Reducing transmission time qAx_SafRedTransTime

This signal is part of the safety technology of the IndraDrive and ctrlX DRIVE drive series.

This signal allows the PLC to reduce the time of transfer from normal mode to special mode or from special mode to special mode.

When using the function "Integrated Safety engineering according to IEC 61508":

- Please refer to the notes in the drive documentation about "State Machine of the Integrated Safety Technology" and the Sercos parameter "P-0-3212 (SI control word, channel 1), bit 11".

When using the "Safe Motion" function:

- Please refer to the notes in the drive documentation about "Transition functions" and the Sercos parameter "P-0-3261.0.1 (SMO: Control word state machine, functional) bit 0".

Signal level

1: Signals to the drive that the adjustment of the command value system is now complete and that the drive enables monitorings of the selected special mode.

0: No effect

Handwheel selection bit 0 and bit 1 qAx_HandwSel_00 ... 01

Two handwheels can be connected to the NC control. As soon as one of the two signals is set, the axis can be traversed with the selected handwheel. The signals **Manual +** and **Manual -** have no effect.

If the axis is traversed with the handwheel, the signals **Manual feed 0...3** are active. In this context, the selectable incremental steps are used as evaluation to scale the handwheel.

Handwheel	Handwheel assignment	
	Bit 1 (qAx_HandwSel_01)	Bit 0 (qAx_HandwSel_00)
No handwheel active	0	0
Handwheel 1	0	1
Handwheel 2	1	0
Reserved	1	1

Tab. 8-5: Handwheel selection, bit 0 and bit 1

Handwheel direction of rotation qAx_HandwDir

This signal can be used to specify the direction of axis rotation regardless of the direction of handwheel rotation.

Signal level

1: The direction of axis rotation is opposite to the direction of handwheel rotation.

0: The directions of axis and handwheel rotation match.

Position handwheel qAx_HandwPosMode

This signal is used to specify the behavior of handwheel mode when exceeding the currently allowed maximum velocity.

☞ The level of this signal is applied by the NC when qAx_HandwSel_0x .

Signal level

1: The distance specified by the handwheel motions is also processed if the maximum velocity is exceeded.

Attention: Axis motions can also occur after completing the handwheel motion.

0: When exceeding the maximum velocity, the distance to be traversed is reduced accordingly.

Manual feed bit 0 ... bit 3 qAx_ManFeed_00 ... 03

These signals are effective in **Setup** mode. When moving the axes manually using the jog button or the handwheel, the **velocities** or **increments** assigned to the corresponding bit combination are enabled. The velocities are defined in the machine parameters AXSP\Dr[x]\AxFun\Jog\Vel\....

When traversing axes via a handwheel, the **incremental steps as evaluation factor** are analyzed to scale the handwheel.

The incremental steps for **x increments** are specified in the machine parameter **/AX/Dr[1]/Jog/VarDistAxJog** "Variable incremental step" (1015 00002).

Manual feed/ incremental step	Machine Parameters	Bit 3 (qAx_ ManFeed_03)	Bit 2 (qAx_ ManFeed_02)	Bit 1 (qAx_ ManFeed_01)	Bit 0 (qAx_ ManFeed_00)
No specification	-	0	0	0	0
Rapid traverse	**/AX/Dr[1]/Jog/Vel/RapidAxVel** "Manual feed, rapid traverse" (1005 00006)	0	0	0	1
Fast	**/AX/Dr[1]/Jog/Vel/FastAxVel** "Manual feed fast" (1005 00005)	0	0	1	0
Medium	**/AX/Dr[1]/Jog/Vel/MediumAxVel** "Manual feed, medium" (1005 00004)	0	0	1	1
Slow	**/AX/Dr[1]/Jog/Vel/SlowAxVel** "Manual feed, slow" (1005 00003)	0	1	0	0
x increments	**/AX/Dr[1]/Jog/Vel/VarDistAxVel** "Manual feed for variable steps" (1005 00008)	0	1	0	1

1000 increments	**/AX/Dr[1]/Jog/Vel/DefStepAxVel** "Manual feed for defined steps" (1005 00007)	0	1	1	0
100 increments		0	1	1	1
10 increments		1	0	0	0
1 increment		1	0	0	1

Tab. 8-6: Manual feed, Bit 0 ... Bit 3 - Incremental step

Reference switch qAx_HomeSwitch

Transmits the signal of the reference switch to the drive if it is not directly wired at the drive. However, this has to be configured in the drive.

- The signal control word S-0-0145 has to be contained in the master data telegram (S-0-1050.0.6)
- Bit 0 of S-0-0400 has to be configured in the signal control word (S-0-0145)

Signal level

1: The reference switch is active.

0: The reference switch is inactive.

Axis override 100% qAx_Override100

This signal hides the axis override of an asynchronous axis. In this case, the axis traverses at the preselected or programmed velocity. The signal refers to programmed asynchronous motions, external command specification and setup mode (jogging).

The signal does not affect the motions interpolated by the drive (e.g. approaching reference point).

Signal level

1: Axis override is hidden.

0: Axis override is active.

Override bit 0 ... bit 15 qAx_Override_00 ... 15

The traversing velocity of asynchronous axes can be influenced using the **override function**. It affects the velocities programmed in the **part program** and the preselected velocities in **Setup** mode.

This signal is used to enable one override scaling level for an auxiliary axis, if the override is neither hidden via the interface (qAx_Override100) nor via the machine parameters.

The levels of the override switch are queried by the NC in **binary coded** form. **Channel 0 (!)** assigns a value to each binary code via the machine parameter **/CHAN/Ch[1]/Path/Ovrd/ChOvrdStep[1], /AX/AsynchrAxOvrdIf/AxOvrdStep[1]** "Feed override for channel" (7030 00010). The assigned value interprets the respective velocity as a percentage value.

Customer input 1 ... 8 qAx_Custom1 ... 8

NC configurable interface that can be assigned as required.

(This option is currently not yet active)

Hiding standstill error qAx_TrqErrOff

This signal suppresses an internal error state if the standstill torque is exceeded. In this case, monitoring of the standstill torque has to be programmed in the PLC.

☞ Irrespective of this signal, the "Standstill error" (iAx_TrqExceed) signal is set when the standstill torque is exceeded!

Signal level

1: If the standstill torque is exceeded, the NC **does not** generate an internal error state. The overrun is tolerated.

0: Error state when exceeding the standstill torque.

Hiding coupling error qAx_LagErrOff

This signal suppresses an internal error state for gantry axes if the coupling overrun is exceeded. In this case, monitoring of the coupling overrun has to be programmed in the PLC.

☞ Irrespective of this signal, the signal "coupling lag" (iAx_CoupleLag) is set of the coupling lag is exceeded!

Signal level

1: The NC **does not** generate an internal error state if the coupling overrun is exceeded. The overrun is tolerated.

0: Coupling lag limit monitoring generates an error.

Gantry on master position qAx_MasterPos

⚠ CAUTION **The signal may only be set if the encoders have been commissioned.**

Otherwise, mechanical damages can result.

For a Gantry slave, this signal is used to transfer the target position of the master axis without any offset as target position of the slave axis. The signal affects the slave axis.

Prerequisites:

- Master axis and slave axis have known actual position values
- Difference of positions less than the maximum following error configured in the machine parameters.
- The signal is also effective for incremental measuring systems with distance-coded encoders if referencing with "Travel distance" is set.
- *The signal applies under following conditions:*
  - Drive in reference.
  - Maximum following error not exceeded or following error monitoring deactivated.

In case of cyclically absolute encoders, the axis positions are not known prior to referencing. Thus, due to safety reasons, the distance between the gantry axes determined from the actual position values provided by the encoders remains until referencing.

For axes with absolute value encoders, the actual position values are also known without referencing. If the encoders are commissioned, a possible inclination can be detected from the values they provide.

Signal edge

0 --> 1: The inclined position of the slave axis is compensated interpolating as soon as the drives are in operation and all requirements are met. If the requirements are not met, a warning is issued (see MTX Extension, Operating Manual) and the edge is discarded. After the requirements have been corrected, the edge has to be specified again.

Applying actual value offset qAx_TakeActOffs

Applying actual value offset of this slave as command value offset.

Signal edge

0 --> 1: Applying actual value offset at open controller.

The command value offset of a Gantry slave is created from the control of the current actual value offset and follows as long as the controller is open. When the controller closes, the value is frozen.

The signal has to be set for the slave axis.

Reducing delayed error reaction qAx_RedDelayErrClass1

This signal is related with iAx_DelayErrClass1 (delayed error reaction).

The PLC can use this signal to reduce the time up to the standstill of the axis by the NC.

☞ For further information, refer to the chapter "Delayed NC error reaction for non-fatal drive errors" in the "Functional Description" manual.

Signal edge

0 --> 1: The delayed error reaction is completed. The NC stops the axis.

Discharging axis qAx_Discharge

When this signal is set, the axis is no longer checked within the Inpos logic. Nevertheless, the axis can remain in the configuration of a channel if the signal **Drive on** has additionally not been set.

Freezing target position qAx_FrzIpoPos

This signal prevents the actual drive position from being corrected with the position control loop open ("Drive under control" (iAx_DrvAct) is not set).

Signal level

1: The actual position of the drive is not applied to the NC when the position control loop is open.

0: The NC applies the actual drive position when the position control loop is open.

Torque reduction qAx_TrqLi

This signal is used to change the torque limit (**Sercos parameter S-0-0092**) valid in the drive.

This signal is of particular significance with regard to clampable axes While the axis is clamped, the position control loop can remain closed, but the torque of the drive can be reduced to minimize mechanical stress. To a large extent, this prevents mechanical displacement when clamping is applied or released.

The changed torque limit value is set individually axis-by-axis via the machine parameter **RedMaxTorq** "Reduced torque at active torque reduction" (1003 00010) and can also be programmed via the RedTorque (RDT) NC function.

The torque can only be switched at axis standstill.

If the drive does not accept the torque limit (e.g. parameter is write-protected), no error message is displayed. However, the signal "Torque limited" (iAx_TrqLim) is not set.

Signal edge

0 --> 1: The torque reduction is transmitted to the drive via the Sercos interface.

1 --> 0: The original torque limit value from the Sercos parameter S-0-0092 is transmitted to the drive.

Drive On qAx_DrvOn

This signal enables the torque in the drive if it is ready for operation (iAx_DrvReady set). The control loop is closed.

Resetting **Drive on** initiates a shutdown. If the axis is still moving, it is set to standstill at the preset deceleration ramp.

At standstill, the position control loop is opened and the **Drive in operation** signal is reset. Torque is no longer applied to the drive.

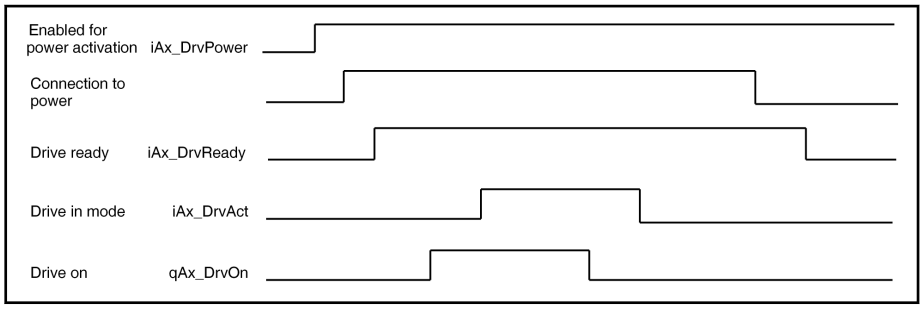


Fig. 8-2: Drive on

Example:

Mode of operation of switching on and off

When the machine is started or voltage is applied, the NC control, the servo drive and the spindle drive start with the initialization phase.

During the initialization phase of the individual components, the digital interface to communicate between the NC and the drives is established.

This digital interface corresponds to the settings of the **Sercos interface**.

The initialization of the **Sercos interface** consists of four system-internal automatic phases. After the interface initialization and once the cyclic operation is active, the **Enabled for power activation** signal is output.

Power can be switched on for the corresponding drives. When the power was connected, the **Drive ready** signal is output.

The position control loop is closed by setting **Drive on**. The drive is under torque and the **Drive in operation** signal is output.

Switching off or disconnection from the power is initiated by the PLC sequential program resetting the **Drive on** signal.

If the axis is still moving, it is set to standstill at the preset deceleration ramp.

At standstill, the position control loop is opened and the **Drive in operation** signal is reset. Torque is no longer applied to the drive.

The power can be switched off in the next step.

The **Enabled for power activation** signal remains set as long as the system components (NC and drives) are initialized.

Feed inhibit qAx_DrvLock

This signal is effective in each operation mode if the control loop is closed and the torque is applied to the axis.

It prevents a programmed or manually preset axis motion. However, it does no influence the "Travel command", "Negative traversing direction" and "Axis in position" input signals.

Signal level

1: Either the axis cannot be traversed or it is stopped with the parameterized deceleration if any motion is already there.

If **Feed inhibit** is set for an axis participating in an interpolation with other axes, the other axes cannot be traversed either.

0: Axis feed is not inhibited. If an axis traversing command is initiated (via the NC part program or manually), the axis can start traversing immediately.

If a traveling axis was previously set to standstill with **Feed inhibit**, it starts accelerating to its programmed velocity with reset.

Example:

Feed inhibit

The machine is switched on and the control loop has already been closed. A negative traversing motion has been programmed for an axis via the NC part program.

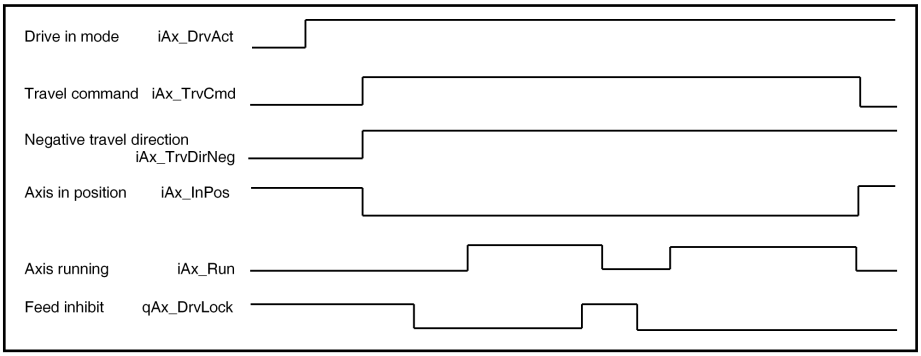


Fig. 8-3: Drive in operation

The **Drive in operation** signal informs the PLC sequential program that the control loop is closed.

Travel command and **neg. traversing direction** signals are output as soon as the NC block (in which the traversing motion is programmed) is executed. The traversing direction is only up-to-date if a travel command is present.

The **Axis in position** signal is reset when the travel command is activated.

There is no traversing motion until the **Feed inhibit** signal is set.

If "Feed inhibit" is reset, the programmed traversing motion is performed and **Axis running** is set.

If **Feed inhibit** is set again during the traversing motion, the traversing motion is stopped and **Axis running** is reset when the axis is at standstill.

When **Feed inhibit** is reset, the axis accelerates up to the previously applied velocity, **Axis running** is set and the traversing motion is completed.

When the programmed end position is reached, the **Travel command** and **Axis running** signals are reset and the **Axis in position**signal is set.

8.2.2 Input signals (NC --> PLC)

Known reference point iAx_RefKnown

This signal is set as soon as the reference point of the axis is approached or, in the case of an absolute encoder, this is detected by the control. The control evaluates the Sercos parameter **Status actual position value** S-0-0403 for this purpose. The NC does not monitor any limit switches as long as the **Reference point known** signal has not yet been set.

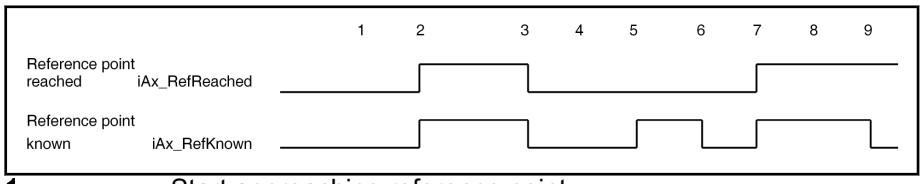
After this signal has been set once, it is only reset if a new Sercos phase startup is initiated or a malfunction of the drive is present (e.g. defective encoder).

If the signal has to display the current state **S-0-403, Actual position value status** at any time, bit 0 of S-0-403 has to be transferred cyclically from the drive to the NC via S-0-144.

Signal level

1: The reference point was approached or the axis is equipped with absolute encoders.

0: No reference point has been approached yet.



1 Start approaching reference point
2 Approaching reference point completed
3 Start approaching reference point
4 Abort approaching reference point (drive identified reference point)
5 Reset
6 Start approaching reference point
7 Approaching reference point completed
8 Drive lost reference point due to an error
9 Reset

Fig. 8-4: Reference point

Reference point reached iAx_RefReached

In contrast to the **Reference point known** signal, this signal is set only if traveling to the reference point has been completed successfully. If traveling to the reference point was cancelled, neither of the two signals is set.

While a subsequent control reset checks the status of the actual position values in the drive (parameter S-0-0403) and sets (or clears) the **Reference point known** signal, the **Reference point reached** signal remains unchanged.

This signal is reset at the beginning of each reference point approach.

Signal level

1: The last reference point approach was properly completed when the reference point was reached.

0: Either no reference points have been approached so far, or traveling to the reference point is still active and the reference point has not yet been reached, or the last reference point approach was cancelled.

Travel command iAx_TrvCmd

The signals **Travel command**, **Neg. traversing direction**, **Axis running** and **Axis in position** affect each other.

In the signal diagrams and explanations on the signals **Manual+/-** (Manual+ qAx_JogPlus, page 71) and (Feed inhibit qAx_DrvLock, page 81), the relations are exemplarily shown.

In relation with the **Travel command** signal. the signal **Neg. traveling direction** is output.

Signal level

1: A traversing motion is specified. It can be performed either manually or via the NC part program.

0: No command is present for a traversing motion.

Negative traversing direction iAx_TrvDirNeg

This signal is effective only with the **Travel command** signal. If no travel command is set, this signal is irrelevant.

Signal level

1: If "Negative traversing direction" and "Travel command" are set, the respective axis has to execute a traversing motion in negative direction or the previous motion of this axis was executed in negative direction.

0: If "Negative traversing direction" is not set and "Travel command" is set, the respective axis has to execute a traversing motion in positive direction or the previous motion of this axis was executed in positive direction.

Axis running iAx_Run

The signals **Travel command**, **Neg. traversing direction**, **Axis running** and **Axis in position** affect each other.

In the signal diagrams and explanations on the signals **Manual+/-** (Manual+ qAx_JogPlus, page 71) and (Feed inhibit qAx_DrvLock, page 81), the relations are exemplarily shown.

If the traversing motion of an axis was completed on NC side, no **travel command** is active anymore and either the signal **Axis in position** or the signal **Axis running** is output.

Signal level

1: "Axis running" remains set as long as the axis executes a programmed or manually preset motion.

This signal is not set in case of **drive-controlled reference point approach** even if the axis is still moving.

0: The axis does not move.

Axis in position iAx_InPos

The signal is active if the control loop is closed and torque is applied to the axis.

Signal level

1: The axis is located in the parameterized inpos window and no travel command is present. The inpos window is specified by the Sercos parameter S-0-0057.

The "In position rough" window is specified by the Sercos parameter S-0-0261, for Diax/EcoDrive and HNC100 by the machine parameter **/NCO/InPosWin/[1]/InPosRough** "In position window" (1015 00100).

0: The axis is not in the parameterized inpos window.

Axis on grid position iAx_NotchPos

This signal indicates when the Hirth axis has reached a grid position.

The signal is 0 as long as the reference point was not approached. This signal is set to 1 as soon as the axis is in the selected inpos window.

Signal level

1: The Hirth axis is in the selected inpos window.

0: Indicates that the Hirth axis has not yet approached a reference point.

Axis reset iAx_Reset

This signal indicates that the asynchronous axis accepted the control reset task and that the asynchronous axis is ready to accept new motion specifications.

Axis in front of end point iAx_DistCtrl

This signal indicates that the axis is in a range in front of its end point defined via the machine parameter **/SFU/AxDistCtrl/Dr[1]/EnablAxDistCtrl, /SFU/AxDistCtrl/Dr[1]/DistEndPoint** "Distance to end point" (1015 00010).

Signal level

1: Axis target position in the specified distance.

0: Axis target position outside the specified distance or not configured or approaching reference point is active or the handwheel is active.

Axis velocity reached iAx_ProgSpReach

Signal level

1: The asynchronous axis reached the programmed speed or modulo velocity <Axis>=S<Value>. The position of the override switch is taken into account.

0: The speed or the modulo velocity of the asynchronous axis has not yet been reached or the velocity interpolation was cancelled with <Axis>=S0 or an positioning process.

Fixed stop reached iAx_FxStopReached

This signal is output as long as the axis is positioned at a fixed stop.

The signal is reset when the axis leaves the 'fixed stop' monitoring window (machine parameter **/MPS/Dr[1]/MaxDevPoStop** "Fixed stop monitoring window" (1003 00032)) or when fixed stop is canceled with "FsReset" (FSR).

Signal level

1: Axis reached the fixed stop.

0: Fixed stop left or cancelled.

Fixed stop active iAx_FxStopAct

This signal is output as long as the traversing motion to the fixed stop is active.

A motion to the fixed stop is triggered by the "FsProbe" (FSP, Measuring at fixed stop) or "FsMove" (FSM, Move to fixed stop) functions.

Signal level

1: Axis moves to fixed stop.

0: No traversing motion to fixed stop active.

Si operation mode bit 0 ... bit 3 iAx_SafOpMode_00 ... 03

These signals are part of the safety technology of the IndraDrive and ctrlX DRIVE drive series.

The NC reads the selected operation mode of the drive (P-0-3215 or P-0-3262) and passes the information on to the PLC via these outputs.



The safe motions SMM14 to SMM16 are not displayed in the bit interface.

Safety function	Safety mode			
	Bit 3 (iAx_ SafOpMode_03)	Bit 2 (iAx_ SafOpMode_02)	Bit 1 (iAx_ SafOpMode_01)	Bit 0 (iAx_ SafOpMode_00)
Normal mode NO	0	0	0	0
Safe Stop 1 (EMERGENCY STOP) (SS1ES)	0	0	0	1
Safe Stop 1 (SS1) / Safe Stop 2 (SS2)	0	0	1	0
Safe motion (SMM1)	0	0	1	1
Safe motion (SMM2)	0	1	0	0
Safe motion (SMM3)	0	1	0	1
Safe motion (SMM4)	0	1	1	0
Safe motion (SMM5)	0	1	1	1
...				
Safe motion (SMM13)	1	1	1	1

Tab. 8-7: Safety mode

Status of safe position iAx_SafStatePos

This signal is part of the safety technology of the IndraDrive and ctrlX DRIVE drive series.

The drive uses this signal to report whether safe referencing has been performed. This is a requirement for the safety function "Safe absolute position" .

The signal is generated from P-0-3256, SMO encoder function: Status evaluation formed.

Signal level

1: The drive is safely referenced.

0: No safe referencing.

Status output of controller iAx_SafCtrlOutputState

This signal is part of the "Integrated Safety engineering according to IEC 61508" function of the IndraDrive series.

The drive uses this signal to provide feedback about its safe state.

Signal level

1: Safe drive state.

0: No safe drive state.

Position switching point 1 ... 8 iAx_PosSwitch1 ... 8

This signal monitors the actual position of an axis. A maximum of eight axis monitoring positions (electric limit switches) can be assigned to an axis.

Using the machine parameters SFU\WayPoint\Point[x] (**/SFU/WayPoint/Point[1]/DrIndWayPoint, /SFU/WayPoint/Point[1]/MonTypeWayPoint** "Monitored axis" (2010 00100), **/SFU/WayPoint/Point[1]/PosWayPoint** "Point position" (2010 00110), **/SFU/WayPoint/Point[1]/IfSigWayPoint** "Interface signal of point" (2010 00120), **/SFU/WayPoint/Point[1]/MonTypeWayPoint** "Cyclic monitoring of rotary axes" (2010 00130)), a total of up to 64 axis monitoring positions can be defined for the complete system.

Create "monitoring windows" by **combining** two signals.

For further information, refer to the "Machine parameters" manual. Additional documentation titles are provided in chapter 1 "About this Documentation".

Signal level

1: The actual axis position is greater than or equal to the monitoring position.

0: The actual axis position is lower than the monitoring position.

Chanel number bit 0 ... bit 5 iAx_ChIndex_00 ... 05

This signal is used to output the binary coded channel number of the corresponding channel.

"0" is output if the axis is not assigned to any channel (e.g. an asynchronous axis).

Channel number	Bit 5 (iAx_ ChIndex_05)	Bit 4 (iAx_ ChIndex_04)	Bit 3 (iAx_ ChIndex_03)	Bit 2 (iAx_ ChIndex_02)	Bit 1 (iAx_ ChIndex_01)	Bit 0 (iAx_ ChIndex_00)
No channel	0	0	0	0	0	0
Channel 1	0	0	0	0	0	1
...	...	...	...	...	...	...
Channel 12	0	0	1	1	0	0
...	...	...	...	...	...	...
Channel 60	1	1	1	1	0	0

Tab. 8-8: Channel number, bit 0 ... 5

If axes are lent to another channel, they are output as follows:

State	Channel information
Channel is activated (for program selection or manual data input) with one axis	Number of this channel
Channel is deactivated (for program deselection or end of manual data input) with one axis	Number of this channel
Channel releases axis with "RemAxis" and axis is not released in its original channel	Number of original channel
Channel releases axis with "RemAxis" and axis is released in its original channel	0

Tab. 8-9: Channel output

Definitions:

Lent axis:	An axis is lent if a channel accesses an "axis of an inactive channel" while the axis was not released with "RemAxis" (RAX).
Original channel:	After the control startup, all synchronous axes belong to an original channel.

Axis override 0% iAx_Override0

This signal indicates for an asynchronous axis that its axis override is set to 0% and that no override 100% is selected via the interface (qAx_Override100).

Signal level

1: 0% acts on the axis.

0: A value not equal to 0% acts on the axis.

Axis override 100% iAx_Override100

This signal indicates for an asynchronous axis that its axis override is set to 100% and/or that override 100% is selected via the interface (qAx_Override100).

Signal level

1: A value of 100 % is applied to the axis.

0: Less than 100 % are applied to the axis.

SCS signal status 0 ...15 iAx_ScsState00 ... 15

In the signal status word (Sercos parameter S-0-0144), the configured bit signals from the cyclic telegram of the axis drive are transmitted to the NC.

The status signals can be configured in the parameter S-0-0026. To transmit the signals in the cyclic telegram, apply the parameter S-0-0144 to the configuration list of the drive telegram (S-0-0016).

Example:

Configuration:

- S-0-0026 = (330, 331,0, 333, 0, 335, 336, 337, 0, 0, 0, 0, 0, 0, 0, 0):
  Configuration of the signal status word
- S-0-0016 = (40, 144):
  Configuration of the drive telegram

The following status messages are assigned to the individual signals:

- iAx_ScsState00: S-0-0330 message $n_{act}= n_{com}$
- iAx_ScsState01: S-0-0331 message $n_{act}= 0$
- iAx_ScsState02: -
- iAx_ScsState03: S-0-0333 message $M_d \geq M_{dx}$
- iAx_ScsState04: -
- iAx_ScsState05: S-0-0335 message $n_{com}>n_{limit}$
- iAx_ScsState06: S-0-0336 message In Position
- iAx_ScsState07: S-0-0337 message $P \geq P_x$
- iAx_ScsState08: -
- iAx_ScsState09: -
- iAx_ScsState10: -
- iAx_ScsState11: -
- iAx_ScsState12: -
- iAx_ScsState13: -
- iAx_ScsState14: -
- iAx_ScsState15: -

☞ For a detailed description of all status messages available, refer to the "Sercos interface specification".

Customer output 1 ... 8 iAx_Custom1 ... 8

NC configurable interface that can be assigned as required.

(This option is currently not yet active)

Index master axis bit 0 ... bit 7 iAx_MasterAxindex_00 ... 07

Each axis is provided with the binary-coded axis number of your master axis. The following applies:

- If the axis is not coupled, the value of the master axis number is 0.
- If the axis is the master, the master axis number is identical to the axis number.

Standstill error iAx_TrqExceed

This signal monitors the standstill torque specified for an axis.

☞ For gantry axes, this signal refers to the sum of standstill torque values (irrespective of the sign) of all axes of a group of coupled axes. The signal is set only for the master axis.

Signal level

1: The standstill torque specified for an axis has been exceeded.

0: The standstill torque specified for an axis has not been exceeded.

Lagging error iAx_CoupleLag

This signal reports the exceedance of the coupling distance of a gantry axis.

Signal level

1: The maximum coupling lag specified for an axis has been exceeded.

0: The maximum coupling lag specified for an axis has not been exceeded.

Gantry command value offset active iAx_CmdOffsExst

Signal level

1: There is a command value offset for this slave.

Compensable Gantry command value offset exceeded iAx_CmdOffsExceed

Signal level

1: The compensable command value offset has been exceeded for this slave.

Delayed error reaction iAx_DelayErrClass1

This signal has some relation with iAx_DrvErrClass1 (error of state class 1).

If the function "Delayed NC error reaction for non-fatal drive errors" is enabled, this signal is set together with iAx_DrvErrClass1.

☞ For further information, refer to the chapter "Delayed NC error reaction for non-fatal drive errors" in the "Functional description" manual.

Signal level

1: A non-fatal drive error occurred and the drive provides time for the PLC and the NC to perform a coordinated motion standstill.

0: No delayed error reaction is active or the delayed error reaction is completed.

Error of state class 1 iAx_DrvErrClass1

This signal is a copy of the corresponding signal in the Sercos drive status word of the axis in question (the drive switches off and is locked).

It can only be unlocked by the command "Reset diagnostic class-1" set automatically by the NC at **control reset**.

The pending error can be analyzed manually via the Sercos monitor or automatically by direct PLC reading.

Parameter S-0-0011 consists of two bytes and is structured as follows:

Bit 0:	Reserved	
Bit 1:	Amplifier switch off temperature	S-0-0203
Bit 2:	Motor switch off temperature	S-0-0204
Bit 3:	Reserved	
Bit 4:	Control voltage error	
Bit 5:	Feedback error (encoder, measuring system)	
Bit 6:	Error in electronic commutation system	
Bit 7:	Overcurrent	
Bit 8:	Overvoltage	
Bit 9:	Undervoltage error	
Bit 10:	Phase error in power supply	
Bit 11:	Excessive controller deviation	S-0-0159
Bit 12:	Communication error	
Bit 13:	Position limit value exceeded	S-0-0049
Bit 14:	Reserved	
Bit 15:	Manufacturer-specific error	S-0-0129

Tab. 8-10: Structure of parameter S-0-0011

Change in state class 2 iAx_DrvChangeClass2

This signal is copy of the corresponding signal in the Sercos drive status word of the axis in question (switch off prewarning). The bit is set when the state class 2 is modified and reset after reading the status via parameter S-0-0012. It can be read manually via the Sercos monitor or directly by the PLC.

The parameter S-0-0012 consists of 2 bytes and includes the following warnings:

Bit 0:	Overload warning	
Bit 1:	Amplifier overtemperature warning	
Bit 2:	Motor overtemperature warning	S-0-0201
Bit 3:	Reserved	
Bit 4:	Reserved	
Bit 5:	Reserved	
Bit 6:	Reserved	
Bit 7:	Reserved	
Bit 8:	Reserved	
Bit 9:	Undervoltage in intermediate circuit	
Bit 10:	Reserved	

Bit 11:	Reserved	
Bit 12:	Reserved	
Bit 13:	Reserved	
Bit 14:	Reserved	
Bit 15:	Manufacturer -specific warning	S-0-0095

Tab. 8-11: Warnings of parameter S-0-0012

The **State class 2** mask (S-0-0097) can be used to mask the switching off prewarnings with regard to its effect on the change bit.

Change in state class 3 iAx_DrvChangeClass3

This signal is a copy of the corresponding signal in the Sercos drive status word of the axis in question (operating state messages). The bit is set when state class 3 is modified and reset after reading the status via parameter S-0-0013. It can be read manually via the Sercos monitor or directly by the PLC.

The parameter S-0-0013 consists of two bytes and contains the following messages:

Bit 0:	$n_{act} = n_{com}$	S-0-0330
Bit 1:	$n_{act} = 0$	S-0-0331
Bit 2:	$\lvert n_{act} \rvert < \lvert n_x \rvert$	S-0-0332
Bit 3:	$\lvert Md \rvert \geq \lvert Md_x \rvert$	S-0-0333
Bit 4:	$\lvert Md \rvert \geq \lvert Md_{lim} \rvert$	S-0-0334
Bit 5:	$\lvert n_{com} \rvert > \lvert n_{lim} \rvert$	S-0-0335
Bit 6:	In Position	S-0-0336
Bit 7:	$\lvert P \rvert \geq \lvert P_x \rvert$	S-0-0337
Bit 8:	-	
Bit 9:	$\lvert n_{act} \rvert \geq$ min. spindle speed	S-0-0339
Bit 10:	$\lvert n_{act} \rvert \geq$ max. spindle speed	S-0-0340
Bit 11:	In position rough	S-0-0341
Bit 12:	Target position reached	S-0-0342
Bit 13:	Interpolator stopped	S-0-0343
Bit 14:	-	
Bit 15:	Manufacturer-specific operating states	S-0-0182

Tab. 8-12: Messages of the parameter S-0-0013

The "State class 3" mask (S-0-0098) can be used to mask the operating states with regard to its effect on the change bit.

Reduced torque iAx_TrqLim

This signal indicates that the corresponding axis is operated at a reduced maximum torque.

Signal level

1: Maximum torque of the axis is reduced.

0: Maximum torque of the axis is not reduced.

Test mode (dry run) iAx_DryRun

This signal is set when the axis is in test mode.

In test mode, axes are still interpolated internally, but the calculated command values are only shown on the display and not transmitted to the drive. The axis is held in the same position as it was when test mode was activated.

The test mode can be automatically activated for non-connected drives (no Sercos ring present) via a softkey or via the machine parameter 1001 00010 **/ MAIN/Dr[1]/EnablVirtMode** "Virtual drive".

Enabled for power activation iAx_DrvPower

This signal is closely related to the signals **Drive on**, **Drive ready** and **Drive in operation** (see S-0-0135, bit 14/15).

The functional relationship is shown in detail in the signal diagram for the signal (Drive on qAx_DrvOn, page 81)).

Signal level

1: The initialization of the Sercos interface, the drive and the NC control are completed and no errors are present. This signal is a prerequisite to connect power to the drive.

0: The power cannot be connected at the drive.

Drive ready iAx_DrvReady

This signal is closely related to the signals **Drive on**, **Enabled for power activation** and **Drive in operation** (see S-0-0135, bit 14/15).

The functional relationship is shown in detail in the signal diagram for the signal (Drive on qAx_DrvOn, page 81).

Signal level

1: Power is connected to the corresponding drive. The signal remains set as long as the power is not switched off.

0: No power is connected to the drive.

Drive in operation iAx_DrvAct

This signal is closely related to the signals **Drive on**,**Enabled for power activation** and **Drive ready** (see S-0-0135, bit 14/15).

The functional relationship is shown in detail in the signal diagram for the signal (Drive on qAx_DrvOn, page 81)).

Signal level

1: The control loop is closed and the drive is under torque. The axis can be traversed if there are not other inhibits active such as **Feed inhibit**.

0: The control loop is open and the drive is not under torque.

9 Spindle-related interface signals

9.1 Overview on spindle-related interface signals

9.1.1 General information

Spindle-related interface signals have an effect on NC functions related to the control of a machining spindle.

An interface is available for each configured spindle (for n spindles, the spindle-related interface is also mapped n times on the corresponding flag).

For spindles assigned to one channel, the channel-related interface is also significant.

The following description refers to the interface signals for one spindle.



The supported scope of functions of a spindle depends on the spindle type (machine parameter SP/SpFunc/Sp[1]/Base/SpType "Spindle type" (1040 00001)) and other settings in the machine parameters and drives.

9.1.2 Overview on output signals (PLC -> NC)

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
0.0	qSp_CAxOn	C-axis on	1.0	qSp_TurnCW	Spindle M3 manual
0.1	qSp_CAxOff	C-axis off	1.1	qSp_TurnCCW	Spindle M4 manual
0.2	qSp_JogPlus	Jog spindle M3	1.2	qSp_Stop	Spindle M5 manual
0.3	qSp_JogMinus	Jog spindle M4	1.3	qSp_Orientate	Spindle M19 manual
0.4	-	Reserved	1.4	-	Reserved
0.5	-	Reserved	1.5	-	Reserved
0.6	-	Reserved	1.6	-	Reserved
0.7	qSp_Reset	Spindle reset	1.7	-	Reserved
2.0	qSp_SafOpModeSwitch	Mode selection (BA)	3.0	qSp_Gear1Act	Acknowledgement GTS 1
2.1	qSp_SafDrvLock	Drive inhibit	3.1	qSp_Gear2Act	Acknowledgement GTS 2
2.2	qSp_SafEnablCtrl	Enabling button	3.2	qSp_Gear3Act	Acknowledgement GTS 3
2.3	qSp_SafSwitch1	Safety switch 1 (S1)	3.3	qSp_Gear4Act	Acknowledgement GTS 4
2.4	qSp_SafSwitch2	Safety switch 2 (S2)	3.4	-	Reserved
2.5	qSp_ SafCheckInputState	Check input, safety status	3.5	-	Reserved
2.6	qSp_SafTechState	Status of safety signals	3.6	-	Reserved
2.7	qSp_SafRedTransTime	Shorten transfer time	3.7	qSp_GearIdleAct	Acknowledgement in idle
4.0	qSp_ManSpeed_00	Jog speed, bit 0	5.0	qSp_SpeedLimit	Speed limit

Bit	Symbolic address	PLC output signal	Bit	Symbolic address	PLC output signal
4.1	qSp_ManSpeed_01	Jog speed, bit 1	5.1	qSp_SValueSD	S-value specified via SD
4.2	qSp_ManSpeed_02	Jog speed, bit 2	5.2	qSp_HomeSwitch	Reference switch
4.3	-	Reserved	5.3	-	Reserved
4.4	-	Reserved	5.4	-	Reserved
4.5	-	Reserved	5.5	-	Reserved
4.6	-	Reserved	5.6	-	Reserved
4.7	-	Reserved	5.7	qSp_Override100	Spindle override 100 %
6.0	qSp_Override_00	Override, bit 0	7.0	qSp_Override_08	Override, bit 8
6.1	qSp_Override_01	Override, bit 1	7.1	qSp_Override_09	Override, bit 9
6.2	qSp_Override_02	Override, bit 2	7.2	qSp_Override_10	Override bit 10
6.3	qSp_Override_03	Override, bit 3	7.3	qSp_Override_11	Override bit 11
6.4	qSp_Override_04	Override, bit 4	7.4	qSp_Override_12	Override bit 12
6.5	qSp_Override_05	Override, bit 5	7.5	qSp_Override_13	Override bit 13
6.6	qSp_Override_06	Override, bit 6	7.6	qSp_Override_14	Override bit 14
6.7	qSp_Override_07	Override, bit 7	7.7	qSp_Override_15	Override bit 15
8.0	qSp_Custom1	Customer input 1	9.0	-	Reserved
8.1	qSp_Custom2	Customer input 2	9.1	-	Reserved
8.2	qSp_Custom3	Customer input 3	9.2	-	Reserved
8.3	qSp_Custom4	Customer input 4	9.3	-	Reserved
8.4	qSp_Custom5	Customer input 5	9.4	-	Reserved
8.5	qSp_Custom6	Customer input 6	9.5	-	Reserved
8.6	qSp_Custom7	Customer input 7	9.6	-	Reserved
8.7	qSp_Custom8	Customer input 8	9.7	-	Reserved
10.0	-	Reserved	11.0	-	Reserved
10.1	-	Reserved	11.1	-	Reserved
10.2	-	Reserved	11.2	-	Reserved
10.3	-	Reserved	11.3	-	Reserved
10.4	-	Reserved	11.4	-	Reserved
10.5	-	Reserved	11.5	-	Reserved
10.6	-	Reserved	11.6	qSp_DrvOn	Drive on
10.7	qSp_RedDelayErr-Class1	Delayed Reduce error reaction	11.7	qSp_DrvLock	Spindle inhibit

Tab. 9-1: Overview on output signals (PLC --> NC)

9.1.3 Overview on input signals (NC --> PLC)

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
0.0	iSp_CAxAct	C-axis is active	1.0	iSp_ProgSpReach	Speed reached
0.1	iSp_CAxSwitch	C-axis switching	1.1	iSp_SpLim	Speed limited
0.2	iSp_TurnCmd	Rotation command	1.2	iSp_Stop	Spindle stop
0.3	iSp_TurnDirM4	Direction of rotation M4	1.3	iSp_OrientateFinish	Spindle oriented
0.4	-	Reserved	1.4	iSp_OrientateAct	Orient spindle active
0.5	iSp_InPos	Spindle in position	1.5	-	Reserved
0.6	iSp_PosCtrl	Position control active	1.6	-	Reserved
0.7	iSp_Reset	Spindle reset	1.7	-	Reserved
2.0	iSp_SafOpMode_00	Safety mode, bit 0	3.0	iSp_Gear1Sel	GTS 1 selection
2.1	iSp_SafOpMode_01	Safety mode, bit 1	3.1	iSp_Gear2Sel	GTS 2 selection
2.2	iSp_SafOpMode_02	Safety mode, bit 2	3.2	iSp_Gear3Sel	GTS 3 selection
2.3	iSp_SafOpMode_03	Safety mode, bit 3	3.3	iSp_Gear4Sel	GTS 4 selection
2.4	iSp_SafStatePos	Safe position status	3.4	iSp_GearChange	GTS change
2.5	iSp_ SafCtrlOutputState	Status output of controller	3.5	iSp_IdleSpeed	Idling speed reached
2.6	-	Reserved	3.6	-	Reserved
2.7	-	Reserved	3.7	iSp_GearIdleSel	Idle gear selection
4.0	-	Reserved	5.0	-	Reserved
4.1	-	Reserved	5.1	-	Reserved
4.2	-	Reserved	5.2	-	Reserved
4.3	-	Reserved	5.3	-	Reserved
4.4	-	Reserved	5.4	-	Reserved
4.5	-	Reserved	5.5	-	Reserved
4.6	-	Reserved	5.6	iSp_Override0	Spindle override 0 %
4.7	iSp_ToolLimAct	Tool limitations active	5.7	iSp_Override100	Spindle override 100 %
6.0	iSp_ScsState00	SCS signal status 0	7.0	iSp_ScsState08	SCS signal status 8
6.1	iSp_ScsState01	SCS signal status 1	7.1	iSp_ScsState09	SCS signal status 9
6.2	iSp_ScsState02	SCS signal status 2	7.2	iSp_ScsState10	SCS signal status 10
6.3	iSp_ScsState03	SCS signal status 3	7.3	iSp_ScsState11	SCS signal status 11
6.4	iSp_ScsState04	SCS signal status 4	7.4	iSp_ScsState12	SCS signal status 12
6.5	iSp_ScsState05	SCS signal status 5	7.5	iSp_ScsState13	SCS signal status 13
6.6	iSp_ScsState06	SCS signal status 6	7.6	iSp_ScsState14	SCS signal status 14
6.7	iSp_ScsState07	SCS signal status 7	7.7	iSp_ScsState15	SCS signal status 15
8.0	iSp_Custom1	Customer output 1	9.0	iSp_CoupleIndex_00	No. of coupling, bit 0

Bit	Symbolic Address	PLC input signal	Bit	Symbolic Address	PLC input signal
8.1	iSp_Custom2	Customer output 2	9.1	iSp_CoupleIndex_01	No. of coupling, bit 1
8.2	iSp_Custom3	Customer output 3	9.2	iSp_CoupleIndex_02	No. of coupling, bit 2
8.3	iSp_Custom4	Customer output 4	9.3	-	Reserved
8.4	iSp_Custom5	Customer output 5	9.4	-	Reserved
8.5	iSp_Custom6	Customer output 6	9.5	-	Reserved
8.6	iSp_Custom7	Customer output 7	9.6	-	Reserved
8.7	iSp_Custom8	Customer output 8	9.7	-	Reserved
10.0	iSp_Master	Spindle is master	11.0	iSp_DrvErrClass1	Error of state class 1
10.1	iSp_CoupleErr	Coupling error	11.1	iSp_DrvChangeClass2	Modification of state class 2
10.2	iSp_Synchr1	Synchronous run 1	11.2	iSp_DrvChangeClass3	Modification of state class 3
10.3	iSp_Synchr2	Synchronous run 2	11.3	-	Reserved
10.4	-	Reserved	11.4	iSp_DryRun	Test mode (dry run)
10.5	-	Reserved	11.5	iSp_DrvPower	Enabled for power activation
10.6	-	Reserved	11.6	iSp_DrvReady	Drive ready
10.7	iSp_DelayErrClass1	Delayed error reaction	11.7	iSp_DrvAct	Drive in operation

Tab. 9-2: Overview on input signals (NC -> PLC)

9.2 Signal description

9.2.1 Output signals (PLC--> NC)

General information

Depending on their function, the MTX evaluates the signals either as **static signals** with the signal levels logic 0 and logic 1 or as **edge-triggered signals** with the signal edges 0 --> 1 or 1 --> 0.

☞ The supported scope of functions of a spindle depends on the spindle type (machine parameter SP/SpFunc/Sp[1]/Base/SpType "Spindle type" (1040 00001)) and other settings in the machine parameters and drives.

Only the signal "qSp_Reset" is evaluated for spindles of type "External spindle".

C-axis On qSp_CAxOn

The PLC uses this signal to switch a spindle to C-axis mode.

Application:

In special cases (spindle/tool turret), it can be necessary to switch on the C-axis mode immediately after control startup.

Signal edge

0 --> 1: The running spindle is decelerated and switched to C-axis mode.

The NC-internal switching is affected by the settings in the machine parameter **/SP/SpFunc/Sp[1]/SpTurrAx/MeasSysSpTurrAx** "C-axis: Tool turret " (1040 00060).

C-axis Off qSp_CAxOff

The PLC uses this signal to switch a spindle back to spindle mode.

Signal edge

0 --> 1: The drive is switched back to spindle mode.

The NC-internal switching is affected by the settings in the machine parameter **/SP/SpFunc/Sp[1]/SpTurrAx/MeasSysSpTurrAx** "C-axis: Tool turret " (1040 00060).

1 --> 0: No function.

Jog spindle M3 qSp_JogPlusJog spindle M4 qSp_JogMinus

These signals are effective in the NC mode **Setup** and operate together with the coded signals "Jog speed bit 0 ... bit 2".

Signal level

1: The speed has to be specified using the coded "Jog speed, bit 0 ... bit 2" signals. Furthermore, the control loop has to be closed (drive set in operation), spindle inhibit has to be reset and a gear range has to be activated. As long as the **Jog spindle M3** or **Jog spindle M4** signal is set, the spindle rotates at the corresponding speed in the selected direction of rotation.

0: The spindle is decelerated up to standstill. The control loop remains closed.

Example:

In setup mode, the spindle is to be traversed in clockwise direction at a predefined speed.

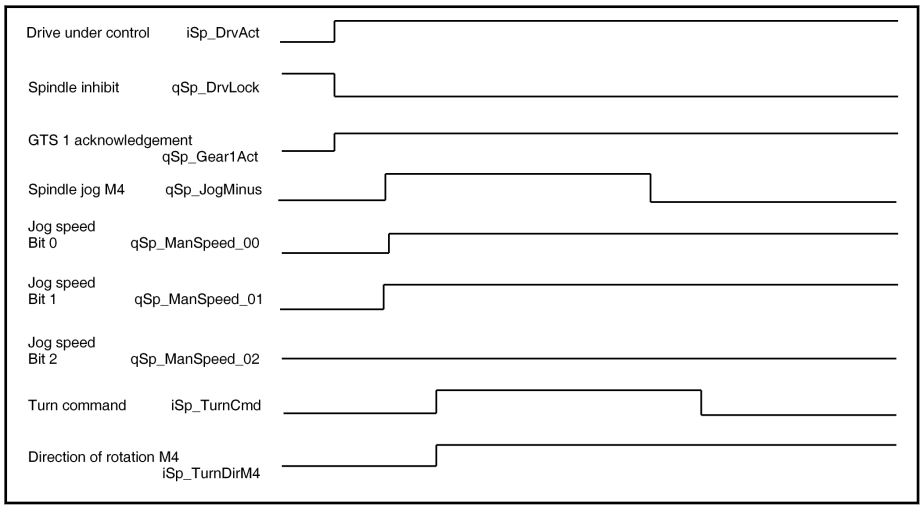


Fig. 9-1: Jogging spindle

The control loop is closed (**Drive in operation** is set), **Spindle inhibit** is reset and gear range 1 (**Gear range 1 acknowledgement**) is activated.

At the same time that the **Jog spindle M3**signal is set, a spindle speed must be input (**Jog speed, bit 0 - Jog speed, bit 2**). The jog speed is defined for each spindle in the machine parameter **/SP/SpFunc/Sp[1]/Jog/Speed/Jog Speed[1]** "Jog spindle speed" (1040 00015).

The **command for rotation** is set by the NC control and the **direction of rotation M4** is output together with the spindle command.

Since spindle inhibit has already been reset, the spindle accelerates to the preselected speed.

If the **Spindle inhibit**signal is still set when the command for rotation is output, the direction of rotation M4 is only output after spindle inhibit is reset.

As soon as the **Jog spindle** signal is reset, the spindle deceleration is initiated. If the spindle is at standstill, the **command of rotation** is reset.

☞ In normal mode, the spindle is generally only traversed in a controlled and not in the position control loop. Thus, it is possible that the command of rotation is reset while the spindle is still decelerating. To ensure that the spindle is at standstill, the "Spindle stop" signal has to be queried additionally.

Spindle reset qSp_Reset

This signal can be used to cancel the current motion specification or an active command (e.g. orient spindle) of the corresponding spindle. The spindle acknowledges the cancelation task the **Spindle reset** signal (iSp_Reset).

Signal edge

0 --> 1: The signal **spindle reset** (iSp_Reset) is set.

1 --> 0: No effect.

Spindle M3 manual qSp_TurnCWSpindle M4 manual qSp_TurnCCW

Alternatively to the NC part program (programming M3, M4), these signals are used to determine the direction of spindle rotation:

- Spindle M3 manual:

  Spindle clockwise
- Spindle M4 manual:

  Spindle counterclockwise.

In this context, the signal **Spindle M5 manual** is significant (refer to the example Spindle M5 manual qSp_Stop, page 98).

Signal edge

0 --> 1: Using this edge, a spindle speed and the corresponding direction of spindle rotation is specified for the spindle drive. The spindle starts to rotate if the control loop is closed (**Drive in operation**) and no **Spindle inhibit** is set.

The spindle speed corresponds to the speed programmed last.

1 --> 0: No effect.

Spindle M5 manual qSp_Stop

Alternatively to the NC part program (programming M5), this signal is used to activate the spindle stop function.

In this context, the signals "Spindle M3 manual" and "Spindle M4 manual" are also significant.

Signal edge

0 --> 1: The spindle stops at the parameterized deceleration. The previously active gear range and spindle speed remain stored control-internally. The control loop remains closed and the spindle drive is under torque.

1 --> 0: No effect.

Example:

The spindle is to be accelerated manually in counterclockwise direction up to the latest programmed speed.

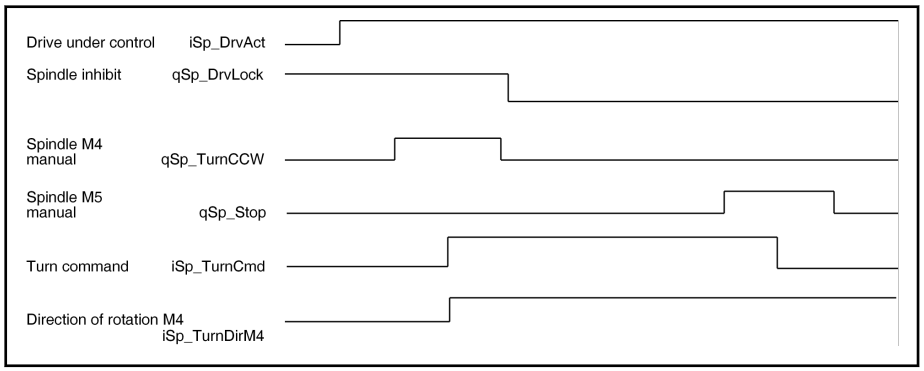


Fig. 9-2: Spindle M5 manual

The **Drive in operation** signal indicates that the control loop is closed and that the spindle is under torque.

When setting the **Spindle M4 manual** signal, the latest programmed speed is transmitted to the spindle drive.

Due to the speed specification, the NC sets the **command of rotation** and the **direction of rotation M4** signals.

When the **command of rotation** is output, the **Spindle M4 manual** signal can be reset. The spindle does not rotate while **Spindle inhibit**is set.

Only when the **Spindle inhibit** is reset, the spindle starts rotating and **Spindle rotates** be set. By setting the **Spindle M5 manual** signal, the rotation of the spindle is stopped and the **command of rotation** and **direction of rotation M4** are reset by the NC.

Spindle M19 manual qSp_Orientate

Alternatively to the NC part program (M19), this signal calls the "Orient spindle" function. Thus, the control loop has to be closed and the drive in operation.

In this context, the signals **Spindle in position**, **Orient spindle active** and **Spindle oriented** are significant.

☞ This function is only available for Sercos spindles (machine parameter **/SP/SpFunc/Sp[1]/Base/SpType** "Spindle type selection" (1040 00001)).

Signal edge

0 --> 1: Orienting spindle is initiated. The position results from the current content of the system state SysSpCmdData[n]/Pecorinos. The direction of rotation depends on Sercos parameter S-0-0154.

1 --> 0: No effect.

Example:

The spindle is to be oriented manually. The control loop is closed.

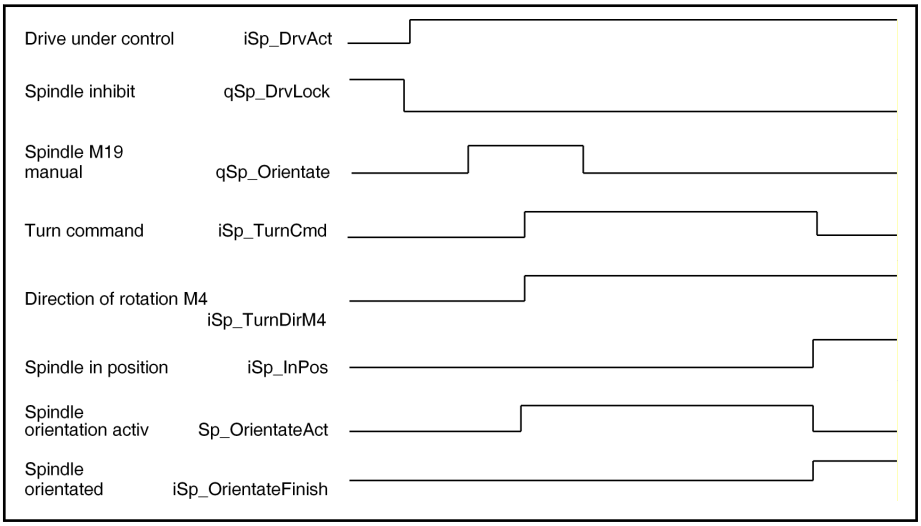


Fig. 9-3: Spindle M19 manual

The control loop is closed (**Drive in operation** is set) and **Spindle inhibit** is reset.

When the signal **Orient spindle manually** is set, the spindle receives the command to rotate. The **command for rotation** is set.

Depending on the direction of rotation, **Direction of rotation M4** is set when the spindle is rotating counterclockwise (M4).

The signal **Spindle orientate manually** can be reset when the signal **rotation command** is output.

If the spindle is rotating, **Orient spindle active** is set.

Once the spindle has been oriented to the flag position, **Spindle oriented** and **Spindle in position** are set and the **command of rotation**.

Operation mode selection qSp_SafOpModeSwitch

This signal is part of the "Integrated Safety engineering according to IEC 61508" function of the IndraDrive series.

The operation mode selection of the PLC is transferred as channel 1 to the drive via the Sercos interface in real time.

Signal level

1: Normal mode NO

0: Special mode SO

(Special mode at standstill/in motion)

Drive inhibit qSp_SafDrvLock

This signal is part of the "Integrated Safety engineering according to IEC 61508" function of the IndraDrive series.

The "Drive inhibit" signal is transferred as channel 1 to the drive using the PLC via the Sercos interface in real time. The drive releases itself from torque when drive inhibit is selected.

Signal level

1: Drive inhibit on.

0: Drive inhibit off.

Enabling button qSp_SafEnablCtrl

This signal is part of the "Integrated Safety engineering according to IEC 61508" function of the IndraDrive series.

Switching to "Special mode with motion" via PLC is transferred as channel 1 to the drive via the Sercos interface in real time.

Signal level

1: Special mode with motion

0: Special mode at standstill

Safety switch 1 (S1) qSp_SafSwitch1

Safety switch 2 (S2) qSp_SafSwitch2

These signals are part of the "Integrated Safety engineering according to IEC 61508" function of the IndraDrive series.

Up to four different parameter sets can be stored in the drive for the "special mode with motion". The parameter set selected via PLC is transferred as channel 1 to the drive via Sercos in real time.

Up to MTX14V06, the MTX supports a maximum of 2 save motions. From MTX14V08, a maximum of 4 safe motions can be used in connection with Safety engineering according to IEC 61508.

Please refer to the notes in the drive documentation about "State machine of integrated Safety technology" and the Sercos parameter "P-0-3212 (SI control word, channel 1) bit 3 and bit 4".



If the SI switch 2 information "P-0-3212 (SI control word, channel 1) bit 4" is cyclically transmitted to the drive, MTX 4 expects configured safe motions!

When using a maximum of 2 safe motions, only the SI switch 1 information "P-0-3212 (SI control word, channel 1) bit 3" is cyclically transmitted to the drive.

	Bit 1 (qSp_SafSwitch2)	**Bit 0 (qSp_SafSwitch1)**
SMM1	0	0
SMM2	0	1
SMM3	1	0
SMM4	1	1

Tab. 9-3: Selecting safe motions

Check input of safety status qSp_SafCheckInputState

This signal is part of the "Integrated Safety engineering according to IEC 61508" function of the IndraDrive series.

The safe status signal set via PLC is transferred as channel 1 to the drive via the Sercos interface in real time.

Status of safety signals qSp_SafTechState

This signal is part of the "Integrated Safety engineering according to IEC 61508" function of the IndraDrive series.

To perform checks, the control signals of the PLC (qSp_Saf...) transferred via channel 1 to the drive have to be adjusted dynamically. As a result, the PLC sets the signals to "0" for one cycle. At the same time, the PLC sets the safe-

ty signal status from "valid" to "made dynamic" and thus informing the drive that the control signals are invalid.

The signals are then reset.

Signal level
1: The control signals of the safety technology are made dynamic.

0: The control signals of the safety technology are valid.

Reducing transmission Time qSp_SafRedTransTime

This signal is part of the safety technology of the IndraDrive and ctrlX DRIVE drive series.

This signal allows the PLC to reduce the time of transfer from normal mode to special mode or from special mode to special mode.

When using the function "Integrated Safety engineering according to IEC 61508":

- Observe the notes in the drive documentation about "state machine of the integrated safety technology" and the Sercos parameter "P-0-3212 (SI control word, channel 1), bit 11".

When using the "Safe Motion" function:

- Please refer to the notes in the drive documentation about "Transition functions" and the Sercos parameter "P-0-3261.0.1 (SMO: Control word state machine, functional) bit 0".

Signal level
1: Signals to the drive that the adjustment of the command value system is now complete and that the drive enables monitorings of the selected special mode.

0: No effect.

Acknowledgement GTS 1...4 qSp_Gear1Act ... 4ActAcknowledgement idling qSp_GearIdleAct

The current gear range is reported using these signals. In case of gear range inconsistency, "spinning" is activated at next spindle task.

At control startup, the acknowledged gear range is applied to the NC.

For analog spindles (**/SP/SpFunc/Sp[1]/Base/SpType** "Spindle type selection" (1040 00001)), gear shifting is not supported. For analog spindles, GTS 1 has to be acknowledged.

Jog speed bit 0 ... bit 2 qSp_ManSpeed_00 ... 02

These signals are effective exclusively in conjunction with the **Jog spindle M3** or **Jog spindle M4** signals.

Depending on the coding, the speed defined in the machine parameter **/SP/SpFunc/Sp[1]/Jog/Speed/JogSpeed[1]** "Jog spindle speed" (1040 00015) is activated. If the preselected spindle speed is not in the range of the current gear range, it is limited to the minimum or maximum spindle speed of the activated gear range.

Machine Parameters	Jog speed		
	Bit 2 (qSp_ ManSpeed_02)	Bit 1 (qSp_ ManSpeed_01)	Bit 0 (qSp_ ManSpeed_00)
.../JogSpeed [1]	0	0	0
.../JogSpeed [2]	0	0	1

Machine Parameters	Jog speed		
	Bit 2 (qSp_ ManSpeed_02)	Bit 1 (qSp_ ManSpeed_01)	Bit 0 (qSp_ ManSpeed_00)
.../JogSpeed [3]	0	1	0
.../JogSpeed [4]	0	1	1
.../JogSpeed [5]	1	0	0
.../JogSpeed [6]	1	0	1
.../JogSpeed [7]	1	1	0
.../JogSpeed [8]	1	1	1

Tab. 9-4: Jog speed

Speed limitation SD qSp_SpeedLimit

The PLC uses this signal to activate the speed limitation entered into the system data "/SysSpCmdData[n]/MinSpeedLimit" and "/SysSpCmdData[n]/MaxSpeedLimit".

Signal edge

0 --> 1: Applies the current content of system data and activates the speed limitation.

1 --> 0: Speed limitation is switched off.

S-value specification via SD qSp_SValueSD

This signal allows to specify the spindle s-value via the system data "/SysSpCmdData[n]/Speed" and "/SysSpCmdData[n]/OriPos".

If the interface signal "qSp_SValueSD" is set, the following restrictions apply to the spindle functionality:

- G96 does not work!

  The combination G96 and "Speed specification via SD" causes a runtime error!
- M40 does not work!

  Only speed values programmed in the channel cause an automatic gear shift.
- SD(205,2,x) (programmed spindle speed) always provides the latest programmed value in the part program.

Signal level

1: The content of the system date "SysSpCmdData[n]/Speed" is used as speed signal.

The content of the system date "SysSpCmdData[n]/OriPos" is used as orientation position.

0: The S-values given in the part program are effective.

Reference switch qSp_HomeSwitch

Transmits the signal of the reference switch to the drive if it is not directly wired at the drive. However, this has to be configured in the drive.

☞
- The signal control word S-0-0145 has to be contained in the master data telegram (S-0-1050.0.6)
- Bit 0 of S-0-0400 has to be configured in the signal control word (S-0-0145)

Signal level

1: The reference switch is active.

0: The reference switch is inactive.

Spindle override 100 % qSp_Override100

This signal hides the spindle override. The spindle rotates at programmed speed.

Signal level

1: The spindle override is hidden.

0: The spindle override is applied.

Override bit 0 ... bit 15 qSp_Override_00 ... 15

The current spindle speed can be influenced via the **Override** function. The "Override" function is applied to the speeds programmed in the NC part program and the speed in "Setup" mode.

This interface can be used to continuously specify 16 bits. The MTX interprets the value directly as override value in 0.01%.

Customer input 1 ... 8 qSp_Custom1 ... 8

NC configurable interface that can be assigned as required.

(This option is currently not yet active)

Reducing delayed error reaction qSp_RedDelayErrClass1

This signal is related to iSp_DelayErrClass1 (delayed error reaction).

The PLC can use this signal to reduce the time up to the spindle standstill by the NC.

☞ For further information, refer to the chapter "Delayed NC error reaction for non-fatal drive errors" in the "Functional Description" manual.

Signal edge

0 --> 1: The delayed error reaction is completed. The NC stops the spindle.

Drive On qSp_DrvOn

This signal switches on the torque in the drive when the drive is ready for operation (iSp_DrvReady is set). The control loop is closed.

Resetting **Drive on** initiates a shutdown. If the spindle is still moving at this point, it is set to standstill at the preset deceleration ramp and switched to state spindle stop (M5).

When the axis is at standstill, the control loop is opened and the **Drive in operation** signal is reset. Torque is no longer applied to the drive.

☞ The entire sequence does not apply to analog spindles.

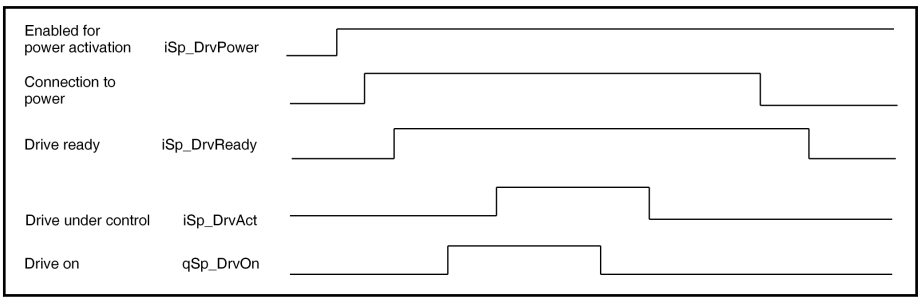


Fig. 9-4: Drive on

Example:

Mode of operation of switching on and off

When the machine is started or voltage is applied, the NC control, the servo drive and the spindle drive start with the initialization phase.

During the initialization phase of the individual components, the digital interface to communicate between the NC and the drives is established.

This digital interface corresponds to the settings of the **Sercos interface**.

The initialization of the **Sercos interface** consists of four system-internal automatic phases. After the interface initialization and once the cyclic operation is active, the **Enabled for power activation** signal is output.

Power can be switched on for the corresponding drives. When the power was connected, the **Drive ready** signal is output.

The position control loop is closed by setting **Drive on**. The drive is under torque and the **Drive in operation** signal is output.

Switching off or disconnection from the power is initiated by the PLC sequential program resetting the **Drive on** signal.

If the axis is still moving, it is set to standstill at the preset deceleration ramp.

At standstill, the position control loop is opened and the **Drive in operation** signal is reset. Torque is no longer applied to the drive.

The power can be switched off in the next step.

The **Enabled for power activation** signal remains set as long as the system components (NC and drives) are initialized.

Spindle inhibit qSp_DrvLock

This signal is effective in every mode when the control loop is closed and torque is applied to the spindle.

It prevents the execution of a programmed or manually entered spindle motion. If "Spindle inhibit" is set, the idling speed is not carried out.

The signals "command of rotation", "direction of rotation M4" and "Spindle in position" are not affected.

Signal level

1: The spindle cannot be rotated. If a rotating motion is already active, the spindle is stopped at parameterized deceleration.

0: The spindle rotation is not inhibited. If a command to rotate the spindle is initiated (via the NC part program or manually) and the appropriate gear range is active, the spindle can start rotating immediately. If a rotating spindle was set to standstill with "Spindle inhibit", this spindle now starts accelerating to its initial speed.

Example:

Spindle inhibit

The machine is switched on and the control loop has already been closed. A rotary motion (e.g. M3 S1000) has been programmed via the NC part program.

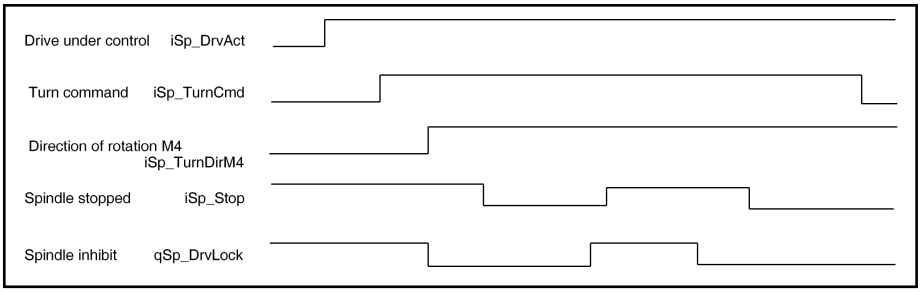


Fig. 9-5: Spindle inhibit

The **Drive in operation** signal informs the PLC sequential program that the control loop is closed. When executing the NC block, in which the rotary motion was programmed, the **command of rotation** signal is output.

As long as **Spindle inhibit** is set, the rotary motion is not performed. **Direction of rotation M4** is not set and **Spindle stop** is set. The direction of rotation can only be queried a command of rotation is present.

When **Spindle inhibit** is reset, the programmed rotary motion is executed, the **Direction of rotation M4** is set according to the motion and **Spindle stop** is reset.

If **Spindle inhibit** is set again during the rotary motion, the rotary motion is stopped and **Spindle stop** is set when the spindle is at standstill.

When **Spindle inhibit**is reset, the spindle accelerates to the previous speed and **Spindle stop** is reset.

The **command of rotation** signal is reset when the spindle is stopped via the **M5** spindle stop function (NC part program or interface signal).

9.2.2 Input signal (NC--> PLC)

General information

☞ The supported scope of functions of a spindle depends on the spindle type (machine parameter SP/SpFunc/Sp[1]/Base/SpType "Spindle type" (1040 00001)) and other settings in the machine parameters and drives. The signals are not set for spindles of type "external spindle".

C-axis active iSp_CAxAct

This signal indicates whether the corresponding spindle is in C-axis mode (positioning interface) or in spindle mode (speed interface). Spindles, which can be switched to C-axis mode, have **a normal axis interface** for C-axis mode in addition to the spindle interface. The C-axis signal is active indicates the relevant interface.

Signal level

1: The spindle is in C-axis mode: The axis interface is valid, the spindle interface is not operated by the NC.

0: The spindle is in spindle operation: The spindle interface is operated by the NC. The axis interface is not valid.

C-axis switching iSp_CAxSwitch

This signal indicates that the spindle is currently in the switching phase between spindle mode and C-axis mode.

The **command of rotation** signal is always output at the same time.

Signal level

1: Spindle is in the switching phase.

0: Spindle is not in the switching phase.

Command of rotation iSp_TurnCmd

The **Command of rotation, Direction of rotation M4, Spindle stop Spindle in position** signals affect each other.

The signal diagrams and explanation on the signals "spindle M3 manual", "spindle M4 manual", "spindle M5 manual (see Spindle M3 manual qSp_TurnCW, page 98) and spindle inhibit (see Spindle inhibit qSp_DrvLock, page 105), the relations are exemplarily shown.

The **Direction of rotation M4** signal is output with **Command of rotation**.

Signal level

1: The spindle is to perform a rotary movement that is specified via a **manual** specification (e.g. spindle M3 manual) or via the **NC parts program** (e.g. M3 S1500). The prerequisite is that the control loop is closed, i.e. the **Drive in operation** signal is set.

0: The rotary motion was completed via a programmed "Spindle stop M5" or the "Spindle M5 manual" signal (qSp_Stop).

In addition, the command of rotation is not set if **Spindle is oriented** or **positioned**or is the **gear range is shifted**.

Direction of rotation M4 iSp_TurnDirM4

This signal is only effective together with the **command of rotation** signal (i.e. **direction of rotation M4** and **command of rotation** have to be in the PLC sequential program and linked by an AND operation.

If no command of rotation is set, the "Direction of rotation M4 signal" is irrelevant.

No "Direction of rotation M4" signal is output or its status remains unchanged for the functions "Gear shifting", "Idling speed" and "Orient spindle".

Signal level

1: If "Direction of rotation M4" and "Command of rotation" are set, the spindle has to perform a rotary motion in **counter-clockwise direction (M4)**.

0: If "Direction of rotation M4" is not set and "Command of rotation" is set, the spindle has to perform a rotary motion in **clockwise direction (M3)**.

Spindle in Position iSp_InPos

This signal is only relevant only together with the functions **Orient spindle** or **Position spindle**.

Prerequisite: The control loop is closed and torque is applied to the spindle (signal **Drive in operation** is set).

Signal level

1: The spindle is oriented or positioned and is located in the in-pos window defined in the drive.

0: The spindle was not oriented and/or is located in an unparameterized inpos window.

Position control active iSp_PosCtrl

Signal level

1: The spindle is controlled via position interface.

0: Spindle is activated via speed interface.

Spindle reset iSp_Reset

This signal indicates that the spindle accepted the control reset task and that the spindle is again ready to accept new motion specifications.

Speed reached iSp_ProgSpReach

Signal level

1: The speed at the spindle corresponds to the programmed spindle speed and/or is in the speed tolerance range specified for each spindle by the superimposition of the machine parameters **/SP/SpFunc/Sp[1]/SpeedWin/AbsWin** "'Speed reached' window in min^-1" (1040 00020) and **/SP/SpFunc/Sp[1]/SpeedWin/PercWin** ""Speed reached" window in % (1040 00021)".

If the speed is limited by programmed speed limits or gear range speed limits, this signal is output when the set limit is reached. In addition, the position of the override switch is taken into account.

0: The spindle speed is not in the speed tolerance band.

Speed limited iSp_SpLim

This signal indicates that the programmed speed is limited by the active gear range or by the programmed spindle speed limit (SMin, SMax).

Signal level

1: Programmed spindle speed is limited.

0: Programmed spindle speed is not limited.

Spindle stop iSp_Stop

This signal relates to the spindle speed.

For Sercos spindles, this signal is only generated if bit 0 of the Sercos ident number **S-0-0331, Status n_act = 0** is transferred cyclically from the drive to the NC via the S-0-144.

Signal level

1: M5 was executed (via NC part program or interface) or the speed 0 (S0) was set via the NC part program or the **Spindle inhibit** or **Override 0** were set.

0: The spindle is rotating.

☞ For Sercos spindles, note the Sercos ident numbers **S-0-0124, Standstill window**.

For external and analog spindles with encoder, note the machine parameter **SP/SpFunc/Sp[]/SpeedWin/StandstillWin** "Standstill window" (1040 00022).

Spindle oriented iSp_OrientateFinish

Signals **Orient spindle active** and **Spindle in position** are significant together with this signal. The functional relation is shown in the signal diagram in Spindle M19 manual qSp_Orientate, page 99.

Signal level

1: The "Spindle orient" function is completed and the spindle is in the orientation position.

0: The orientation function is completed and the spindle is no longer on the position approached as "Orientation position" before.

Orienting spindle active iSp_OrientateAct

This signal is only effective together with the **Orient spindle** function. The functional relationship is shown in detail in the signal diagram (see chapter 9.2.1 "Output signals (PLC--> NC)" on page 96).

Signal level

1: The "Orient spindle" function has been activated via the NC part program (e.g. by programming M19) or the qSp_Orientate signal.

0: Orient spindle is not activated.

Si operation mode bit 0 ... bit 3 iSp_SafOpMode_00 ... 03

These signals are part of the safety technology of the IndraDrive and ctrlX DRIVE drive series.

The control reads the selected mode of the drive (P-0-3215 or P-0-3262) and transfers it to the PLC using these outputs.

☞ The safe motions SMM14 to SMM16 are not displayed in the bit interface.

Safety function	Safety mode			
	Bit 3 (iSp_SafOpMode_03)	**Bit 2 (iSp_SafOpMode_02)**	**Bit 1 (iSp_SafOpMode_01)**	**Bit 0 (iSp_SafOpMode_00)**
Normal mode NO	0	0	0	0
Safe Stop 1 (EMERGENCY STOP) (SS1ES)	0	0	0	1
Safe Stop 1 (SS1) / Safe Stop 2 (SS2)	0	0	1	0
Safe motion (SMM1)	0	0	1	1
Safe motion (SMM2)	0	1	0	0
Safe motion (SMM3)	0	1	0	1
Safe motion (SMM4)	0	1	1	0
Safe motion (SMM5)	0	1	1	1
...				
Safe motion (SMM13)	1	1	1	1

Tab. 9-5: Safety mode

Status of safe position iSp_SafStatePos

This signal is part of the safety technology of the IndraDrive and ctrlX DRIVE drive series.

A prerequisite for the "Safe absolute position" safety function is that the drive uses this signal to report whether referencing is safe.

The signal is generated from P-0-3256, SMO encoder function: Status evaluation formed.

Signal level

1: The drive is safely referenced.

0: No safe referencing.

Status output of controller iSp_SafCtrlOutputState

This signal is part of the "Integrated Safety engineering according to IEC 61508" function of the IndraDrive series.

The drive uses this signal to provide feedback about its safe state.

Signal level

1: Safe drive state.

0: No safe drive state.

Selection GTS 1 ... 4 iSp_Gear1Sel ... 4Sel

Depending on the gear range selection (manual or via the NC part program), the preselected gear range is output.

The gear range can be selected directly via the NC part program by programming **M41 - M44** (gear range 1 - gear range 4). The gear range is derived from the programmed **speed S ....** together with **automatic gear range detection M40**.

If the speed ranges in the individual gear ranges overlap, the gear range with the higher spindle drive speed is output.

In the context of gear range switching, the **idling speed reached** signal is important.

Behavior after switch-on: After switching on or starting up the control, the gear stage is output, which is reported as an **acknowledgement** by the PLC program via the corresponding inputs. If no gear stage is acknowledged, the NC selects the highest stage configured in the machine parameters.

If **automatic gear range detection M40** is active after switching on, the gear range is only output after a **speed S...** has been programmed together with a **direction of rotation M3** or **M4**.

Signal edge

0 →1: For SP/SpFunc/Sp[1..32]/Gear/ChangeMethod = gear applies:

The change of gear is to be performed using this edge. To minimize the mechanical impact on the gear change, the **Spindle spinning** function (if available in the spindle drive) is activated in the spindle drive via the Sercos interface). As soon as the idling speed is reached, the NC sets the **Idling speed reached** signal. Now, mechanical switching can be performed. After the switching operation is complete, the current gear range must be fed back to the respective interface **Acknowledgement GTS 1 ... Acknowledgement GTS 4**. Gear switching is completed by enabling the corresponding parameter block via the Sercos interface.

For SP/SpFunc/Sp[1..32]/Gear/ChangeMethod = winding applies:

Use this edge to change the winding. After the switching operation is complete, the currently enabled winding has to be fed back to the respective interface **Acknowledgement GTS 1 ... Acknowledgement GTS 4**.

GTS switching iSp_GearChange

This signal is only effective together with the gear switching function triggered by **M-functions M40, 41, 42**.

Signal level

1: A change of gear range is executed.

0: The change of gear range is completed.

Idling speed reached iSp_IdleSpeed

This signal is effective together with the gear switching function and the feedback signal of the current gear range.

The spindle override setting is not taken into account when the signal is output.

Signal level

1: The spindle speed corresponds to the idling speed. This signal is normally used to switch gear ranges.

0: Idling speed not yet reached.

Selection gear-idling iSp_GearIdleSel

Depending on the gear selection (manual or via the NC part program), the preselected gear range or idling position of the gear is output.

Via the NC part program, the idling position of the gear can be selected directly by programming the **M48** function. Neither a **M3** or **M4** direction of rotation, **Orient spindle M19** nor a spindle speed **S...** may be programmed together with **M48**.

In this context, the **Idling speed reached** signal is also important.

Signal edge

0 --> 1: This signal edge is used to switch the gear to idle position . To minimize the mechanical impact of the gear change, the **Spindle spinning** function - if available in the spindle drive - is activated in the spindle drive via the Sercos interface.

As soon as the spinning speed (possibly accompanied by a pendulum motion) is reached, the NC sets the **Idling speed reached** signal and the mechanical switching can be carried out.

After switching, the idling position of the gear has to be fed back to the **Idle gear acknowledgement** interface. Gear switching is now completed.

1 --> 0: The "gear switching in idling position" function is deselected.

Tool limitations active iSp_ToolLimAct

This signal indicates that tool-specific limitations are active.

For more information, refer to the "MTX Functional Description Basics" in the chapter "Spindle functions" under "Tool-specific spindle limitations".

Signal level

1: Tool limitations active.

0: Tool limitations are not active.

Spindle override 0 % iSp_Override0

This signal indicates that the override of the spindle is set to 0 % and that no "Override 100 %" has been selected using the interface (qSp_Override100).

Signal level

1: 0 % is applied to the spindle.

0: A value not equal to 0 % is applied to the spindle.

Spindle override 100 % iSp_Override100

This signal indicates that the override of the spindle is set to 100 % and/or that "Override 100 %" has been selected using the interface (qSp_Override100).

Signal level

1: A value of 100 % is applied to the spindle.

0: Less than 100 % are applied to the spindle.

SCS signal status 0 ...15 iSp_ScsState00 ... 15

In the signal status word (Sercos parameter S-0-0144), the configured bit signals from the cyclic telegram of the spindle drive are transmitted to the NC.

The status signals can be configured in the parameter S-0-0026. To transmit the signals in the cyclic telegram, apply the parameter S-0-0144 to the configuration list of the drive telegram (S-0-0016).

Example:

Configuration:

- S-0-0026 = (330, 331,0, 333, 0, 335, 336, 337, 0, 0, 0, 0, 0, 0, 0, 0):
  Configuration of the signal status word
- S-0-0016 = (40, 144):
  Configuration of the drive telegram

The following status messages are assigned to the individual signals:

iSp_ScsState00: S-0-0330 message $n_{act}= n_{com}$

iSp_ScsState01: S-0-0331 message $n_{act}= 0$

iSp_ScsState02: -

iSp_ScsState03: S-0-0333 message $M_d \geq M_{dx}$, as "Symbol_D"

iSp_ScsState04: -

iSp_ScsState05: S-0-0335 message $n_{com}>n_{lim}$

iSp_ScsState06: S-0-0336 message In Position

iSp_ScsState07: S-0-0337 message $P \geq P_x$, as "Symbol_D"

NC-A 7.0: -

NC-A 7.1: -

NC-A 7.2: -

NC-A 7.3: -

NC-A 7.4: -

NC-A 7.5: -

NC-A 7.6: -

NC-A 7.7: -

☞ For a detailed description of all status messages available, refer to the Sercos interface specification.

Customer output 1 ... 8 iSP_Custom1 ... 8

NC configurable interface that can be assigned as required.

(This option is currently not yet active)

No. of coupling bit 0 ... bit 2 iSp_CoupleIndex_00 ... 02

When spindle coupling is active, this signal is used to output the coupling number for each member of a group of coupled spindles.

Spindle is master iSp_Master

This signal indicates the master of a group of coupled group.

1: Spindle is master.

0: Spindle is slave.

Coupling error iSp_CoupleErr

If spindle coupling is active (see iSp_CoupleIndex_00 - 02), this signal provides information on the current state of the group of coupled spindles for the master spindles (iSp_Master is set).

Signal level

1: An error was reported in the coupling link. Clear this error by a "spindle reset" of the master spindle or an "overall reset".

0: The couple link is in normal state.

Synchronous run 1 iSp_Synchr1

If spindle coupling is active (see iSp_CoupleIndex_00 - 02), this signal provides the following information on the **angular offset** for slave spindles (iSp_Master is not set).

Signal level

1: The synchronous run window not violated and a programmed angular offset has been executed.

0: The synchronous run window is violated or a programmed angular offset is just being executed.

Synchronous run 2 iSp_Synchr2

If spindle coupling is active (see iSp_CoupleIndex_00 - 02), this signal provides the following information on the **error window** for slave spindles (iSp_Master is not set).

Signal level

1: The synchronous run error window is not violated.

0: The synchronous run error window is violated.

Delayed error reaction Sp_DelayErrClass1

This signal is related to iSp_DrvErrClass1 (error of state class 1).

If the function "Delayed NC error reaction for non-fatal drive errors" is enabled, this signal is set together with iSp_DrvErrClass1.

For further information, refer to the chapter "Delayed NC error reaction for non-fatal drive errors" in the "Functional Description" manual.

Signal level

1: A non-fatal drive error occurred and the drive provides time for the PLC and the NC to perform a coordinated motion standstill.

0: No delayed error reaction is active or the delayed error reaction is completed.

Error of state class 1 iSp_DrvErrClass1

This signal reports an error of state class 1 (drive switch-off with lock) and is a copy of the corresponding signal in the Sercos drive status word of the drive in question. Locking can only be canceled by the command "Reset state class1" which is set automatically by the NC at control reset.

The pending error can be evaluated manually via the NC user interface or automatically by direct PLC reading S-0-0011.

Parameter S-0-0011 consists of two bytes and is structured as follows:

Bit 0:	-	
Bit 1:	Amplifier switch off temperature	S-0-0203
Bit 2:	Motor switch off temperature	S-0-0204

Bit 3:	-	
Bit 4:	Control voltage error	
Bit 5:	Feedback error (encoder, measuring system)	
Bit 6:	Error in electronic commutation system	
Bit 7:	Overcurrent	
Bit 8:	Overvoltage	
Bit 9:	Undervoltage error	
Bit 10:	Phase error in power supply	
Bit 11:	Excessive controller deviation	S-0-0159
Bit 12:	Communication error	
Bit 13:	Position limit value exceeded	S-0-0049
Bit 14:	-	
Bit 15:	Manufacturer-specific error	S-0-0129

Tab. 9-6: Structure of parameter S-0-0011

☞ This signal is only generated for Sercos spindles (**/SP/SpFunc/Sp[1]/Base/SpType** "Spindle type" (1040 00001)).

Signal level

1: A state class 1 error is present.

0: There is no state class 1 error present.

☞ For more information on the errors of state class 1, refer to the interface specification.

Change in state class 2 iSp_DrvChangeClass2

This signal reports a change of state class 2 (switch off prewarning) and is a copy of the corresponding signal in the Sercos drive status word of the drive in question.

The bit is set when state class 2 is modified and reset after reading the status via the parameter S-0-0012. It can be read manually via the NC user interface or automatically by direct PLC reading of S-0-0012.

The parameter S-0-0012 consists of two bytes and contains the following warnings:

Bit 0:	-	
Bit 1:	Amplifier overtemperature warning	S-0-0311
Bit 2:	Motor overtemperature warning	S-0-0312
Bit 3 to 14:	-	
Bit 15:	Manufacturer -specific warning	S-0-0181

Tab. 9-7: Warnings of parameter S-0-0012

The State class 2 mask (S-0-0097) can be used to mask the switching off prewarnings with regard to its effect on the change bit.

☞ This signal is only generated for Sercos spindles (**/SP/SpFunc/Sp[1]/Base/SpType** "Spindle type" (1040 00001)).

Change in state class 3 iSp_DrvChangeClass3

This signal reports a change of state class 3 (messages of operating states) and is a copy of the corresponding signal in the Sercos drive status word of the drive in question. The bit is set when state class 3 is modified and reset after reading the status via parameter S-0-0013. It can either be read manually via the NC user interface or automatically by direct PLC reading of S-0-0013.

Parameter S-0-0013 consists of two bytes and contains the following messages:

Bit 0:	$n_{act} = n_{com}$	S-0-0330
Bit 1:	$n_{act} = 0$	S-0-0331
Bit 2:	$\lvert n_{act} \rvert < \lvert n_x \rvert$	S-0-0332
Bit 3:	$\lvert Md \rvert \geq \lvert Md_x \rvert$	S-0-0333
Bit 4:	$\lvert Md \rvert \geq \lvert Md_{lim} \rvert$	S-0-0334
Bit 5:	$\lvert n_{com} \rvert > \lvert n_{lim} \rvert$	S-0-0335
Bit 6:	In Position	S-0-0336
Bit 7:	$\lvert P \rvert \geq \lvert P_x \rvert$	S-0-0337
Bit 8:	-	
Bit 9:	$n_{act} \geq$ min. spindle speed	S-0-0339
Bit 10:	$\lvert n_{act} \rvert \geq$ max. spindle speed	S-0-0340
Bit 11:	In position rough	S-0-0341
Bit 12:	Target position reached	S-0-0342
Bit 13:	Interpolator stopped	S-0-0343
Bit 14:	-	
Bit 15:	Manufacturer-specific operating states	S-0-0182

Tab. 9-8: Messages of parameter S-0-0013

The "State class 3" mask (S-0-0098) can be used to mask the operating states with regard to its effect on the change bit.

☞ This signal is only generated for Sercos spindles (**/SP/SpFunc/Sp[1]/Base/SpType** "Spindle type" (1040 00001)).

Test mode (dry run) iSp_DryRun

This signal indicates whether a spindle has been switched to test mode. In test mode, a spindle is still interpolated internally. However, the calculated command values are only shown on the display and not transmitted to the drive.

The test mode can be automatically activated for unconnected drives (no Sercos ring present) via a softkey or via the machine parameter 1001 00010 **/MAIN/Dr[1]/EnablVirtMode** "Virtual drive".

Enabled for power activation iSp_DrvPower

This signal is closely related to the signals **Starting lockout**, **Drive on**, **Drive ready** and **Drive in operation**.

The functional relationship is shown in detail in the signal diagram for the signal **Starting lockout** (Drive On qSp_DrvOn, page 104) .

Signal level

1: The initialization of the Sercos interface, the spindle drive and the NC control are completed and no errors are present. This signal is a prerequisite to connect power to the drive.

0: The power cannot be connected at the drive.

☞ This signal is only generated for Sercos spindles (**/SP/SpFunc/Sp[1]/Base/SpType** "Spindle type" (1040 00001)).

Drive ready iSp_DrvReady

This signal is closely related to the signals **Starting lockout**, **Drive on**, **Enabled for power activation** and **Drive in operation**.

The functional relationship is shown in detail in the signal diagram for the signal **Starting lockout** (Drive On qSp_DrvOn, page 104) .

Signal level

1: Power is connected to the corresponding drive. The signal remains set as long as the power is not switched off.

0: No power is connected to the drive.

☞ This signal is only generated for Sercos spindles (**/SP/SpFunc/Sp[1]/Base/SpType** "Spindle type" (1040 00001)).

Drive in operation iSp_DrvAct

This signal is closely related to the signals **Starting lockout**, **Drive on**, **Enabled for power activation** and **Drive ready**.

The functional relationship is shown in detail in the signal diagram for the signal **Starting lockout** (Drive On qSp_DrvOn, page 104) .

☞ This signal is only generated for Sercos spindles (**/SP/SpFunc/Sp[1]/Base/SpType** "Spindle type" (1040 00001)).

Signal level

1: The control loop is closed and the drive is under torque. The spindle can be rotated unless it is not locked by any other function, e.g. **Spindle inhibit**.

0: The control loop is open and the drive is not under torque.

☞ When changing from logic 1 to logic 0, spindle stop (M5) is triggered NC-internally.

10 Auxiliary functions

10.1 General information

Auxiliary functions are programmed in the NC part program.

All **bit-coded auxiliary functions** programmed in an NC block are mapped in parallel to flags.

Output of **non-bit-coded auxiliary functions** can be preset using machine parameter **/AUXF/OutBehav/OutBehavMode** "Output mode of the auxiliary functions" 301000015:

0: Only bit-coded auxiliary functions are output in parallel.

1: All auxiliary functions of an NC block are output in parallel. Parallel data transfer avoids machine downtimes.

The following conventions apply to auxiliary functions:

- Every channel has several auxiliary functions. The names (auxiliary function types) of auxiliary functions (M, S, T, ...) are specified in machine parameter **/AUXF/ClassDef/Class[1]/AuxFuncName** "Names of auxiliary functions" (3010 00030).
- *An auxiliary function can be mapped as a*
  - **Bit-coded auxiliary function**
  - **32-bit BCD-coded auxiliary function**
  - **64-bit BCD-coded auxiliary function**
  - **INTEGER-coded auxiliary function (32 bits)**
  - **FLOAT-coded auxiliary function (32 bits)**
  - **DOUBLE-coded auxiliary function (64 bits)**

  on an operand range (machine parameter **/AUXF/ClassDef/Class[1]/EnablAuxFunc, /AUXF/ClassDef/Class[1]/AuxFuncChDep, /AUXF/ClassDef/Class[1]/Bit, /AUXF/ClassDef/Class[1]/Bcd32, /AUXF/ClassDef/Class[1]/Int, /AUXF/ClassDef/Class[1]/Float, /AUXF/ClassDef/Class[1]/Double** "Classes of auxiliary functions" (3010 00040)). These auxiliary functions can be defined both channel-dependently and channel-independently.
- For each channel, several auxiliary function types can be declared as bit-coded auxiliary functions (machine parameter **/AUXF/ClassDef/Class[1]/AuxFuncName** "Names of auxiliary functions" (3010 00030)), e.g. the auxiliary function with the designation **M** and the auxiliary function with the designation **H**.
- The start addresses of bit-coded auxiliary functions are set individually for each channel in machine parameter **/AUXF/PlcIf/ChDep/ChIf[1]/AddrChDepBit** "Addresses of the channel-specific bit-coded auxiliary functions" (2060 00009).
- The output of the auxiliary functions is asynchronously with regard to the procession of the IEC tasks. This means that also within PLC procession, an auxiliary function can be transferred by the CNC to PLC in the flag area.

  If an auxiliary function is output within a PLC cycle and has already been acknowledged, the auxiliary function output cannot be recorded using trace function, since this record is only carried out at the end of the PLC task.

- For this reason, auxiliary functions of type **Double** must always be defined so that they require acknowledgement to ensure the consistency of the auxiliary function value.

  Furthermore, the values of the auxiliary function can only be accessed before the acknowledgement was carried out. The value can change inconsistently after the acknowledgement!
- Only the change signal (bit 0) may be reset to acknowledge non-bit-coded auxiliary functions.
- If several auxiliary functions that are not bit-coded and of the same type are defined, they are all stored one after the other within the flag area. The parameter index (Macoda ID 3010 00080) defines where the auxiliary functions of a type are located. The index always starts with 0.

Assignment of auxiliary function types and addresses in the flag area:

NC	PLC	IEC type	Macoda ID for flag address	
			Channel-dependent	Channel-independent
32 bit BCD	DWORD	MT_AuxFuncBCD32_t	2060 00010	2060 00003[6]
64 bit BCD	DWORD1, DWORD2	MT_AuxFuncBCD64_t	2060 00011	2060 00003[7]
INTEGER	DINT	MT_AuxFuncInt_t	2060 00012	2060 00020[1]
FLOAT	REAL	MT_AuxFuncReal_t	2060 00013	2060 00020[2]
DOUBLE	LREAL	MT_AuxFuncLReal_t	2060 00014	2060 00020[3]

Tab. 10-1: Auxiliary function types

10.2 Bit-coded auxiliary functions

Auxiliary functions are most often required in the PLC sequential program in the form of individual signals. In this way, for example, the coolant is switched on/off, a clamp is opened/closed or a cylinder is extended/retracted.

Auxiliary functions are programmed in the NC part program in order to trigger these functions. The decoded auxiliary functions are mapped on an operand range (e.g. flags). The absolute size of this range is dependent on the number of auxiliary functions which are to be decoded as well as on the number of channels (machine parameter **/MAIN/NofCh** "Number of channels" (9040 00001)).

The start address of the operand range is defined for each channel in machine parameter **/AUXF/PlcIf/ChDep/ChIf[1]/AddrChDepBit** "Addresses of the channel-specific bit-coded auxiliary functions" (2060 00009).

Example:

Bit-coded auxiliary functions

3 channels are defined in the NC. For each channel, 128 (machine parameter **/AUXF/NofAuxFunc/ChDep/ChDepBit** "Number of auxiliary functions [2]" (3010 00010)) decoded auxiliary functions of type **M** are declared (machine parameter **/AUXF/ClassDef/Class[1]/AuxFuncName** "Names of auxiliary functions" (3010 00030)); the operand range is to be at maker M200.0 for channel 0 (machine parameter **/AUXF/PlcIf/ChDep/ChIf[1]/AddrChDepBit** "Addresses of the channel-specific bit-coded auxiliary functions [1]" (2060 00009)). The auxiliary functions of the remaining channels follow one another (machine parameter **/AUXF/PlcIf/ChDep/ChIf[2]/AddrChDepBit** "Addresses of the chan-

nel-specific bit-coded auxiliary functions [2]" (2060 00009) -> M240, Machine parameter **/AUXF/Plclf/ChDep/Chlf[3]/AddrChDepBit** "Addresses of the channel-specific bit-coded auxiliary functions [3]" (2060 00009) -> M280).

Owing to these parameters, the flag field has the following structure:

Channel	M-function	Flag field
0	M31... M0	M203.7 ... M200.0
	M128... M96	M215.7 ... M212.0
1	M31... M0	M243.7 ... M240.0
	M128... M96	M255.7 ... M252.0
2	M31... M0	M283.7 ... M280.0
	M128... M96	M295.7 ... M292.0

Tab. 10-2: Structure of the flag field

In this example, a 16-byte flag field is defined for every channel. In these flag fields, **every bit-coded auxiliary function occupies one flag**.

The bit-coded auxiliary functions may **require acknowledgement** or may **not require acknowledgement** as is defined in the machine parameter **/AUXF/BitAckn/Byte000_015/Byte000 ... /AUXF/BitAckn/Byte176_191/Byte191** "requirement of acknowledgement of bit-coded auxiliary functions" (3010 00020).

Acknowledgement is made by the PLC user program resetting the respective bit-coded auxiliary function.

As long as an auxiliary function which requires acknowledgement is not reset by the PLC user program, the following NC block will not be executed.

Auxiliary functions which do not require acknowledgement must likewise be reset by the PLC user program to detect the changed signal level from logic 0 to logic 1 when the auxiliary function is output again.

Bit coded auxiliary functions are first and foremost channel related, i.e. the assigned bit is set on the channel concerned. With M10, for example, the 11th bit of the channel on which the auxiliary function was output is set.

Depending on the application, operation of the PLC user program can be simplified by assigning a **global** effect to some bit coded auxiliary functions.

For this purpose, a specific auxiliary function of a random channel is assigned to the global auxiliary function in the flag field.

10.3 Non-bit-coded channel-independent auxiliary functions

Apart from the bit-coded format (e.g. of type M), auxiliary functions can also be output in BCD code with 32 or 64 bits or as a binary value in INTEGER (32 bit), FLOAT (32 bit) and DOUBLE (64 bit) formats (such as auxiliary functions of type S).

Non-bit-coded, channel independent auxiliary functions are not assigned to any specific channel and therefore have the same effect on every channel.

An operand area is defined for every non bit-coded type of auxiliary function. Every operand area has either 8 bytes (32-bit BCD, INTEGER and FLOAT value) or 12 bytes (64-bit BCD and DOUBLE value).

The address position of these areas is defined in machine parameter **/AUXF/Plclf/ChIndep/AddrChIndepBcd32** "Operand address (general)" (2060 00003)or **/AUXF/Plclf/ChIndep/AddrChIndepInt, /AUXF/Plclf/ChIndep/**

AddrChIndepFloat, /AUXF/PlcIf/ChIndep/AddrChIndepDouble "Operand addresses: INTEGER, FLOAT, DOUBLE" (2060 00020) defined as offset in the flag area.

Example:

Non-bit-coded channel-independent auxiliary functions

The "S" auxiliary function has been declared to be a 32 bit-coded auxiliary function.

The start address of the flag field of the "S" auxiliary function is to begin at the M20 flag byte.

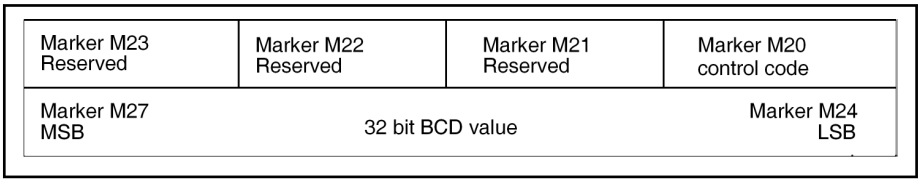


Fig. 10-1: Starting address for the flag field

The control code in flag byte M20 has the following significance:

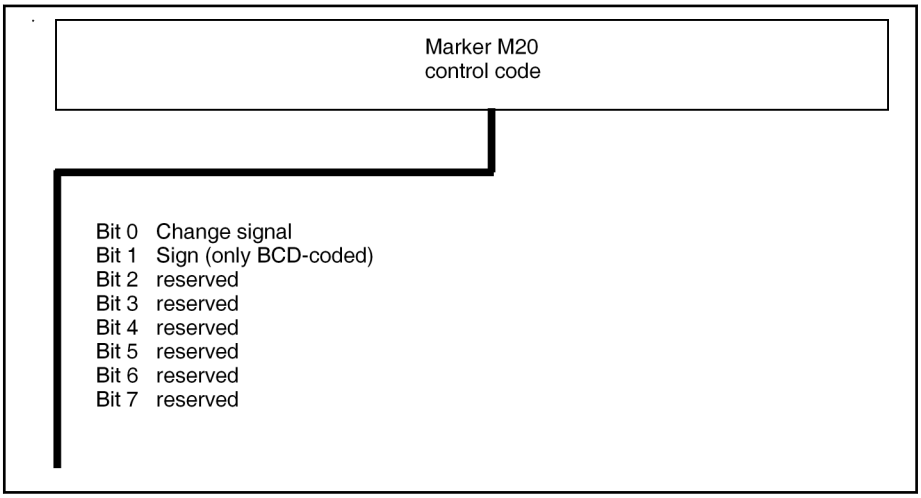


Fig. 10-2: Control code flag byte M20

The example above shows that when a BCD-coded auxiliary function is output to the flag field byte M24 through M27, the BCD value is transferred and the M20.0 change signal is set. The PLC user program must acknowledge the output of the auxiliary function by resetting the change signal on flag M20.0.

If the auxiliary function has been programmed with a negative sign in the part program, the M20.1 flag is also be set.

All BCD-coded auxiliary functions have the same structure as shown in the example above. For 64 bit auxiliary functions, the total length of the corresponding flag fields is 12 bytes instead of 8 bytes.

10.4 Non-bit-coded channel-dependent auxiliary functions

In addition to the non-bit-coded, channel-independent auxiliary functions, these auxiliary functions can also be defined as being channel dependent; in this case, a flag area with the corresponding auxiliary function is assigned to every channel.

The address position of these areas is defined separately for every channel as an offset in the flag area in the machine parameters (-1: auxiliary function area is not used):

- **/AUXF/PlcIf/ChDep/ChIf[1]/AddrChDepBcd32** "Addresses of the channel-specific 32 bit BCD-coded auxiliary functions" (2060 00010)
- **/AUXF/PlcIf/ChDep/ChIf[1]/AddrChDepBcd64** "Addresses of the channel-specific 64 bit BCD-coded auxiliary functions" (2060 00011)
- **/AUXF/PlcIf/ChDep/ChIf[1]/AddrChDepInt** "Addresses of the channel-specific INTEGER-coded auxiliary functions" (2060 00012)
- **/AUXF/PlcIf/ChDep/ChIf[1]/AddrChDepFloat** "Addresses of the channel-specific FLOAT-coded auxiliary functions" (2060 00013)
- **/AUXF/PlcIf/ChDep/ChIf[1]/AddrChDepDouble** "Addresses of the channel-specific DOUBLE-coded auxiliary functions" (2060 00014)

Example:

Non-bit-coded channel-dependent auxiliary functions

The "T" auxiliary function has been declared to be a channel dependent INTEGER coded auxiliary function.

The start address of the flag field of the "T" auxiliary function is to begin:

- at flag field M112 in channel 0
- at flag field M120 in channel 1
- at flag field M128 in channel 2

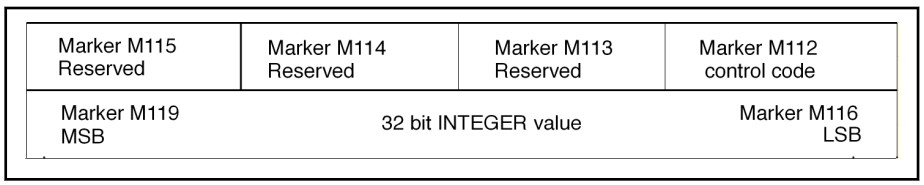


Fig. 10-3: Channel-dependent INTEGER auxiliary function of channel 0

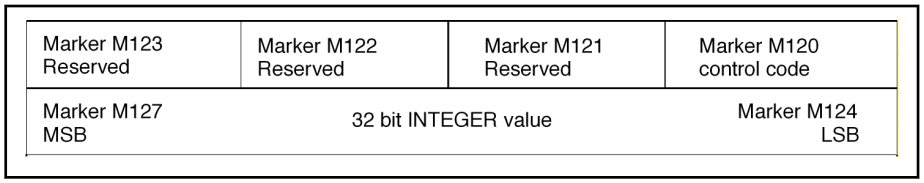


Fig. 10-4: Channel-dependent INTEGER auxiliary function of channel 1

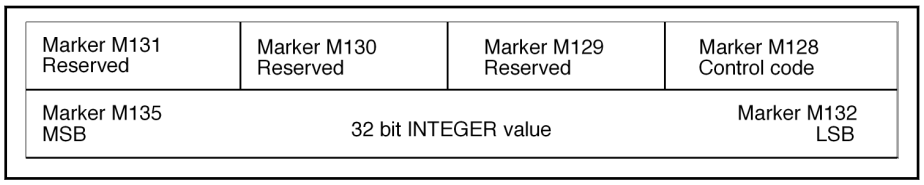


Fig. 10-5: Channel-dependent INTEGER auxiliary function of channel 2

The control code in flag byte M20 has the following significance:

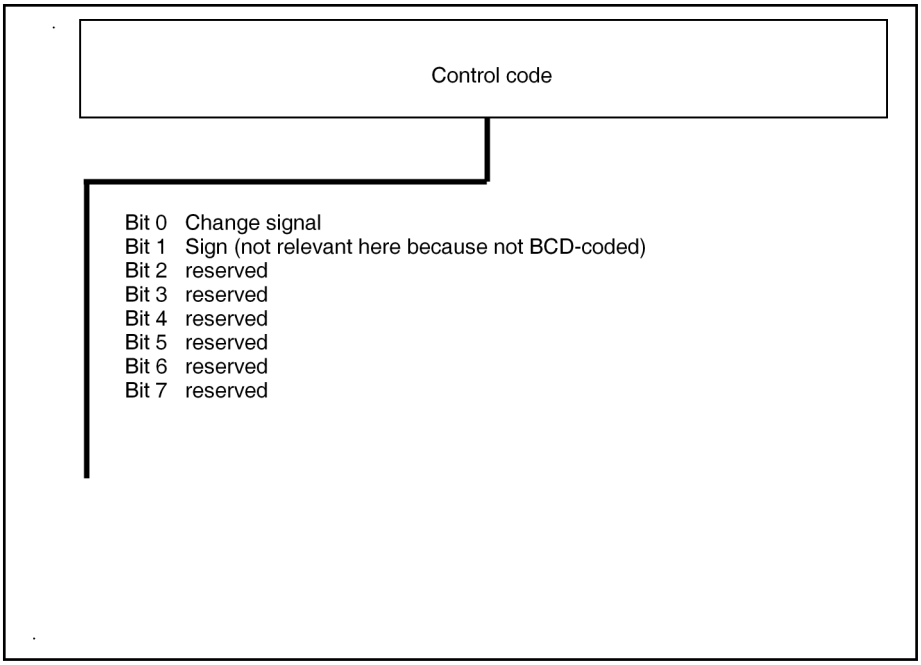


Fig. 10-6: Control code flag byte M20

The example above shows that when an INTEGER coded auxiliary function "T" is output in channel 1 to the flag field with bytes M124 through M127, the binary value is transferred and the M120.0 change signal is also set. The PLC user program has to acknowledge the output of the auxiliary function by resetting the change signal on flag M120.0.

All non-bit-coded auxiliary functions have the same structure as shown in the example above. For 64-bit auxiliary functions, the total length of the corresponding flag fields is 12 bytes instead of 8 bytes.

11 Connecting digital I/Os

Access to digital I/Os from the PLC user program is adapted in the machine parameters of group 4075. The assignment of the I/O signals in the PLC and the use of the PLC addresses are set.

An extension card with digital I/Os is available for the MTX. Furthermore, decentralized I/Os via Profibus DP are possible.

Profibus DP interface

The MTX has a Profibus DP interface which can be used as a **DP master interface** .

This interface connect decentralized Profibus DP periphery components.

The Profibus DP creates the I/O configuration via the IndraLogic user interface. The "DPConfi" diagnostic tool can be used to analyze errors.

☞ An error reported by the DP master in the DP master status does not always force the PLC to stop. If error reaction is required, it has to be programmed in the PLC user program.

12 Program function blocks

12.1 General structure of program function blocks

The integrated program function blocks can be called cyclically and parameterized by the PLC sequential program.

If a function block is called several times before the data from the previous call has already been returned, ensure that another function block instance is used.

An error code is output if an error occurs while executing the function and if it cannot run properly. Errors can be caused by incorrect parameterization such as an invalid channel number. System-internal errors can occur, which are also displayed with a corresponding code.

The **error code** has a length of two bytes.

In the error code, error type or error state are represented with a signed number.

There are two categories of errors and states:

1. **General errors and states**

   These are errors and states caused when **calling** a function with incorrect parameter settings or due to NC-internal errors.

2. **Function-specific errors and states**

   These are errors and states as **acknowledgement** after the execution of a function.

☞ Some of the error messages are function-specific. These are explained in the description of the respective function.

Error codes are defined by Bosch Rexroth and are updated as required with further developments.

The function starts when "Start" is set and "Active" is not set. If "Start" is not reset using the PLC sequential program, it is reset when the function has been completed. "Active" is set until the function is completed. As soon as data is returned, the function is completed and "Active" is reset. The function can be completed within one call.

The programming example of the first program function block is shown in the PLC programming languages LD, FBD , IL and ST. All other programming examples are shown only in LD.

12.2 Reading actual axis values (MT_ScsPos)

12.2.1 Function

This program function block provides the **actual values of all axes** at the time of request.

The actual axis value returned is scaled to 0.1 µm and represented as integer with four bytes.

When calling the block cyclically, ensure that the block is not called unnecessarily often, thus placing an unnecessary load on the rest of the system. This can be achieved, for example, by means of timer programming, time-controlled blocks or by calling the block only in every nth cycle.

The function block can be called once or multiple times in a PLC cycle. Calls can be absolute or conditional.

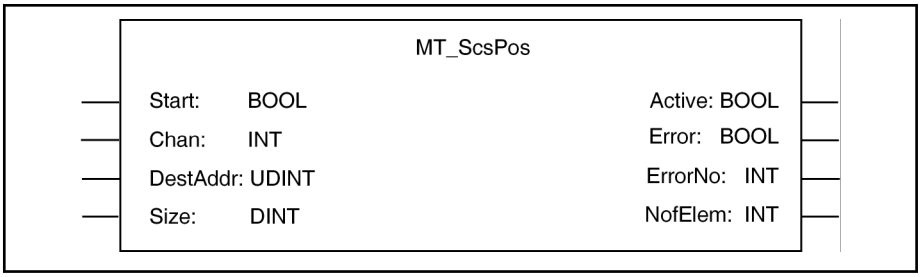


Fig. 12-1: MT_ScsPos function block

Function block parameters

The function block is called with the following parameters:

Start:	Flag for new request
Chan:	Desired channel number
DestAddr:	Address for data (ADR(<Variable>))
Size:	Max. size of the memory for the data in bytes (SIZEOF(<Variable>)

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors
ErrorNo:	Error code: only valid if Error = TRUE
NofElem:	Number of returned elements

☞ This function block provides the actual values of all axes irrespective of the transferred channel number.

The function starts when **Start** is set and **Active** is not set. If "Start" is not reset using the PLC sequential program, it is reset when the function is completed.

Active is set until the function is completed. As soon as data is returned, the function is completed and Active is reset.

12.2.2 Configuration example

Three axes are defined in channel 1. The actual axis values are to be queried with the positive edge of the PLC input %IX40.2.

The task is triggered with a rising edge at the %IX40.2 input.

Given these specifications, the **MT_ScsPos** program function block call can be structured as follows:

Program example in ST (declaration)

```
VAR
  R_TRIG_Inst :     R_Trig;
  StartGetAxPos :   BOOL;
  waitForData :     BOOL;
  Channel :         INT;
  AxesPos :         ARRAY[0..2] OF DINT;
  GetAxesPos :      MT_ScsPos;
  AxPos_X :         DINT;
  AxPos_Y :         DINT;
  AxPos_Z :         DINT;
END_VAR
```

Program example in ST (implementation)

```
(* Definitions *)
Channel := 1;
AxPos_X := 0;
AxPos_Y := 0;
AxPos_Z := 0;

(* Requesting the axis positions via a rising edge of IX 40.2 *)
R_TRIG_Inst ( CLK := %IX40.2 );
StartGetAxPos := R_TRIG_Inst.Q;

(* Requesting the axis positions *)
GetAxesPos (Start:=StartGetAxPos, Chan:=Channel, DestAddr:=ADR(AxesPos), Size:=SIZEOF(AxesPos));

(* Set waitForData flag as long as the function is active *)
IF ( GetAxesPos.Active = TRUE ) THEN
  waitForData := GetAxesPos.Active;
END_IF
IF ( GetAxesPos.Active OR NOT waitForData ) THEN
  RETURN;
END_IF

(* Reset waitForData flag *)
waitForData := FALSE;

(* Error handling when the error occurs *)
IF ( GetAxesPos.Error = TRUE ) THEN
  (* Error handling *)
  RETURN;
ELSE
  IF ( GetAxesPos.NofElem >= 3 ) THEN
    (* Save new axis position *)
    AxPos_X := AxesPos[0];
    AxPos_Y := AxesPos[1];
    AxPos_Z := AxesPos[2];
  END_IF
END_IF
```

12.2.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

1:	Invalid channel number
-6:	Currently there are no resources available to transmit the task
-3:	Data overflow: Only the number of data the memory can save is returned
-1:	general parameter error

12.3 Program selection/deselection (MT_ProgSel)

12.3.1 Function

This program function block allows to **select and deselect an NC part program** via the PLC sequential program.

Mode 3 **(Program change)** allows another program to be selected without resetting the modal functions. This can be used, for example, to select a new program while the spindle is running if "M5" is removed in the switch-on block. The prerequisite for this is that the last program has been completed or has not yet been started.

The external program selection/deselection function block can be called absolutely or conditionally. If several channels are activated, the function block can also be called several times per PLC cycle.

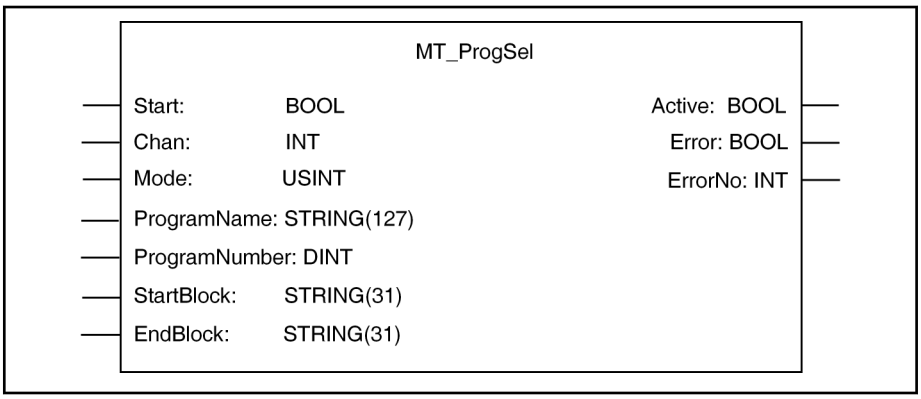


Fig. 12-2: Function block program selection/deselection

Function block parameters

The function block is called with the following parameters:

Start:	Flag for new request
Chan:	Desired channel number
Mode:	1 program selection
	2 program selection with automatic deselection of the old program
	3 program change
	4 program deselection
ProgramName:	Program name (incl. path)

If an empty string is transferred in the parameter **ProgramName**, the program number (with a preceding "P") is used as program name. The MT_ProgSel function block converts the program number into ASCII characters, since the part programs in the MTX are stored according to their characters and not numerically.

If a program to be selected is specified by the program name (ASCII string), ensure that the program name is limited to 30 characters (including the NUL ASCII character as end identification).

For programs that have to be linked (programs or subroutines that contain CPL instructions), only 28 characters are permitted, since two additional characters are required for the extension in the link table.

If the search path to select a program is to be different from the NC-internal search path defined in the machine parameters, the respective directory containing the program can be transferred together with the program name. However, the total length of 127 characters must not be exceeded. For example, to ensure that the "test.cpl" program is always selected from the user FEPROM, the program name to be transferred has to be "/usrfep/test.cpl".

ProgramNumber:	Program number
StartBlock:	Starting block (empty string: no starting block)
EndBlock:	End block (empty string: no end block)

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors
ErrorNo:	Error code: only valid if Error = TRUE

An NC part program can only be selected if **Automatic** mode is active.

The function starts when **Start** is set and **Active** is not set. If "Start" is not reset using the PLC sequential program, it is reset when the function is completed.

Active is set until the function is completed. As soon as the program has been selected/deselected, the function is completed and "Active" is reset.

The **Mode** parameter specifies whether a program is selected or an already active program is to be deselected.

To deselect a program, only the channel number has to be specified. The channel number is always entered as integer.

If an empty string is transferred to the parameter **ProgramName**, the program number (with a preceding "P") is used as the program name. The MT_ProgSel function block converts the program number into ASCII characters, since the part programs in the MTX are stored according to their characters and not numerically.

If the selection of the program is specified via the program name (ASCII string), it has to be ensured that the program name does not contain more than 29 characters and that the character string is terminated with the ASCII character NUL. If the program is not to be selected according to the search path defined in the machine parameters within the NC, the program name can be transferred with the desired directory. However, the total length of 30 characters must not be exceeded. For example, to ensure that the "test.cpl" program is always selected from the user FEPROM, the program name to be transferred has to be "/usrfep/test.cpl".

If the part program is to be started at a certain block, a **starting block** can be defined in addition to the program name. The starting block can be an NC block according to DIN 66025 (always begins with "N") or a CPL block (no "N" in front of the block number). Therefore, the starting block is defined as character string. The starting block length is limited to 9 ASCII characters and has to be completed with the ASCII character NUL at the end.

If the part program does not complete machining (M30) but is to terminate before the end, an **end block** can be defined. The same rules as for defining starting blocks apply to end blocks.

If a program is still active when another program is to be selected, it is sufficient to select the program with mode 2 or 3. Accordingly, a prerequisite for the automatic program deselection is that the NC is either in **Ready** or **Stop** state.

12.3.2 Configuration example

The NC part program with the program name "Labor.npg" is to be activated in channel 1.

The task is triggered with a rising edge at the %IX40.3 input.

The **MT_ProgSel** function block could be called as follows:

Program example in ST (declaration)

```
VAR
  R_TRIG_Inst :      R_Trig;
```

```
  StartProgSelect :  BOOL;
  Mode :             USINT;
  Channel :          INT;
  ProgName :         STRING(31);
  ProgramSelect :    MT_ProgSel;
  activeProgSelect : BOOL;
END_VAR
```

Program example in ST (implementation)

```
(* Definitions *)
Mode := 1;
Channel := 1;
ProgName := 'Labor.npg';

(* Rising edge -->StartProgSelect *)
R_TRIG_Inst ( CLK := %IX40.3 );
StartProgSelect := R_TRIG_Inst.Q;

(* Call ProgramSelect *)
ProgramSelect (Start:=StartProgSelect, Chan:=Channel, Mode:= Mode,
               Programmname:=ProgName, StartBlock:='' ,EndBlock := '');
activeProgSelect := ProgramSelect.Active;

(* Error handling *)
IF ( ProgramSelect.Error = TRUE ) THEN
  (* Error handling *)
END_IF
```

12.3.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-1:	General parameter error

General errors and states as acknowledgement after executing the function:

1:	Invalid channel number

General errors and states occurring during program selection:

2:	Channel is assigned
3:	The control reset function is currently executed in the channel
4:	Program name is too long
5:	No reference points approached
6:	Program cannot be opened
7:	Error during program selection
8:	Either a wrong or no operation mode is selected
25:	Syntax error in the selected program

General errors and states occurring during program deselection:

3:	The control reset function is currently executed in the channel
10:	No program is selected
11:	Program is running and feed hold is not present

12.4 NC block specification (MT_NcBlk)

12.4.1 Function

This program function block allows to specify an NC block via the PLC sequential program. The NC channel has to be in MDI or in automatic mode. The NC block input can be specified while the program is running provided no axis addresses with an impact on the running machining processes are specified. Thus, for example, command values for auxiliary axes or auxiliary functions can be transferred to the NC.

If an NC block with an impact on the running machining process is specified (such as feed specification or a synchronous axis programming), a syntax error results.

The specified block has to be processed completely before another block can be specified in the same channel.

This function block can be called absolutely or conditionally.

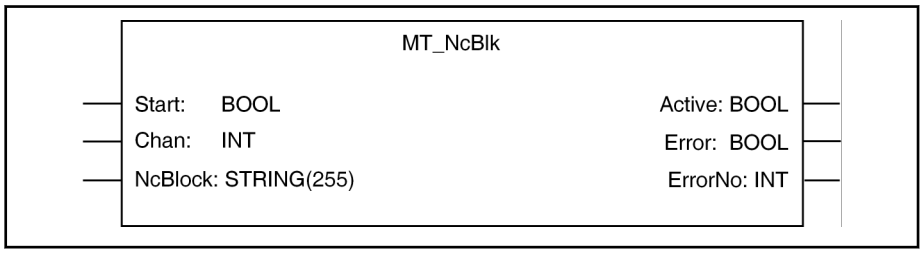


Fig. 12-3: MT_NcBlk function block

Function block parameters

The function block is called with the following parameters:

Start:	Flag for new request
Chan:	Desired channel number
NcBlock:	NC block to be executed

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors
ErrorNo:	Error code: only valid if Error = TRUE

The function starts when **Start** is set and **Active** is not set. If **Start** is not reset using the PLC sequential program, it is reset when the function has been completed.

Active is set until the function has been completed, i.e. the specified block became active. However, the motion does not have to be completed yet.

When an asynchronous motion is specified whose end position has not been reached yet, it is ensured that the axis travel command is already set at the time the function is completed.

When the function is completed, the status "Active" is reset.

The NC executes the NC block immediately after the reception without an additional NC start signal.

12.4.2 Configuration example

The NC block "G91 X10 Y5 Z1 F10000" is to be transferred to channel 1.

The **MT_NcBlk** function block could be called as follows:

Program example in ST (declaration)

```
VAR
  R_TRIG_Inst :      R_Trig;
  StartNcBlockReq :  BOOL;
  Channel :          INT;
  NcSatz :           STRING(255);
  NcBlockReq :       MT_NcBlk;
  activeNcBlockReq : BOOL;
END_VAR
```

Program example in ST (implementation)

```
(* Definitions *)
Channel := 1;
NC block := 'G91 X10 Y5 Z1 F10000';

(* Rising edge --> StartNcBlockReq *)
R_TRIG_Inst ( CLK := %IX41.3 );
StartNcBlockReq := R_TRIG_Inst.Q;

(* Call NcBlockReq *)
NcBlockReq (Start:=StartNcBlockReq, Chan:=Channel, NcBlock:=NcSatz);
activeNcBlockReq := NcBlockReq.Active;

(* Error handling *)
IF ( NcBlockReq.Error = TRUE ) THEN
  (* Error handling *)
END_IF
```

12.4.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-1:	General parameter error

Function-specific errors and states as acknowledgement after executing the function:

1:	Invalid channel number
2:	Channel is assigned
3:	The control reset function is currently executed in the channel
4:	NC block is too long
5:	No reference points approached
7:	NC block selection was cancelled via control reset
8:	Either a wrong or no operation mode is selected
25:	A runtime error was detected in the NC block specification, e.g. a syntax error

12.5 Extended NC block specification (MT_NcBlkExt)

12.5.1 Function

The program function block **Extended NC block selection** allows to specify any NC block via the PLC sequential program. Via the input parameter **Options**, the processing of the specification on NC side can be influenced. The

NC channel has to be in MDI or in automatic mode. The NC block input can be specified while the program is running provided no axis addresses with an impact on the running machining processes are specified. Thus, for example, command values for auxiliary axes or auxiliary functions can be transferred to the NC.

If an NC block with an impact on the running machining process is specified (such as feed specification or a synchronous axis programming), a syntax error results.

The specified block has to be processed completely before another block can be specified in the same channel.

This function block can be called absolutely or conditionally.

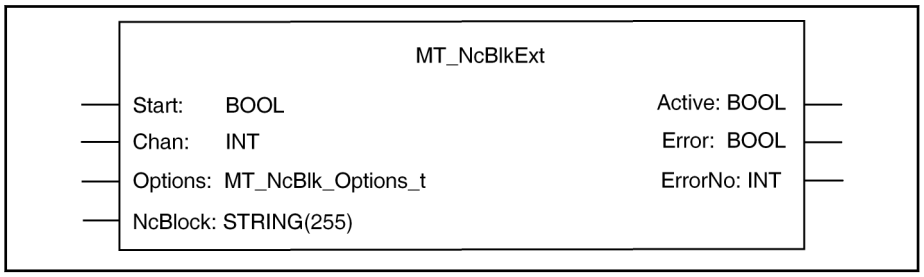


Fig. 12-4: Calling MT_NcBlkExt function block

Function block parameters

The function block is called with the following parameters:

Start:	Flag that executes a new request
Chan:	Desired channel number

Options:	Options is of type "MT_NcBlk_Options_t" and can apply the following values to the control of the block specification:
	`NO_AUTO_START`
	No automatic start after activating the NC block. The processing of the NC block has to be started via the signal qCh_Nc start by the PLC.
	`NO_CHAN_CHANGE`
	If the transferred channel is assigned, the task is not detoured automatically into the "AutoStart" channel (auxiliary channel).
	`PROCESS_DELETE`
	If the block specification is executed in the channel, a program is active (ready) or feed hold is active, it is deselected and the specified NC block becomes active

NcBlock:	NC block to be executed

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors
ErrorNo:	Error code: only valid if Error = TRUE

The function starts when the **Start** bit is set and "Active" is not set. If the "Start" bit is not reset using the PLC sequential program, it is reset when the function has been completed.

Active is set until the function has been completed, i.e. the specified block became active. However, the motion does not have to be completed yet.

When an asynchronous motion is specified whose end position has not been reached yet, it is ensured that the axis travel command is already set at the time the function is completed. The end of the traversing motion can be requested via the "iAx_InPos" interface signal.

When the function is completed, the status "Active" is reset.

If the option bit NO_AUTO_START is not set, the NC executes the NC block immediately after the reception without an additional NC start signal.

12.5.2 Configuration example

The NC block **"G91 X10 Y5 Z1 F10000"** is to be transferred to channel 1 and there should be **no detour** in the auto start channel.

The **MT_NcBlkExt** function block could be called as follows:

Program example in ST (declaration)

```
VAR
  R_TRIG_Inst :       R_Trig;
  StartNcBlockReq :   BOOL;
  Channel :           INT;
  NcSatz :            STRING(255);
  Options :           MT_NcBlk_Options_t;
  NcBlockReq :        MT_NcBlkExt;
  activeNcBlockReq :  BOOL;
END_VAR
```

Program example in ST (implementation)

```
(* Definitions *)
Channel := 1;
NC block := 'G91 X10 Y5 Z1 F10000';
Options := Options.NO_CHAN_CHANGE;

(* Rising edge --> StartNcBlockReq *)
R_TRIG_Inst ( CLK := %IX41.3 );
StartNcBlockReq := R_TRIG_Inst.Q;

(* Call NcBlockReq *)
NcBlockReq (Start:=StartNcBlockReq, Chan:=Channel, Options:=Options, NcBlock:=NcSatz);
activeNcBlockReq := NcBlockReq.Active;

(* Error handling *)
IF ( NcBlockReq.Error = TRUE ) THEN
  (* Error handling *)
END_IF
```

12.5.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-1:	General parameter error

Function-specific errors and states as acknowledgement after executing the function:

1:	Invalid channel number
2:	Channel is assigned
3:	The control reset function is currently executed in the channel
4:	NC block is too long

5:	No reference points approached
7:	NC block selection was cancelled via control reset
8:	Either a wrong or no operation mode is selected
25:	A runtime error was detected in the NC block specification, e.g. a syntax error

12.6 Channel restart (MT_ChanRestart)

12.6.1 Function

The program function block **Channel restart** restarts the channel from the beginning of the selected main program. There is no reset. Thus, the switch-on state is also not applied at reset.

ATTENTION: The modal states are not modified! At the beginning of the main program, all modal functionalities used during the program have to be reset, since all modal functions active at channel restart remain active.

This function block can be called absolutely or conditionally.



Fig. 12-5: MT_ChanRestart function block

Function block parameters

The function block is called with the following parameters:

Start:	Flag that executes a new request
Chan:	Desired channel number
Options:	For future extensions. Options of type "MT_ChanRestart_Options_t".

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors. More exact diagnostics via "ErrorID" and "ErrorIdent"
ErrorId:	Error code: only valid if Error = TRUE
ErrorIdent:	Detailed error description (for future extensions)

The function starts when the **Start** bit is set and "Active" is not set. If the "Start" bit is not reset using the PLC sequential program, it is reset when the function has been completed.

Active is set while processing the channel restart.

When the function is completed, the status "Active" is reset.

12.6.2 Configuration example

The channel entered in "ChanNo" is to be restarted.

The **MT_ChanRestart** function block could be called as follows:

Program example in ST (declaration)

```
VAR
  R_TRIG_Inst :          R_Trig;
  StartChanRestart :     BOOL;
  ChanNo :               INT;
  Options :              MT_ChanRestart_Options_t;
  ChanRestartReq :       MT_ChanRestart;
  activeChanRestartReq : BOOL;
END_VAR
```

Program example in ST (implementation)

```
(* Definitions *)
Channel := 1;
Options := 0;

(* Increasing edge --> StartChanRestart *)
R_TRIG_Inst ( CLK := %IX41.2 );
StartChanRestart := R_TRIG_Inst.Q;

(* Call ChanRestartReq *)
ChanRestartReq (Start:=StartChanRestartReq, Chan:=ChanNo, Options:=Options);
activeChanRestartReq := ChanRestartReq.Active;

(* Error handling *)
IF ( ChanRestartReq.Error = TRUE ) THEN
  (* Error handling *)
END_IF
```

12.6.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorId" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-1:	General parameter error

Function-specific errors and states as acknowledgement after executing the function:

2:	Channel restart is already active
37:	A channel restart is not possible due to a channel error
38:	A channel restart is invalid due to the current operation mode and the current channel state

12.7 Temperature Compensation (MT_TempComp)

12.7.1 Function

This program function block allows to specify a correction value to compensate the temperature impact via the PLC sequential program. The PLC program transfers the temperature correction value for a certain number of axes and becomes immediately active after transfer. The correction value does not be effective depending on the position.

To prevent servo errors, the differences of the correction values must not exceed the jumping ability of the axes.

This function block can be called absolutely or conditionally.

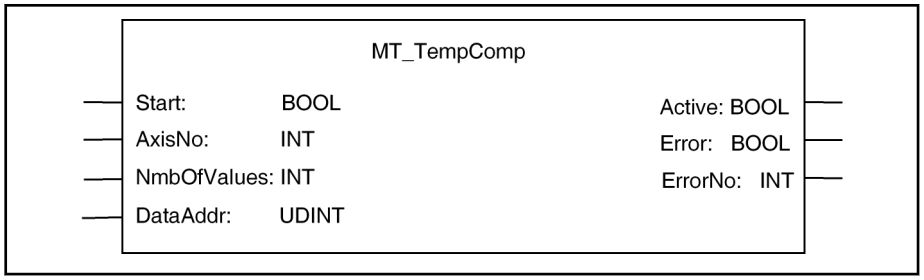


Fig. 12-6: Calling MT_TempComp function block

Function block parameters

The function block is called with the following parameters:

Start:	Flag for new request
AxisNo:	Physical axis index. The values are applied from now on
NmbOfValues:	Number of correction values
DataAddr:	Address of data (array for LREAL values)

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors
ErrorNo:	Error code: only valid if Error = TRUE

The function starts when **Start** is set and **Active** is not set. If "Start" is not reset using the PLC sequential program, it is reset when the function is completed.

Active is set until the function is completed. As soon as the NC has acknowledged the function, the function is completed and "Active" is reset.

The physical axis index has to be entered in **AxisNo** and the number of correction values has to be entered in **NmbOfValues**. On the NC side, the transferred temperature compensation values are calculated starting from the axis with the transferred "axis index". The axis index starts at 1 for the first axis.

The temperature compensation is applied immediately by the NC.

12.7.2 Configuration example

To compensate the temperature impact, a correction value of 123 µm is to be specified for the fourth axis and a correction value of 234 µm for the fifth axis.

The task is triggered with a rising edge at the %IX41.3 input.

The **MT_TempComp** function block could be called as follows:

Program example in ST (declaration)

```
VAR
  R_TRIG_Inst :       R_Trig;
  StartTempComp :     BOOL;
  FirstAxis :         INT;
  TempValues :        ARRAY[0..5] OF LREAL;
  TempCompReq :       MT_TempComp;
  activeTempCompReq : BOOL;
END_VAR
```

Program example in ST (implementation)

```
(* Definitions *)
FirstAxis := 4;
TempValues[4] := 0.123;
TempValues[5] := 0.234;
```

```
(* Rising edge --> StartTempComp *)
R_TRIG_Inst ( CLK := %IX41.3 );
StartTempComp := R_TRIG_Inst.Q;

(* Call TempCompReq *)
TempCompReq (Start:=StartTempComp, AxisNo:=FirstAxis, NmbOfValues:=2,
             DataAddr:=ADR(TempValues[4]));
activeTempCompReq := TempCompReq.Active;

(* Error handling *)
IF ( TempCompReq.Error = TRUE ) THEN
  (* Error handling *)
END_IF
```

12.7.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-1:	General parameter error

Function-specific errors and states as acknowledgement after executing the function:

16:	Too many correction values (possibly displayed with the axis number)
24:	Invalid axis number
33:	The last correction value has not yet been applied by the interpolator

12.8 Reading machine parameters (MT_CfgData)

12.8.1 Function

This program function block provides the values of a machine parameter block.

The variables for the values have to be specified sufficiently high to store all individual parameters of the block.

This function block does not convert value formats. Thus, real values are returned as real numbers.

The function block can be called once or multiple times in a PLC cycle. Calls can be absolute or conditional.

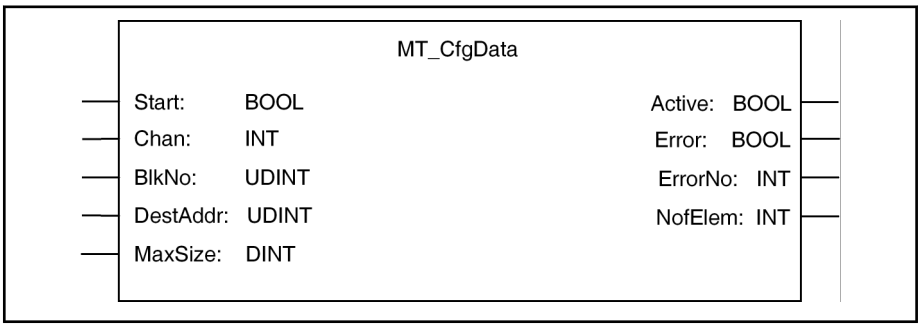


Fig. 12-7: Calling MT_CfgData function block

Function block parameters

The function block is called with the following parameters:

Start:	Flag for new request
Chan:	Desired channel number for channel-specific machine parameters
BlkNo:	Number of machine parameter block
DestAddr:	Address for data
Size:	Max. size of the memory for the data in bytes

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors
ErrorNo:	Error code: only valid if Error = TRUE
NofElem:	Number of returned elements

The function starts when **Start** is set and **Active** is not set. If "Start" is not reset using the PLC sequential program, it is reset when the function is completed.

Active is set until the function is completed. As soon as data is returned, the function is completed and Active is reset.

While reading a channel-specific block, the values that refer to the channel transferred in **Chan** are read. If the value transferred in Chan is -1, the control reset of the block is returned.

For channel-independent parameters, the value in "Chan" is ignored.

12.8.2 Configuration example

The values of the machine parameter **/SP/SpFunc/Sp[1]/Gear/GearStep[1]/MaxSpAcc** "Maximum spindle acceleration" (1040 00031) are to be queried at the positive edge of the PLC input signal %IX40.7.

The task is triggered with a rising edge at the %IX40.7 input.

Given these specifications, the **MT_CfgData** program function block call could be structured as follows:

Program example in ST (declaration)

```
VAR
  R_TRIG_Inst :          R_Trig;
  StartCfgData :         BOOL;
  Channel :              INT;
  BlockNumber :          UDINT;
  BlockValue :           ARRAY[0..2] OF DINT;
  GetConfigData :        MT_CfgData;
  activeGetConfigData :  BOOL;
  MaxSpeedUp :           UDINT;
END_VAR
```

Program example in ST (implementation)

```
(* Definitions *)
Channel := 1;
BlockNumber := 104000031;

(* Rising edge --> StartCfgData *)
R_TRIG_Inst ( CLK := %IX40.7 );
StartCfgData := R_TRIG_Inst.Q;

(* Call GetConfigData *)
GetConfigData (Start:=StartGetCfgData, Chan:=Channel, BlkNo:=BlockNumber,
               DataAddr:=ADR(BlockValue), MaxSize:=SIZEOF(BlockValue);
activeGetConfigData := GetConfigData.Active;
```

```
(* Error handling *)
IF ( GetConfigData.Error = TRUE ) THEN
  (* Error handling *)
END_IF

(* Transfer: Save maximum spindle acceleration of the first spindle *)
MaxSpeedUp := BlockValue[1];
```

12.8.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-3:	Data overflow: Only the number of data the memory can save is returned
-1:	General parameter error

Function-specific errors and states as acknowledgement after executing the function:

100:	Invalid machine parameter number

12.9 Handwheel data (MT_Handw)

12.9.1 Function

For handwheels in relation with Profibus DP, this program function block transfers the current handwheel value including the handwheel number to the NC. Since the data is transmitted via the Profibus DP, the data from the input range has to be transferred to the NC using this program function block

Before applying data, a handwheel has to be selected using a rising edge at the respective AxisInterface (handwheel selection bits 0, 1).

The function block can be called once or multiple times in a PLC cycle. Calls can be absolute or conditional.

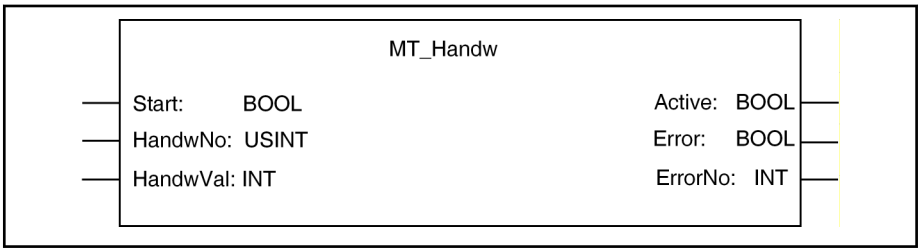


Fig. 12-8: Calling MT_Handw function block

Function block parameters

The function block is called with the following parameters:

Start:	Static signal for a new request
HandwNo:	Handwheel number
HandwVal:	Counter reading of handwheel

The return parameters are:

Active:	Signal that indicates whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors
ErrorNo:	Error code: only valid if Error = TRUE

The function starts when **Start** is set and **Active** is not set. If "Start" is not reset using the PLC sequential program, it is reset when the function is completed.

Active is set until the function is completed. As soon as the data is transferred, the function is completed and "Active" is reset.

12.9.2 Configuration example

The increments for the handwheel are stored starting at the input 80. They are to be transferred to the NC as long as the input %IX40.7 is TRUE.

The **MT_Handw** function block can be called as follows:

Program example in ST (declaration)

```
VAR
  R_TRIG_Inst :          R_Trig;
  StartHandw :           BOOL;
  HandwValue :           INT;
  HandWheelData :        MT_Handw;
  activeHandWheelData : BOOL;
END_VAR
```

Program example in ST (implementation)

```
(* Definition *)
HandwValue := %IW80;

(* Rising edge --> StartHandw *)
R_TRIG_Inst ( CLK := %IX40.7 );
StartHandw := R_TRIG_Inst.Q;

(* Call HandWheelData *)
HandWheelData (Start:=StartHandw, HandwNo:=1, HandwVal:=HandwValue);
activeHandWheelData := HandWheelData.Active;

(* Error handling *)
IF ( HandWheelData.Error = TRUE ) THEN
  (* Error handling *)
END_IF
```

12.9.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-1:	General parameter error

Function-specific errors and states as acknowledgement after executing the function:

1:	Invalid handwheel number

12.10 Sercos parameters (MT_ScsData)

12.10.1 Function

This program function block allows to read and specify a parameter value within the drives. The format to be read or specified is a 4-byte unsigned number (UDINT) as stored in the drives. Any conversion into another format has to be executed in the user program.

When a value is read, the function also provides the attribute that describes the parameter. When a value is specified for the drives, the attribute has to be transferred as well.

When reading from the cyclic telegram (mode 4 and 5), the function block provides the values of the ident number from all applied drives as array of UDINT, whereas the value 16#8000000 is provided for drives in which the ident number is not in the cyclic telegram.

This function block can be called absolutely or conditionally.

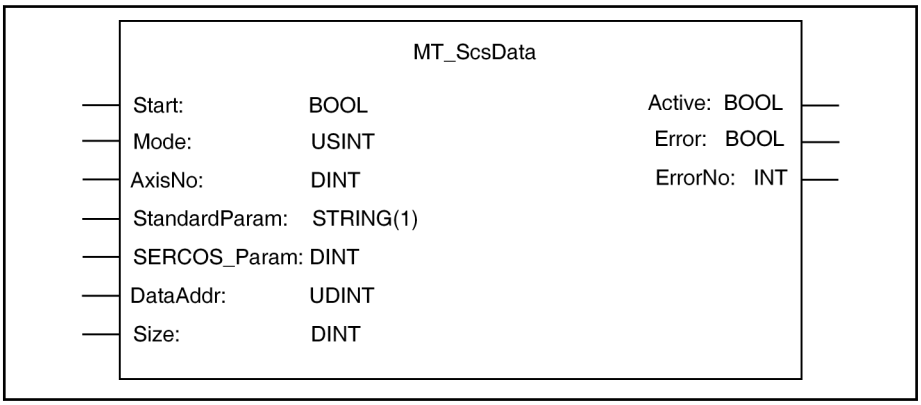


Fig. 12-9: Calling MT_ScsData function block

☞ Repeated function block calls with data request/specification via the service channel (non-cyclic data) can increase the required capacity of the service channel! Therefore, the recommended interval between two function block calls is 500 ms approximately!

☞ Using the Sercos multiplex channel accelerates the data transfer.

Function block parameters

The function block is called with the following parameters:

Start:	Flag for new request
Mode:	1 read attribute and content of the parameter
	2: Specify new value for the parameter
	3: Read the attribute.
	4: Read the content of the parameter from the cyclic telegram
	The values are provided at a minimum temporal interval of 100 ms. The values for all axes are provided
	5: Read the content of the parameter from the cyclic telegram with no temporal delay
	The values for all axes are provided

AxisNo:	Physical axis number; relevant only for the modes 1, 2 and 3 The axis index starts at 1 for the first axis
StandardParam:	"S" = standard parameter "P" = product-specific parameter
Sercos_Param:	Parameter number
DataAddr:	Address of data. The format of data depends on the **mode** transferred during request: • During **Reading/specifying individual values** (modes 1, 2 and 3) data of type **MT_ScsData_t** is returned: **TypeMT_ScsData_t** **Struct** scsAtrr: UDINT; (*Parameter attribute*) scsVal: UDINT; (*Parameter value*) **End_Struct** **End_type** • While reading data from the cyclic telegram (modes 4 and 5), the values of all axes are returned as UDINT array. The value 16#80000000 is returned for drives which are not defined in cyclic telegram for the transferred ident number.
Size:	Memory size for the data in bytes

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors
ErrorNo:	Error code: only valid if Error = TRUE

The function starts when **Start** is set and **Active** is not set. If Start is not reset using the PLC sequential program, Start is reset when the function has been completed.

Active is set until the function is completed. As soon as the data is returned/specified, the function is completed and "Active" is reset.

12.10.2 Configuration example

The current value of the Sercos parameter S-0-0108, Feedrate override of physical axis 1 is to be set to 50 %.

The task is triggered with a rising edge at the %IX45.0 input.

The **MT_ScsData** function block could be called as follows:

Program example in ST (declaration)

```
VAR
  instScsData :          MT_ScsData;
  StartScsData :         BOOL;
  ScsDataMode45 :        ARRAY[1..250] OF DINT;
  ScsDataMode123 :       MT_ScsData_t;// Atribute must be read before writing
  Mode :                 USINT;
  AxNo :                 DINT; // -1 = all exes for mode 4 or 5,
                               // otherwise the number of the axis
  ParaType :             STRING(1);
  ParaNo :               DINT;
  DataAdr :              UDINT;
  DataSize :             DINT;
END_VAR
```

Program example in ST (implementation)

```
(* Definitions */
AxNo := 1;
ParaType := "S";// Feedrate override
ParaNo := 108;
ScsValueMode123.scsVal := 5000; // 50%
Mode := 2;

StartScsData := ...;

(* Determine DataAddr and DataType *)
IF StartScsData THEN
    IF AxNo = -1 THEN
      DataAdr:= ADR(ScsDataMode45); DataSize:= SIZEOF(ScsDataMode45);
    END_IF
    IF AxNo > 0 AND AxNo <= 250 THEN
      DataAdr:= ADR(ScsDataMode123); DataSize:= SIZEOF(ScsDataMode123);
    END_IF
END_IF

(* Call instScsData *)
instScsData(Start:= StartScsData, Mode:= Mode, AxisNo:= AxNo, StandardParam:= ParaType,
            SERCOS_Param:= ParaNo, DataAddr:= DataAdr, Size:= DataSize,
            Active=> , Error=> , ErrorNo=> );

IF instScsData.Active THEN
  StartScsData := FALSE;
END_IF

(* Error handling *)
IF instScsData.Error THEN
  // error handling *)
END_IF
```

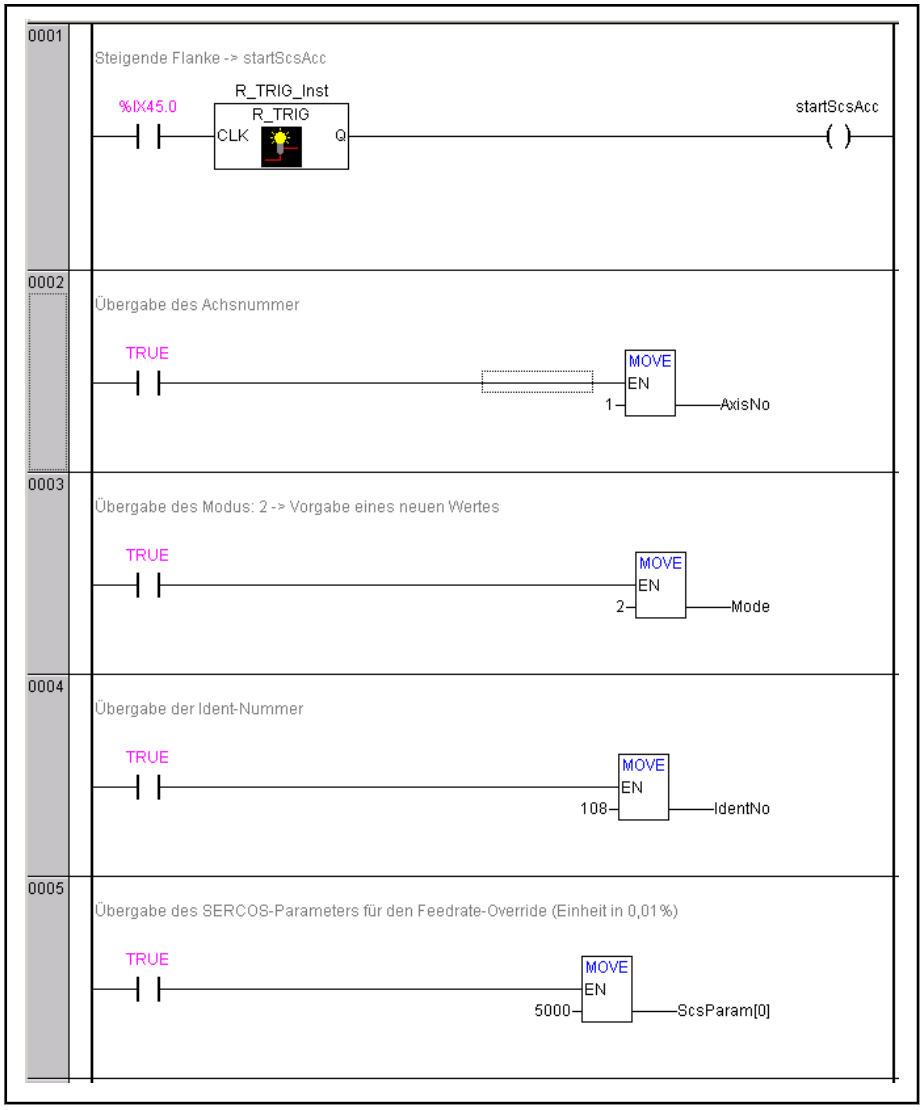


Fig. 12-10: Calling MT_ScsData function block

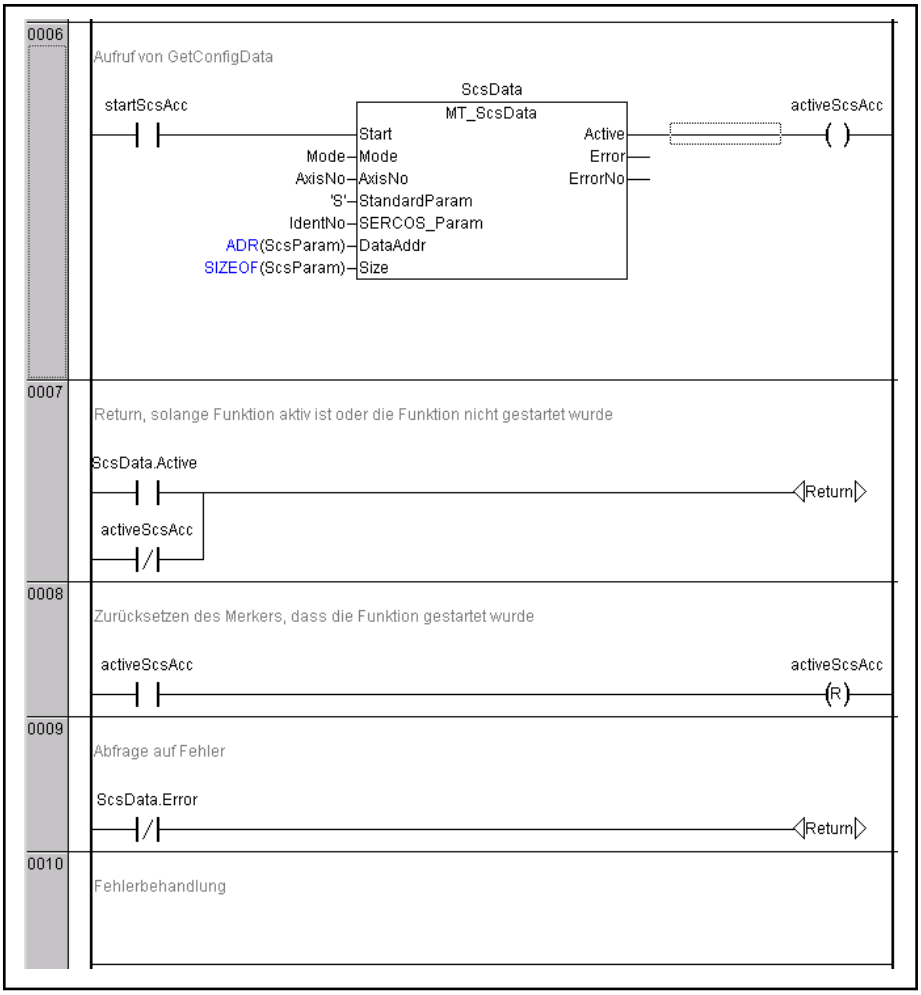


Fig. 12-11: Calling MT_ScsData function block

12.10.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-3:	Data overflow: Only the number of data the memory can save is returned
-1:	General parameter error

Function-specific errors and states as acknowledgement after executing the function:

3:	NCS channel currently assigned
6:	Parameter number unknown or invalid
9:	Access conflict on service channel
14:	Handshake timeout

15:	Busy - timeout
22:	Transmitted date too short
23:	Transmitted date too long
24:	Element cannot be changed
25:	Element can currently not be changed (incorrect startup phase)
26:	Date smaller than minimum value
27:	Date larger than maximum value
28:	Operating date invalid (e.g. bit combination)
29:	Sercos ring currently not available (startup)
30:	Incorrect attribute received
31:	System axis invalid or it does not exist as Sercos axis (dummy axis)
33:	Data length exceeds size of service container
40:	Max number of simultaneously active commands exceeded
41:	Waiting time of the command change bit elapsed
50:	Conversion error (conversion to Sercos representation or vice versa)
109	Element is password-protected
110	Element is transferred cyclically
112	Element cannot be changed
114	Element can currently not be changed (PM)
115	Element can currently not be changed (OM)
117	Command cannot be interrupted
118	Command cannot be started
119	Command cannot be executed
500:	System axis not available as drive
501:	Configuration error (system axis number and axis number in ring do not match)
503:	Invalid Sercos phase for the requested function

12.11 Editing Tool Lists (MT_P_DbRecList)

12.11.1 Function

☞ Only together with "old" tool database tables that are not XML-based.

The "MT_DbRecList" function block has to be used for XML-based tables.

This program function block allows to request or delete a list of tool data blocks.

- **Searching for tool data blocks**

  To search for tool data blocks, up to seven search criteria - each linked by logic AND operators - can be used. For unused search criteria, enter 0 in "Field".

- **Deleting tool data blocks**

  A list of data blocks to be deleted can be specified. This function block deletes each of these data blocks by overwriting their contents with "0" except for sector and location numbers. The tool list requested with certain search criteria can also be used for deletion. The number of data blocks to be deleted is applied to **NofElem**.

The function block can be called once or multiple times in a PLC cycle. Calls can be absolute or conditional.

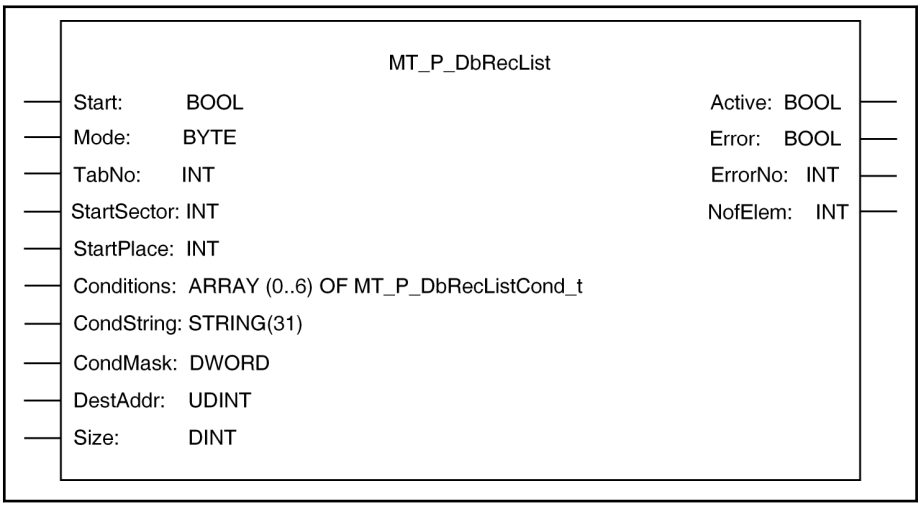


Fig. 12-12: Calling MT_DbRecList function block

Function block parameters

The function block is called with the following parameters:

Start:	Flag for new request
Mode:	1 searches tool data blocks from start of table 2: Search for tool data blocks starting from StartSector/StartPlace 3: Delete content of the tool data blocks acc. to the list
TabNo:	Table number
StartSector:	Sector number from which data blocks are searched for (only for mode 2)
StartPlace:	Place number from which data blocks are searched for (only for mode 2)

Conditions: Up to seven search conditions as array with the following structure:

Type MT_P_DbRecListCond_t

Struct

Field: INT; (* Comparison element for the search:

0 - Search condition is not taken into account

1 - Sector

2 - Query_int1

3 - Query_int2

4 - Query_int3

5 - Query_int4

6 - Query_string

7 - Query_bitfield *)

Value: INT; (* comparison element for the search at Field = 1 - 5 *)

End_struct

End_type

Cond-String: Comparison string for the search at field = 6 Query_string

CondMask: Bit mask only for Field = 7 Query_bitfield

DestAddr: Address of data as array with the following structure:

Type MT_P_DbRecListEntry

Struct

Sector: INT;

Place: INT;

End_struct

End_type

Size: Memory size for the data in bytes

The return parameters are:

Active: Flag that displays whether the function is still active

Error: Error detection: FALSE means that the function has been processed without errors

ErrorNo: Error code: only valid if Error = TRUE

NofElem: Number of returned elements

The function starts when **Start** is set and **Active** is not set. If "Start" is not reset using the PLC sequential program, it is reset when the function is completed.

Active is set until the function is completed. As soon as data is returned, the function is completed and Active is reset.

Search term:

The following values can be specified to select a search term:

0	Ignore search term	
1	Sector	

2	Tool number	(query_int 1)
3	Duplo number	(query_int 2)
4		(query_int 3)
5		(query_int 4)
6	Tool name	(query_string)
7	Tool ID	(bit field)

Tab. 12-1: Selecting search term

12.11.2 Configuration example

The list of all tools with the tool number 123 in the magazine (sector 2) and whose tool life expired is to be searched.

The task is triggered with a rising edge at the %IX40.0 input.

The MT_P_DbRecList function block could be called as follows:

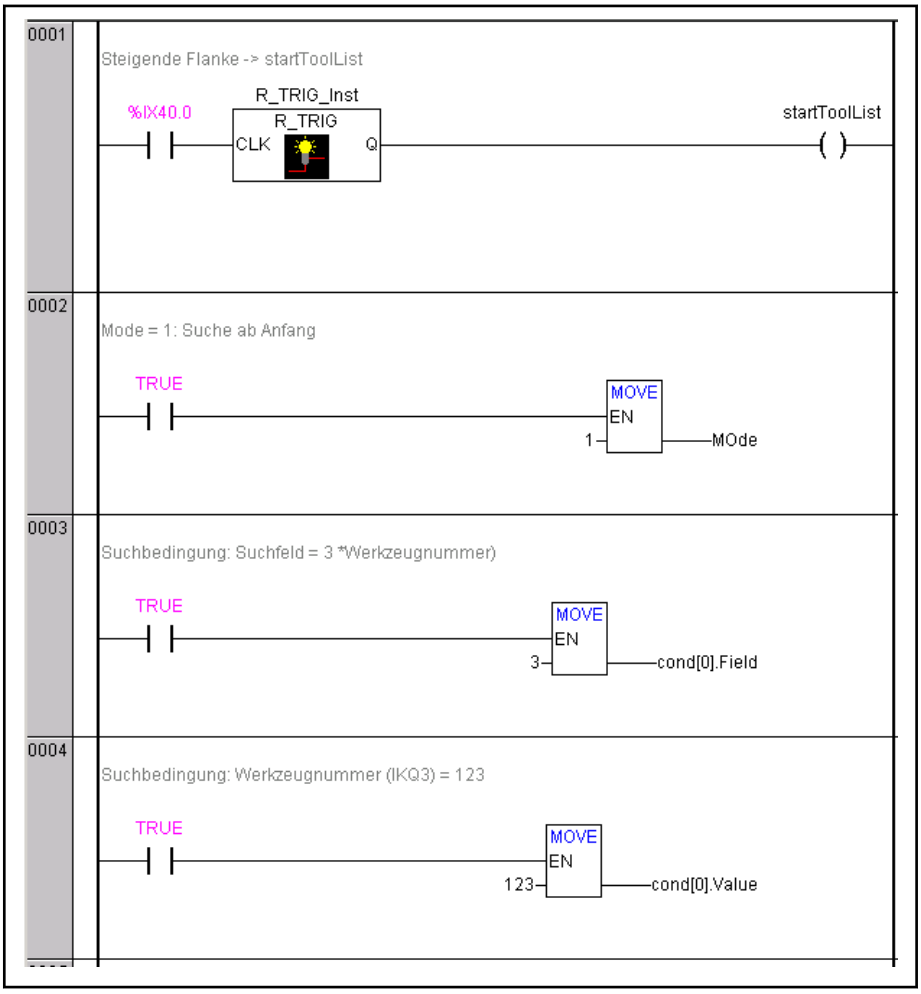


Fig. 12-13: Calling MT_P_DbRecList function block

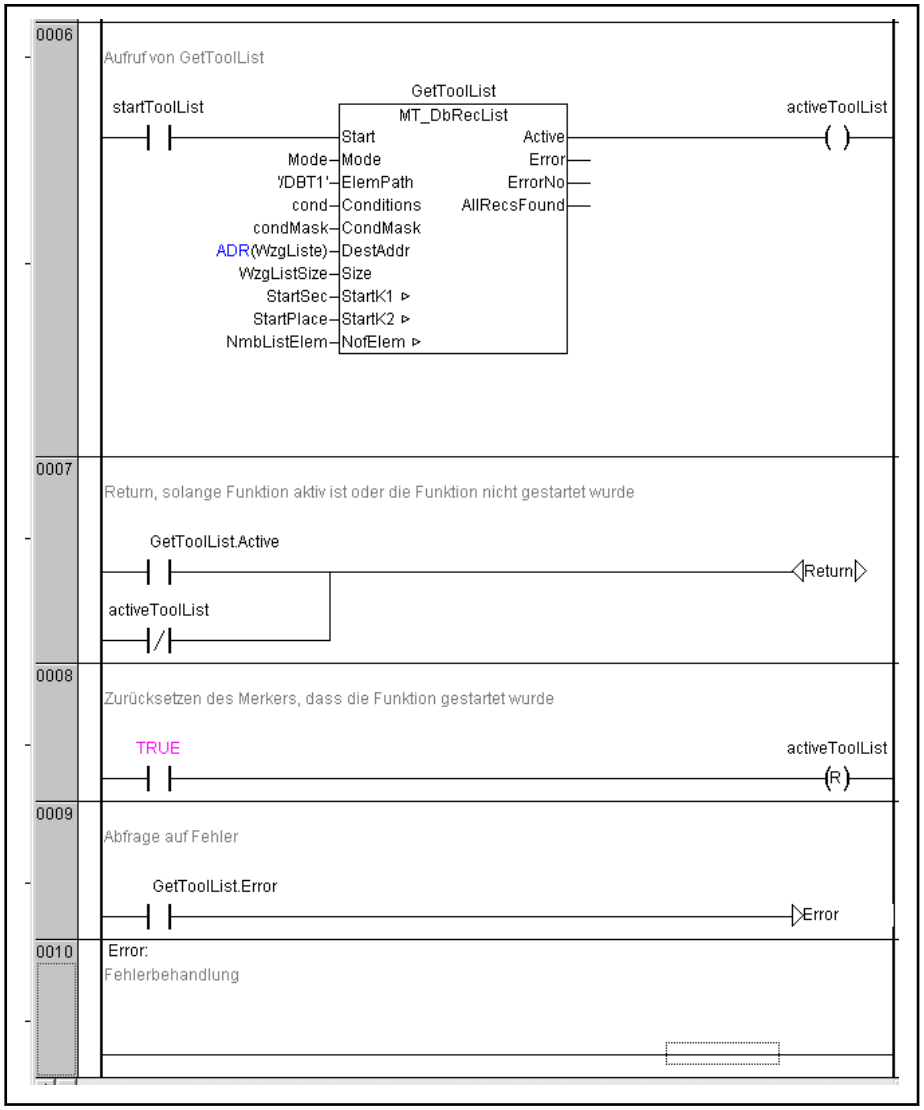


Fig. 12-14: Calling MT_P_DbRecList function block

12.11.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-3:	Data overflow: Only the number of data the memory can save is returned
-1:	General parameter error

General errors and states as acknowledgement after executing the function:

68:	Tool table is empty
1002:	Invalid table number

12.12 Editing data blocks (MT_P_DbData)

12.12.1 Function

☞ Only together with "old" tool database tables that are not XML-based.

The "MT_DbData" function block has to be used for XML-based tables.

This program function block provides various functions for reading and changing tool data.

The function block can be called once or multiple times in a PLC cycle. Calls can be absolute or conditional.

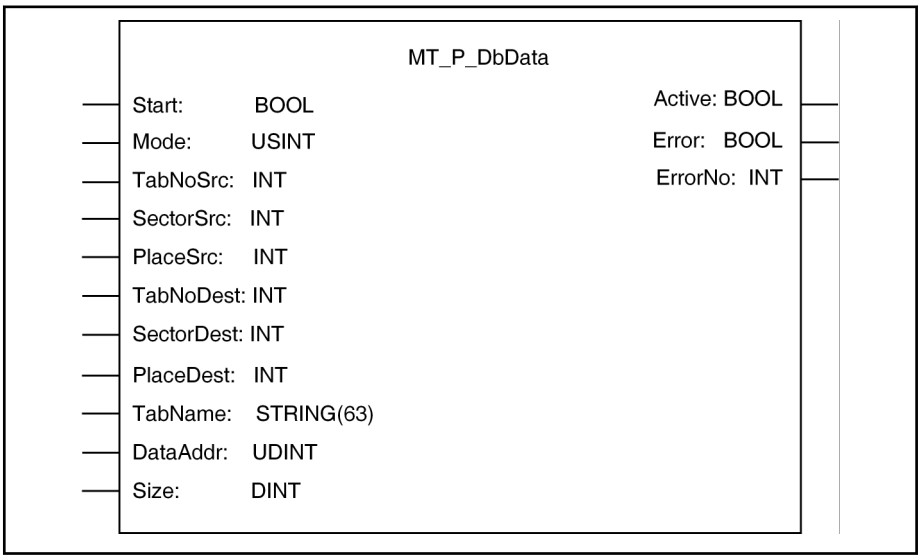


Fig. 12-15: Calling MT_DbData function block

Function block parameters

The function block is called with the following parameters:

Start:	Flag for new request
Mode:	1 Read tool data
	2 Read tool data with access lock for other processes
	3 Write tool data
	4 Enable tool data block
	5 Exchange tool data block
	6 Copy tool data block
	7 Delete tool data block
	8 Read in an ASCII file of the tool data
	9 Generate an ASCII file of the tool data
TabNoSrc:	Table number (source)
SectorSrc:	Sector number (source)
PlaceSrc:	Place number (source)
TabNoDest:	Table number (target) [relevant only for the modes 5 and 6]

SectorDest:	Sector number (target) [relevant only for the modes 5 and 6]
PlaceDest:	Place number (target) [relevant only for the modes 5 and 6]
TabName:	Table name [relevant only for the modes 8 and 9]
DestAddr:	Address of data. The data is returned as type MT_P_DbData:

```
Type MT_P_DbData
  Struct
    Sector: INT
    Place: INT
    Query_int_1: INT
    Query_int_2: INT
    Query_int_3: INT
    Query_int_4: INT
    Query_bitfield: INT
    Query_int: ARRAY[1...40]OFINT
    Query_string: STRING(31)
  End_struct
End_type
```

Size:	Memory size for the data in bytes

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors
ErrorNo:	Error code: only valid if Error = TRUE

The function starts when **Start** is set and **Active** is not set. If "Start" is not reset using the PLC sequential program, it is reset when the function is completed.

Active is set until the function is completed. As soon as data is returned, the function is completed and Active is reset.

12.12.2 Configuration example

The tool data in place 5 of sector 2 of table 1 is to be read including the locking against further access.

The task is triggered with a rising edge at the %IX41.0 input.

The **MT_P_DbData** function block can be called as follows:

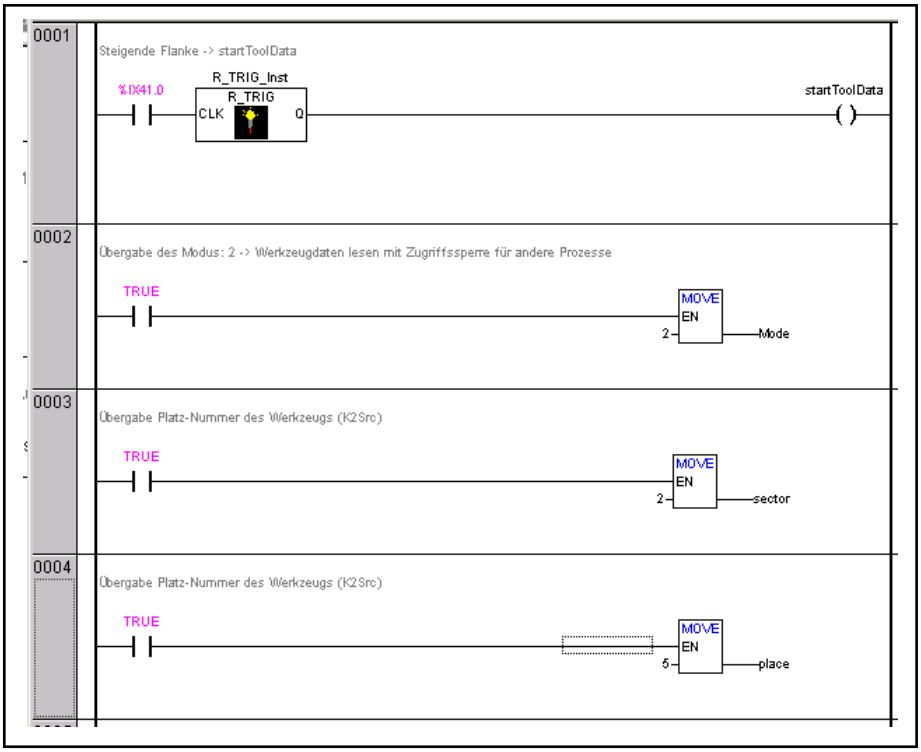


Fig. 12-16: Calling MT_P_DbData function block

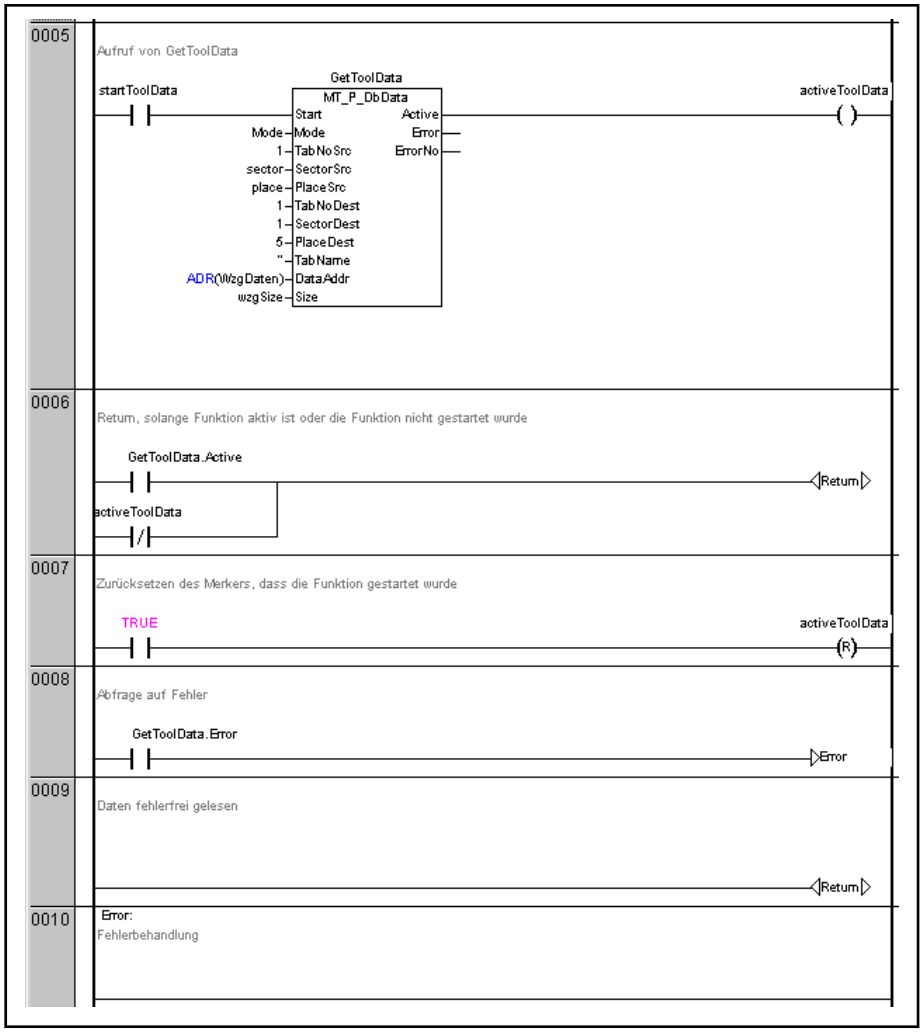


Fig. 12-17: Calling MT_P_DbData function block

12.12.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-3:	Data overflow: Only the number of data the memory can save is returned
-1:	General parameter error

General errors and states as acknowledgement after executing the function:

68:	Tool table is empty
72:	Data block not found
108:	The latest locked data block not yet released

115:	The data block was not locked before it was written back or enabled
1002:	Invalid table number

12.13 Editing data block lists (MT_DbRecList)

12.13.1 Function

This program function block allows to search for data blocks within a database table or to delete data blocks according to a list.

- **Searching for data blocks**

  To search for data blocks, up to 12 search criteria, each linked by logic AND operators, can be used. The search conditions taken into account are selected via the parameter **CondMask**.

- **Deleting data blocks**

  A list of data blocks to be deleted can be specified. This function block deletes each of these data blocks by overwriting their contents with "0", except for the sector and place numbers. The list requested using certain search criteria can also be used for deletion. The number of data blocks to be deleted is entered into **NofElem**.

If an error occurs, the index of the array element in which the error occurred is returned in **NofElem**.

The function block can be called once or multiple times in a PLC cycle. Calls can be absolute or conditional.

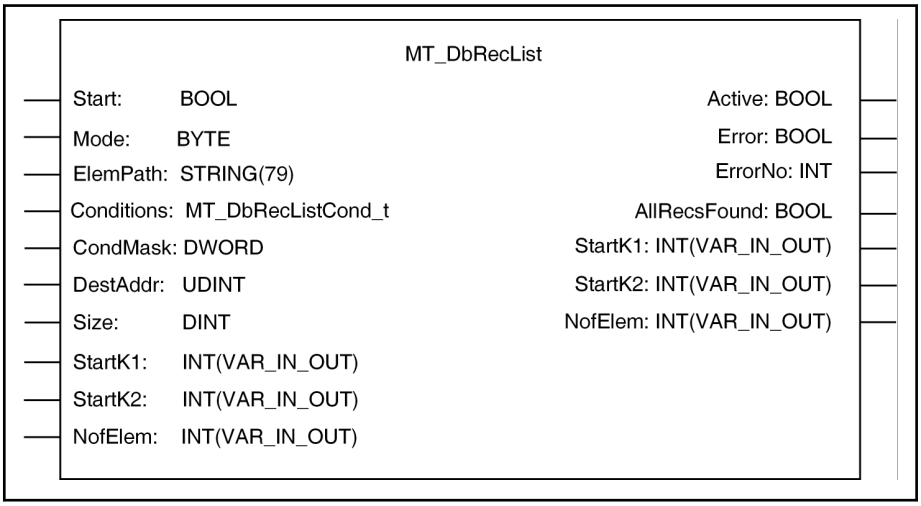


Fig. 12-18: Calling MT_DbRecList function block

Function block parameters

The function block is called with the following parameters:

Start:	Flag for new request
Mode:	1 Search data blocks from beginning of table.
	2 Search for data blocks starting at K1, K2.
	3 Delete content of the data blocks acc. to the list
ElemPath:	Identification of data including Path specifications (e.g. "/DBT1")

Conditions: Search conditions of type MT_DbRecListCond_t:

Type MT_DbRecList Cond_t

Struct

K1: INT; Key1

K2: INT; Key2

SKQ: INT; String

IKQ1: DINT; Integer value 1 (key value)

IKQ2: DINT; Integer value 2 (key value)

IKQ3: DINT; Integer value 3 (key value)

IQ1: DINT; Integer value 1 (no key value)

IQ2: DINT; Integer value 2 (no key value)

IQ3:DINT; Integer value 3 (no key value)

BQ1: DWORD; bit array 1

BQ1_Mask: DWORD; Mask for bit array 1

BQ2: DWORD; bit array 2

BQ2_Mask: DWORD; Mask for bit array 2

BQ3: DWORD; bit array 3

BQ3_Mask: DWORD; Mask for bit array 3

End_struct

End_type

CondMask: Mask to select the conditions to be used in the search:

16#00000001: K1

16#00000002: K2

16#00000004: SKQ

16#00000008: IKQ1

16#00000010: IKQ2

16#00000020: IKQ3

16#00000040: IQ1

16#00000080: IQ2

16#00000100: IQ3

16#00000200: BQ1

16#00000400: BQ2

16#00000800: BQ3

DestAddr: Address of data as array with the following structure:

```
Type MT_DBT1Hd
  Struct
    K1: INT
    K2: INT
    SKQ: STRING(31)
    IKQ1: DINT
    IKQ2: DINT
    IKQ3: DINT
    IQ1: DINT
    IQ2: DINT
    IQ3: DINT
    BQ1: DWORD
    BQ2: DWORD
    BQ3: DWORD
  End_struct
End_type
```

Size: Memory size for the data in bytes

StartK1: When searching with mode 2 (search starts at K1, K2), the value for K1, where the search is to be started, is transferred here

StartK2: When searching with mode 2 (search starts at K1, K2), the value for K2, where the search is to be started, is transferred here

☞ If not all the found elements can be returned during a search, since not enough memory was provided (AllRecsFound = FALSE), the values used to continue the search are returned in StartK1 and StartK2.

NofElem: Only for deletion:
Number of elements to be deleted

The return parameters are:

Active: Flag that displays whether the function is still active

Error: Error detection:
FALSE means that the function has been processed without errors

ErrorNo: Error code:
Only valid if Error = TRUE

NofElem: Only while searching:
Number of returned elements

AllRecs-Found: **TRUE:**

All data blocks corresponding to the search conditions are returned.

FALSE:

More data blocks exist Due to insufficient memory, these cannot be returned

StartK1 / StartK2: If not all the found elements can be returned during a search, since not enough memory was provided (AllRecsFound=FALSE), the values used to continue the search are returned in StartK1 and StartK2

NofElem: **Search:**

Number of returned elements.

Delete:

Only in case of error: Index of the data block in the list where the error occurred. When deleting data blocks, the list is processed starting at the end. Therefore, all the data blocks between the start and the faulty array element are not deleted if an error occurs

The function starts when **Start** is set and **Active** is not set. If "Start" is not reset using the PLC sequential program, it is reset when the function is completed.

Active is set until the function is completed. As soon as data is returned, the function is completed and Active is reset.

12.13.2 Configuration example

It is to be searched for the list of all tools within the tool table DBT1 with the Duplo number 123 that is located in the magazine (K1=2).

The task is triggered with a rising edge at the %IX40.0 input.

The **MT_DbRecList** function block could be called as follows:

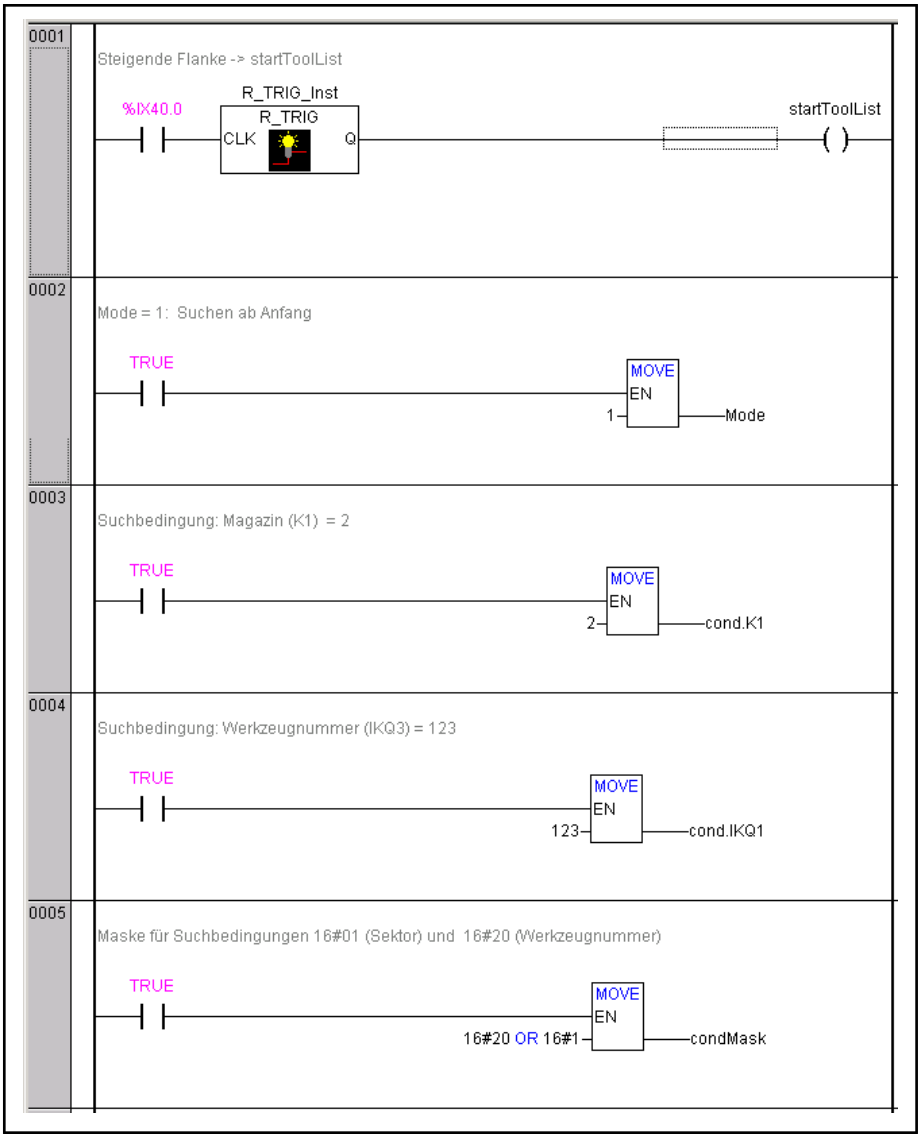


Fig. 12-19: Calling MT_DbRecList function block

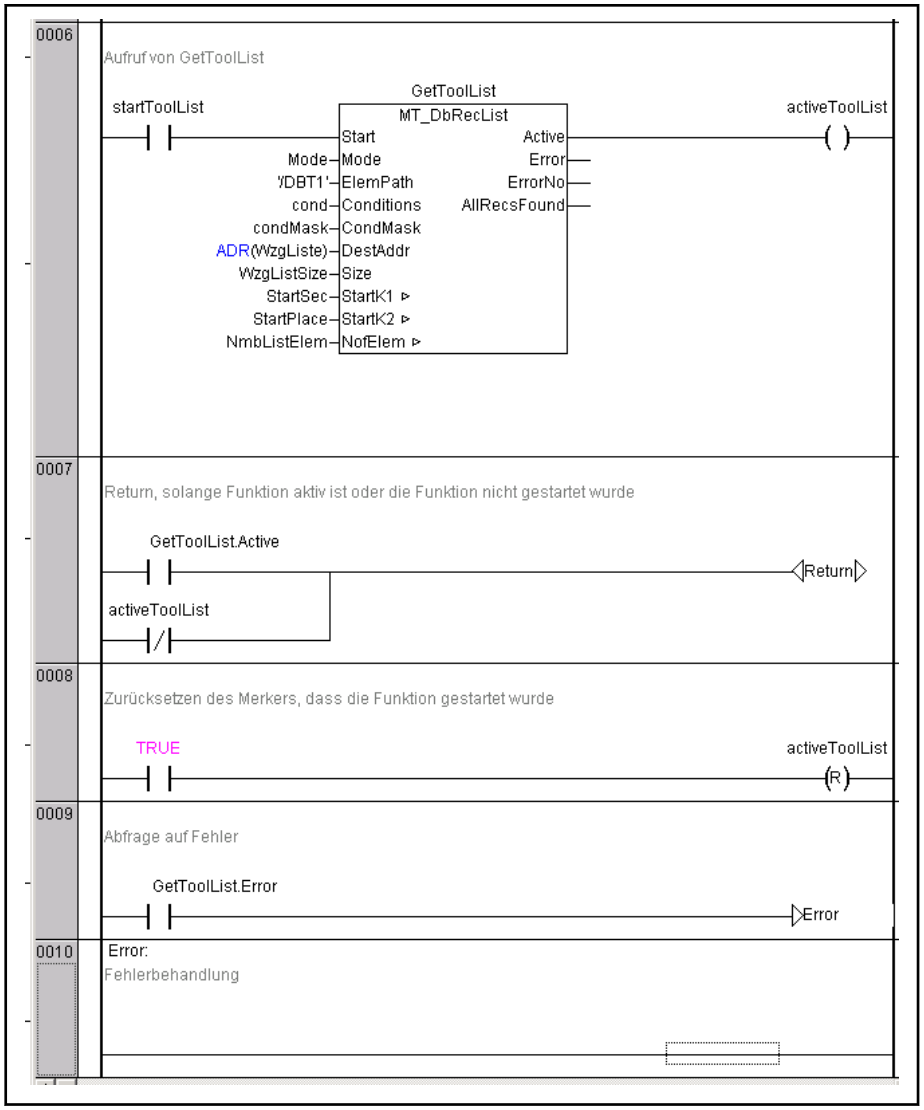


Fig. 12-20: Calling MT_DbRecList function block

12.13.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-3:	Data overflow: Only the number of data the memory can save is returned
-1:	General parameter error

General errors and states as acknowledgement after executing the function:

499:	Error when interpreting XML file
510:	Syntax error in ElemPath (incorrect name)
512:	Syntax error in ElemPath (incorrect element position)

516:	The element specified in ElemPath is not declared in the scheme
518:	Element not declared in scheme
519:	Transferred data size does not correspond to element size
520:	Transferred type ID does not fit to element (ElemPath)
521:	Unknown type ID
525:	Invalid REAL or LREAL value
1001:	Error when interpreting XML file
1002:	Database server not ready
1003:	The entered name too long
1007:	Transferred data range too small
1008:	Buffer for the path too small (events)
1102:	Invalid root element in ElemPath
1104:	Data block is already locked
1109:	Data block too big
1112:	Invalid search conditions
1204:	No data blocks with this key found
1205:	Database is empty
1206:	Last locked data block not released
1208:	No locked data block exists

12.14 Editing data blocks (MT_DbData)

12.14.1 Function

This program function block provides various functions to read and change data blocks (e.g. tool data).

Entire data blocks and partial data blocks can be processed. It is selected via the input variable **ElemPath** (e.g. /DBT1). The data is provided in the format defined on NC side by the scheme files. The respective structures in the PLC program have to be defined to access the files correctly. The corresponding types for the default configuration are available in the MTX library (see below).

When writing database data, the ID of the type by which the variable is defined on the NC side, has to be defined in **TypeId**. This is returned when variables of this type are read. In case of multiple write accesses, this ID has to be specified only once before the first access. It can then be transferred during each successive access.

The function block can be called once or multiple times in a PLC cycle. Calls can be absolute or conditional.

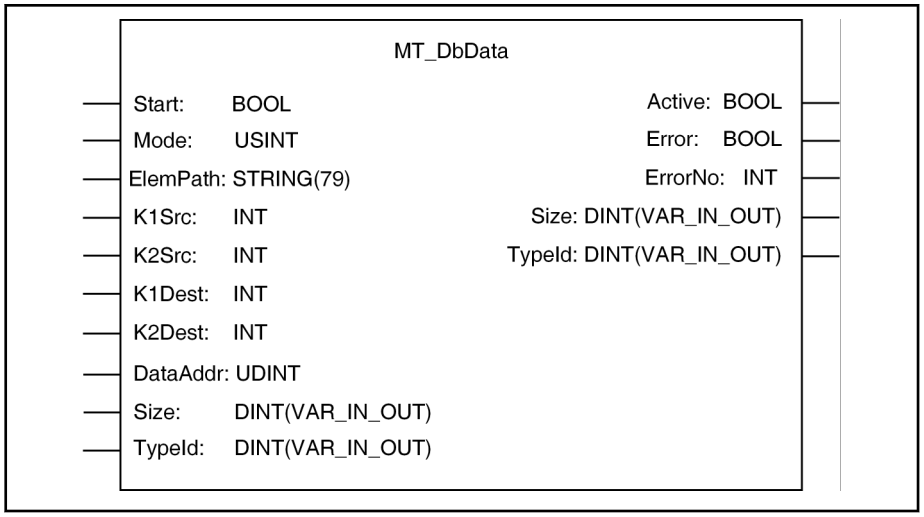


Fig. 12-21: Calling MT_DbData function block

Function block parameters

The function block is called with the following parameters:

Start: Flag for new request

Mode: DbData_Read (value 1) Read data block

DbData_ReadLock (value 2) Read data block with access lock for other processes

DbData_Write (value 3) Write data block

DbData_Unlock (value 4) Enable locked data block

DbData_Move (value 5) Move data block

DbData_Copy (value 6) Copy data block

DbData_Delete (value 7) Delete content of data block

DbData_WriteWithoutLock (value 11) Write data block without locking it before

DbData_Interchange (Value 12) Interchange data block

In the function codes "DbData_WriteWithoutLock" and "DbData_Write", the following options can be transferred additionally by setting single bits via the input parameter "Mode". These additional bits are added via OR operation:

- Level of checking the transmission data in the server (if no bit is set, the minimum check is executed in which only variables of type float, double and string are checked for the valid format):
- Mode = ModeORDbData_DisableBasicCheck

  No check

  Mode = ModeORDbData_EnableFullCheck

  Complete data check
- Transferring only parts of the data block:

  Mode = ModeORDbData_WrOnlyToolspecData

  Only the tool-specific parts of data block are applied

  Mode = ModeORDbData_WrOnlyPlacespecData

  Only the place-specific parts of data block are applied

ElemPath: Identification of data including Path specifications (e.g. "/DBT1")

K1Src: Key 1 (source)

K2Src: Key 2 (source)

K1Dest: Key 1 (target), relevant only for modes 5, 6 and 12

K2Dest: Key 2 (target) [relevant only for the modes 5, 6 and 12]

DataAddr: Address of data.

To access data blocks of the default configuration of the tool table, a data block is provided with the following structure:

```
Type MT_DBT1Rec_t
  Struct
    Hd: MT_DBT1Hd_t
    Ud: MT_DBT1Ud_t
  End_struct
End_type
Type MT_DBT1Ud_t
  Struct
    Tl: MT_Tl_t
    Ed: ARRAY[1...2]OFMT_Ed_t
  End_struct
End_type
TypeMT_Tl_t
  Struct
    Descr: STRING(31)
    LifeU: STRING(3)
    RemTlLife: REAL
  End_struct
End_type
TypeMT_Ed_t
  Struct
    Geo: MT_Geo_t
    Wear: MT_Wear_t
    Life: MT_EdLife_t
  End_struct
End_type
```

```
TypeMT_Geo_t
  Struct
    L1: LREAL
    L2: LREAL
    L3: LREAL
    Rad: LREAL
    Ori: DINT
  End_struct
End_type
TypeMT_Wear_t
  Struct
    L1: LREAL
    L2: LREAL
    L3: LREAL
    Rad: LREAL
  End_struct
End_type
TypeMT_EdLife_t
  Struct
    MonOn: BOOL
    RemLife: REAL
    WarnLim: REAL
    MaxLife: REAL
  End_struct
End_type
```

Size: **Read:**

Memory size for the data in bytes

Write:

Size of the data to be written (in bytes). The transferred size has to match the size of the data block to be written

TypeId: **Write:**

TypeId of the data block

(returned when reading a data block)

If the type of writable variables is a basic type, the TypeId can also be specified as constant of MT_basicTypeId. Otherwise, the TypeId - returned when reading a variable of this type - has to be specified

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors
ErrorNo:	Error code: only valid if Error = TRUE

Size: **Read:**

Size of the read tool data (in bytes) or, in case of error, the required memory size

Write:

Size of the written tool data (in bytes) or, in case of error, the correct size of the data block

TypeId: **Read:**

TypeId of the data block

The function starts when **Start** is set and **Active** is not set. If "Start" is not reset using the PLC sequential program, it is reset when the function is completed.

Active is set until the function is completed. As soon as data is returned, the function is completed and Active is reset.

12.14.2 Configuration example

The data of the tool in place 5 (Key2) of sector 2 (Key1) of tool database DBT1 are to be read including the locking against further access.

The task is triggered with a rising edge at the %IX41.0 input.

The **MT_DbData** function block can be executed as follows:

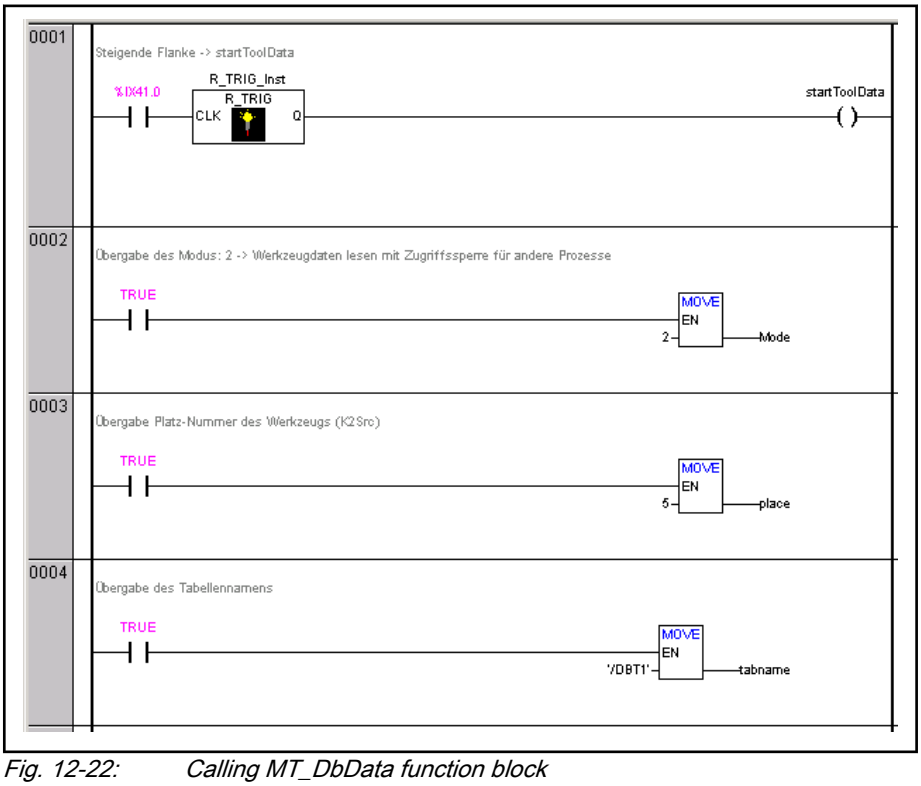


Fig. 12-22: Calling MT_DbData function block

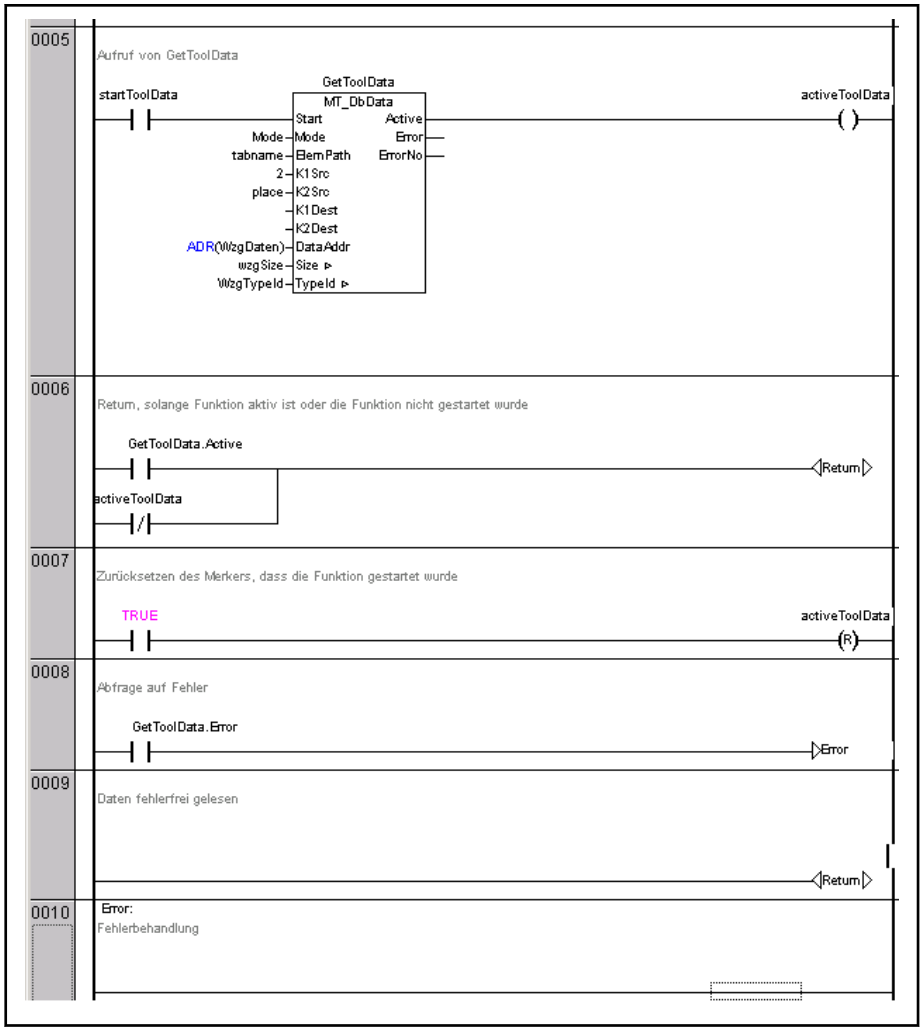


Fig. 12-23: Calling MT_DbData function block

12.14.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-3:	Data overflow: Only the number of data the memory can save is returned
-1:	General parameter error

General errors and states as acknowledgement after executing the function:

499:	Error when interpreting XML file
506:	There are more elements than permitted according to maxOccurs
507:	The content of the element does not correspond to the basic data type according to the scheme

508:	The content of the element does not corresponds to the facet according to the scheme
510:	Syntax error in ElemPath (incorrect name)
512:	Syntax error in ElemPath (incorrect element position)
516:	The element specified in ElemPath is not declared in the scheme
518:	Element not declared in scheme
519:	Transferred data size does not correspond to element size
520:	Transferred type ID does not fit to element (ElemPath)
521:	Unknown type ID
525:	Invalid REAL or LREAL value
1001:	Error when interpreting XML file
1002:	Database server not ready
1003:	The entered name too long
1007:	Transferred data range too small
1008:	Buffer for the path too small (events)
1102:	Invalid root element in ElemPath
1104:	Data block is already locked
1109:	Data block too big
1112:	Invalid search conditions
1204:	No data blocks with this key found
1205:	Database is empty
1206:	Last locked data block not released
1208:	No locked data block exists

12.15 Backing up the database table (MT_DbSave)

12.15.1 Function

This program function block can be used to partially or totally back up a database table in a file. The following options exist:

- The file is deleted and then the data blocks are backed up (mode =1)

  - or -

- The data blocks are added to the end of the file (mode = 2).

The parameter **SelMask** can be specified in connection with K1Sel and K2Sel to back up only certain data blocks.

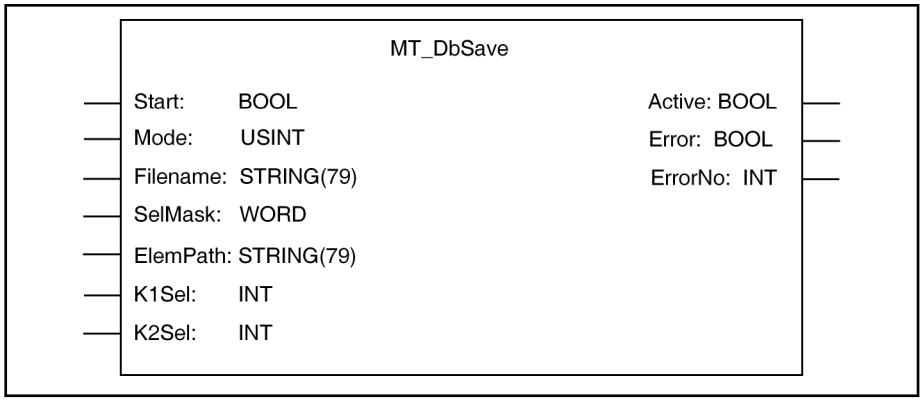


Fig. 12-24: Back up database table

Function block parameters

The function block is called with the following parameters:

Start:	Flag for new request
Mode:	1: Back up database table in an XML file after deleting the file content 2: Back up database table in an XML file by attaching it to the file contents
ElemPath:	Identification of data including Path specifications (e.g. "/DBT1")
SelMask:	Specifications to backup selected data blocks: 16#00: No selected data block; K1Sel and K2Sel are not considered 16#01: Back up all data blocks where K1 corresponds to the specification of K1Sel 16#02: Back up all data blocks where K1 and K2 correspond to the specification of K1Sel and K2Sel
K1Sel:	Select data blocks via Key1
K2Sel:	Select data blocks via Key2
Filename:	Name of backup file

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors
ErrorNo:	Error code: Only valid if Error = TRUE

The function starts when **Start** is set and **Active** is not set. If "Start" is not reset using the PLC sequential program, it is reset when the function is completed.

Active is set until the function is completed. As soon as data is returned, the function is completed and Active is reset.

12.15.2 Configuration example

All data blocks of database table "\DBT1" are to be backed up in the file "tool-tab.dbf". The file content is to be deleted before (mode=1).

The task is triggered with a rising edge at the %IX41.0 input.

The **MT_DbSave** function block could be called as follows:

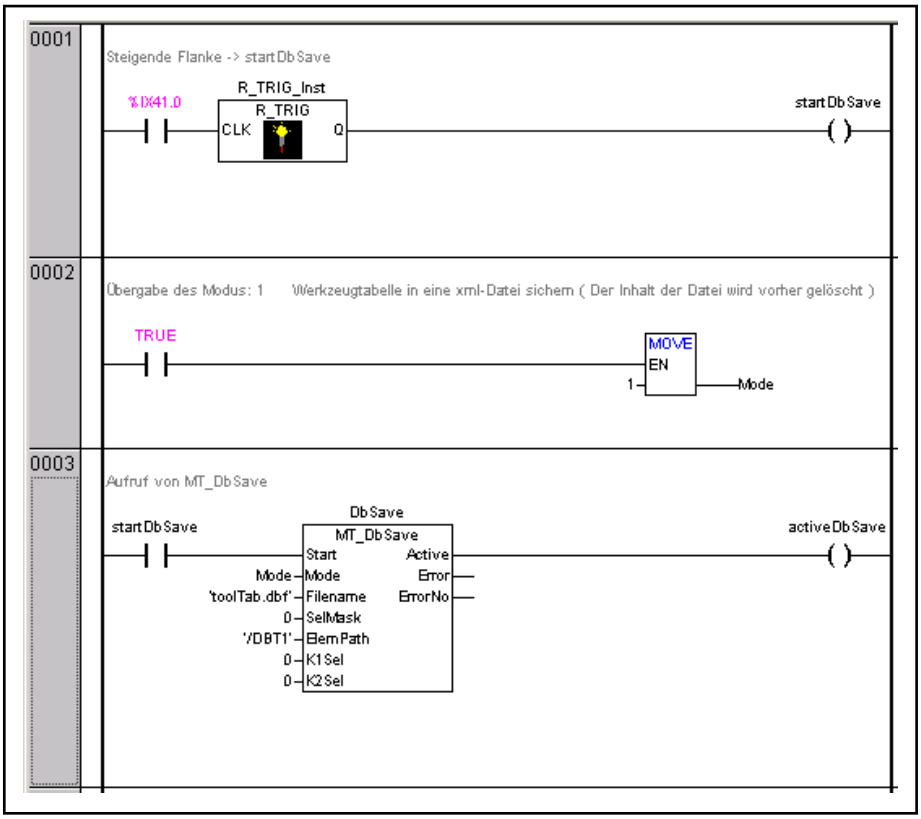


Fig. 12-25: Calling MT_DbSave function block

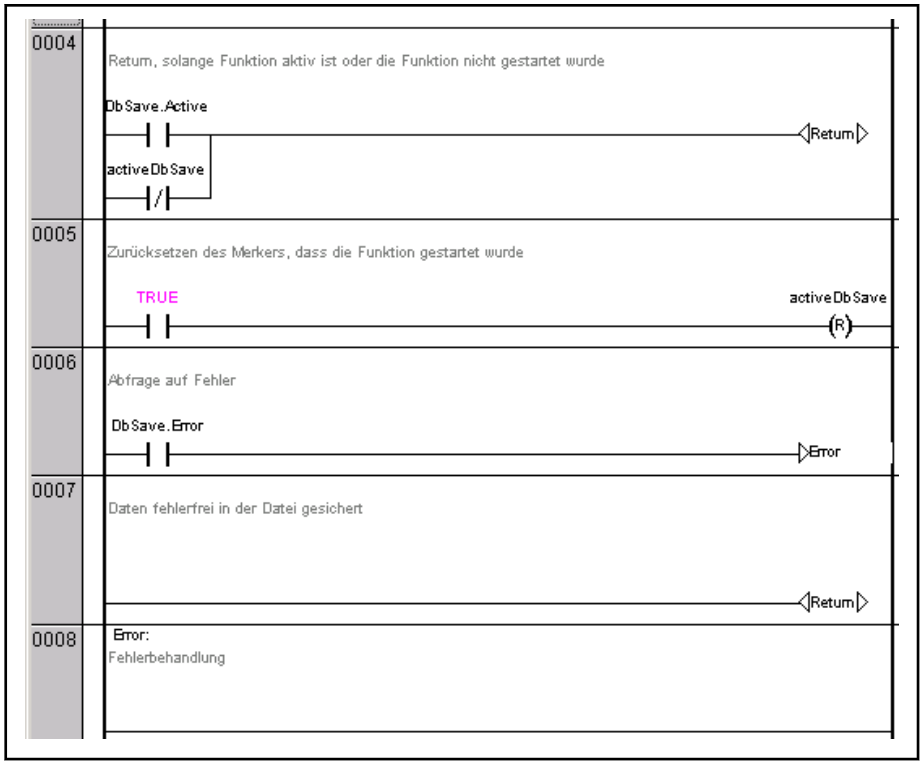


Fig. 12-26: Calling MT_DbSave function block

12.15.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-3:	Data overflow: Only the number of data the memory can save is returned
-1:	General parameter error

General errors and states as acknowledgement after executing the function:

499:	Error when interpreting XML file
510:	Syntax error in ElemPath (incorrect name)
512:	Syntax error in ElemPath (incorrect element position)
516:	The element specified in ElemPath is not declared in the scheme
518:	Element not declared in scheme
519:	Transferred data size does not correspond to element size
520:	Transferred type ID does not fit to element (ElemPath)
521:	Unknown type ID
1001:	Error when interpreting XML file
1002:	Database server not ready

1003:	The entered name too long
1007:	Transferred data range too small
1008:	Buffer for the path too small (events)
1102:	Invalid root element in ElemPath
1104:	Data block is already locked
1109:	Data block too big
1112:	Invalid search conditions
1204:	No data blocks with this key found
1205:	Database is empty
1206:	Last locked data block not released
1208:	No locked data block exists

12.16 Loading the database table (MT_DbLoad)

12.16.1 Function

This program function block can be used to partially or totally load a database table from an XML file.

It can be specified in connection with K1Sel and K2Sel to load only certain data blocks via the parameter **SelMask** .

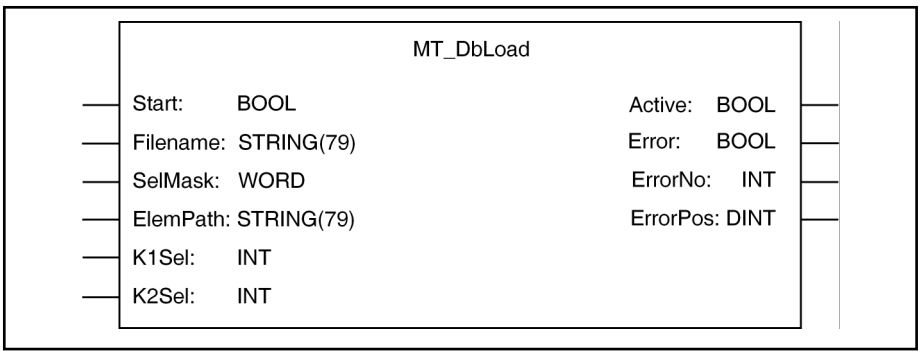


Fig. 12-27: Load database table (MT_DbLoad)

Function block parameters

The function block is called with the following parameters:

Start:	Flag for new request
ElemPath:	Identification of data including Path specifications (e.g. "/DBT1")
SelMask:	Specifications to load selected data blocks: 16#00: No data blocks selected; K1Sel and K2Sel are not used 16#01: Load all data blocks where K1 corresponds to the K1Sel specification 16#02: Load all data blocks where K1 and K2 correspond to the specification of K1Sel and K2Sel
K1Sel:	Select data blocks via Key1

K2Sel:	Select data blocks via Key2
Filename:	Name of the file to be loaded

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors
ErrorNo:	Error code: Only valid if Error = TRUE

The function starts when **Start** is set and **Active** is not set. If "Start" is not reset using the PLC sequential program, it is reset when the function is completed.

Active is set until the function is completed. As soon as data is returned, the function is completed and Active is reset.

12.16.2 Configuration example

All data blocks of database table "\DBT1" are to be loaded from the file "tooltab.dbf".

The task is triggered with a rising edge at the %IX41.0 input.

The **MT_DbLoad** function block can be called as follows:

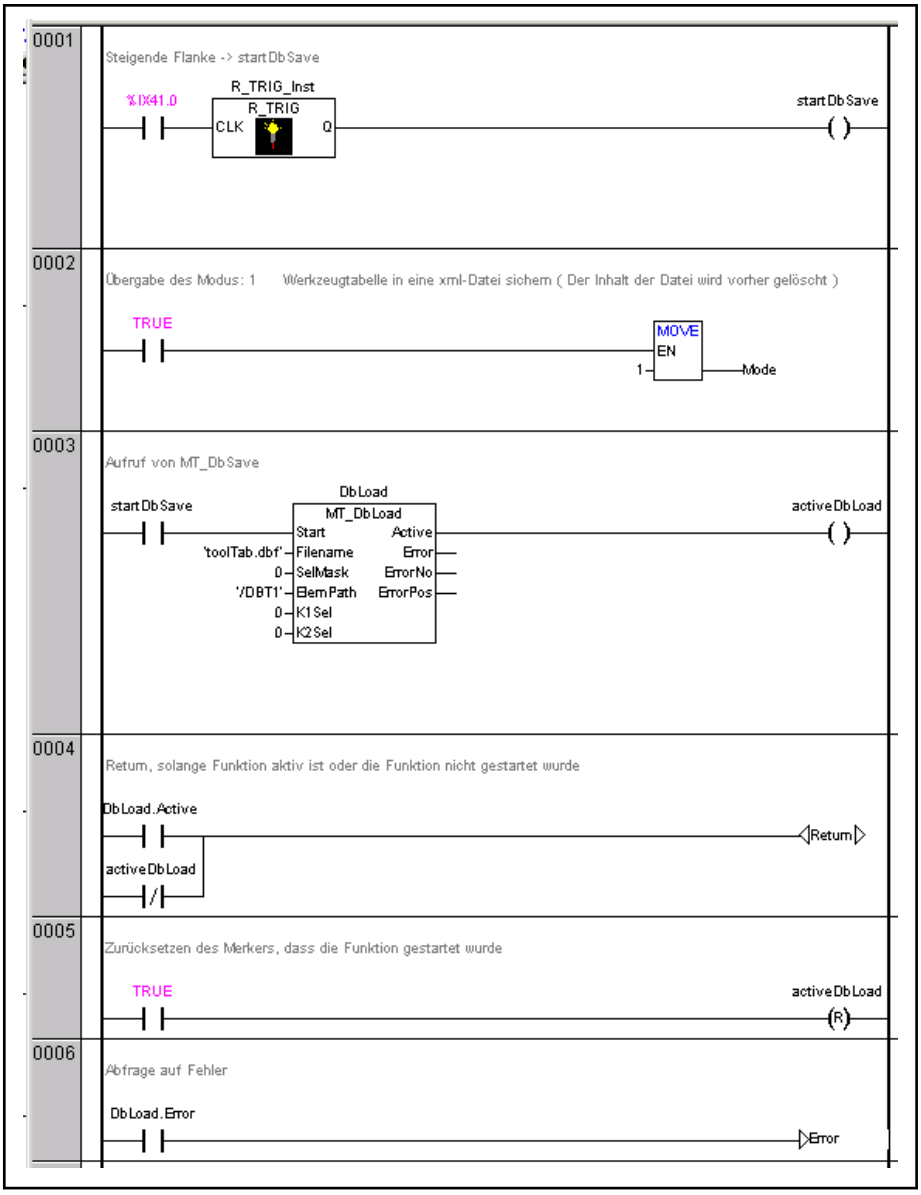


Fig. 12-28: Calling MT_DbLoad function block

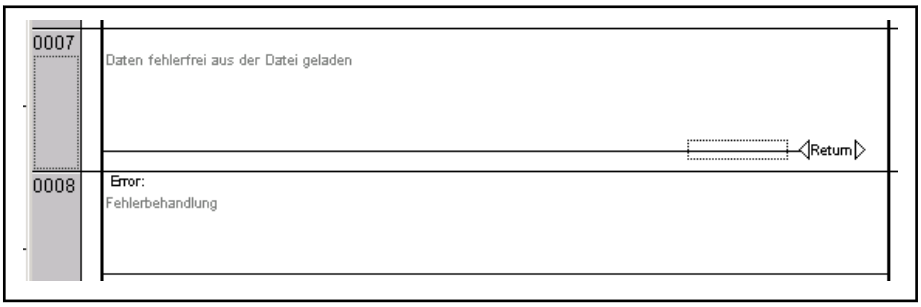


Fig. 12-29: Calling MT_DbLoad function block

12.16.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-3:	Data overflow: Only the number of data the memory can save is returned
-1:	General parameter error

General errors and states as acknowledgement after executing the function:

499:	Error when interpreting XML file
510:	Syntax error in ElemPath (incorrect name)
512:	Syntax error in ElemPath (incorrect element position)
516:	The element specified in ElemPath is not declared in the scheme
518:	Element not declared in scheme
519:	Transferred data size does not correspond to element size
520:	Transferred type ID does not fit to element (ElemPath)
521:	Unknown type ID
1001:	Error when interpreting XML file
1002:	Database server not ready
1003:	The entered name too long
1007:	Transferred data range too small
1008:	Buffer for the path too small (events)
1102:	Invalid root element in ElemPath
1104:	Data block is already locked
1109:	Data block too big
1112:	Invalid search conditions
1204:	No data blocks with this key found
1205:	Database is empty
1206:	Last locked data block not released
1208:	No locked data block exists

12.17 Tool Correction (MT_TCorr)

12.17.1 Function

This program function block allows to set correction values. Up to 16 corrections can be transferred with one call.

The external tool corrections are activated using the G-function "G47 ED-x" (x = 1 - 16) and deactivated using G48.

☞ The correction values are applied in the next program block to be prepared.

To apply the correction values immediately after their input, suspend the NC block preparation (e.g., by **WAIT** or **Block transfer inhibit**) or prepare the already prepared NC blocks again (by **Cancel distance to go**).

This function block can be called absolutely or conditionally.

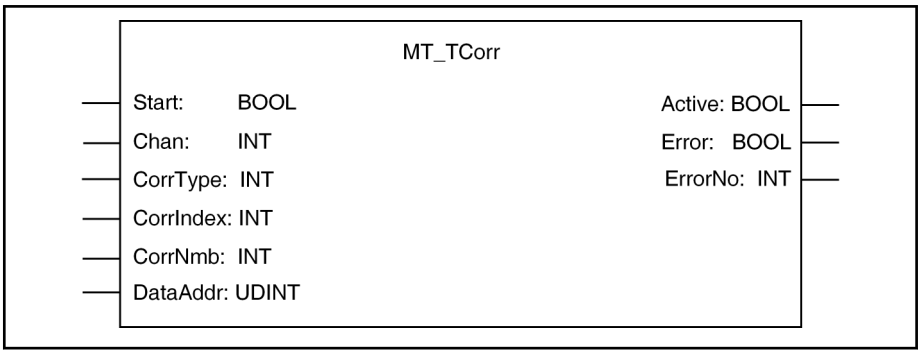


Fig. 12-30: Tool correction (MT_TCorr)

Function block parameters

The function block is called with the following parameters:

Start:	Flag for new request
Chan:	Number of the channel in which the correction values apply
CorrType:	1: Correction with 1 radius and 3 length correction values each and the edge position. 2: Correction with 1 radius and 3 length correction values each, the edge position and angles phi, theta and psi
CorrIndex:	Selection of correction: valid values 1 - 16
CorrNmb:	Number of corrections
DataAddr:	Address of data. Depending on the correction type, the correction values are transferred as array in the following format:

```
CorrType 1:
Type MT_Geo_t
  Struct
    Len1: LREAL
    Len2: LREAL
    Len3: LREAL
    Rad: LREAL
    Ori: DINT
  End_struct
End_type
CorrType 2:
Type MT_GeoExtended_t
  Struct
    Len1: LREAL
    Len2: LREAL
    Len3: LREAL
    Rad: LREAL
    Ori: DINT
    Phi: LREAL
    The: LREAL
    Psi: LREAL
  End_struct
End_type
```

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors
ErrorNo:	Error code: Only valid if Error = TRUE

The function starts when **Start** is set and **Active** is not set. The correction values are transferred to the NC immediately.

If "Start" is not reset using the PLC sequential program, it is reset when the function is completed.

Active is set until the NC has acknowledged the correction specification.

12.17.2 Configuration example

A radius correction of 3.1234 mm and a length correction of 124 mm is to be specified as second correction (G47 ED-2) in channel 1 for a milling tool (correction type 2).

The task is triggered with a rising edge at the %IX42.0 input.

The **MT_TCorr** function block can be called as follows:

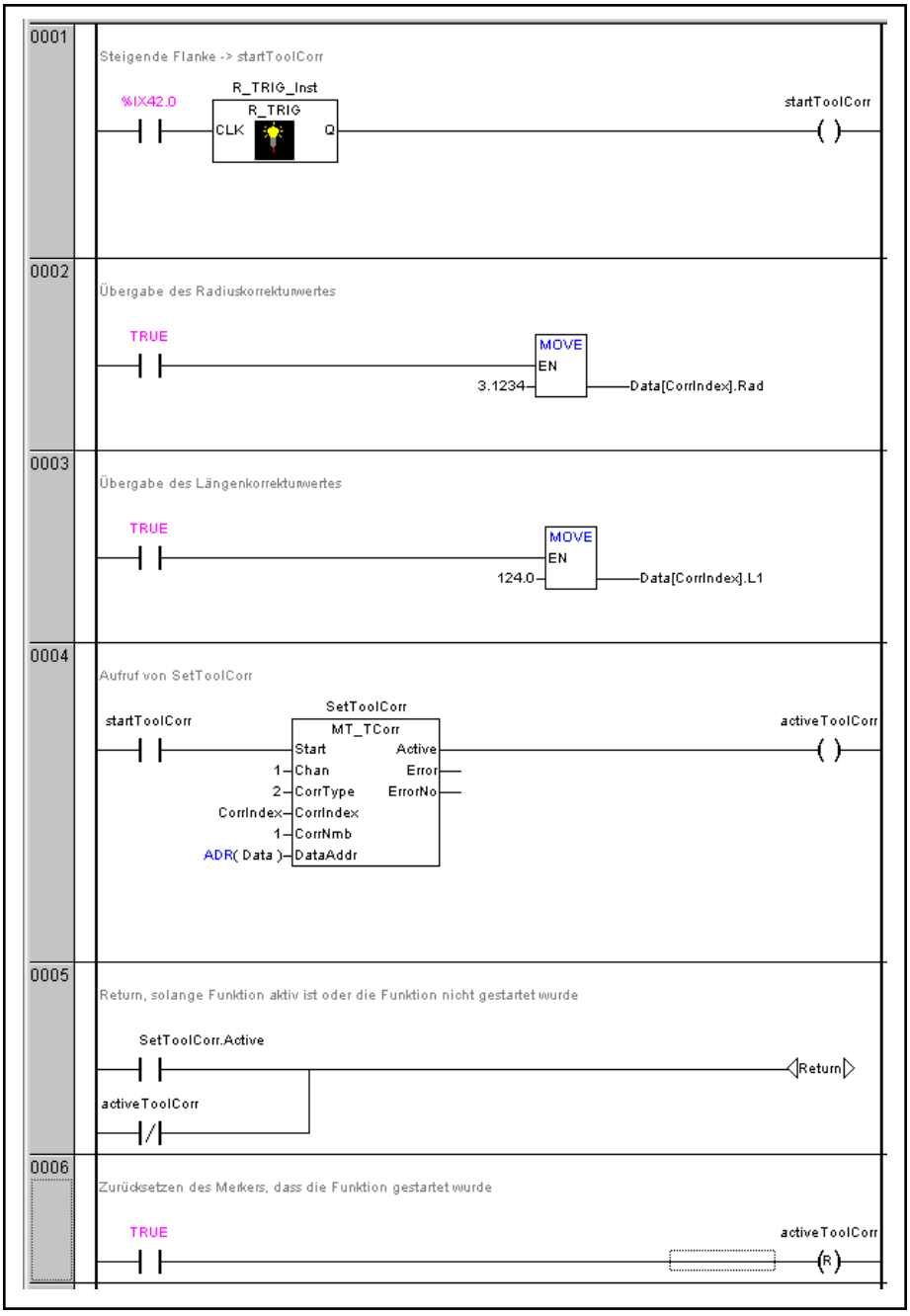


Fig. 12-31: Calling MT_TCorr function block

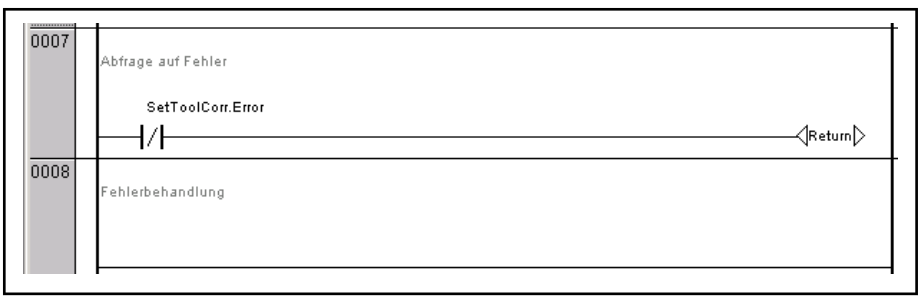


Fig. 12-32: Calling MT_TCorr function block

12.17.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-3:	Data overflow: Only the number of data the memory can save is returned
-1:	General parameter error

General errors and states as acknowledgement after executing the function:

1:	Invalid channel number
19:	Invalid correction index
20:	Max number of corrections exceeded (possibly displayed with correction index)
21:	Invalid type of correction
22:	Invalid tool edge position
36:	Correction value cannot be read, initialized or is not reasonable (e.g. 10^{100}).

12.18 Reading/writing permanent CPL variables (MT_CplData)

12.18.1 Function

This program function block provides the valid value at the time of query of a permanent CPL variable or writes to this CPL variable.

If arrays are edited, the number of the elements to be transferred is limited to 100. If more elements have to be transferred, multiple requirements with the corresponding offsets have to be made.

This function block can be called absolutely or conditionally.

The function block does not convert any types. Source and target data type have to match.

To access CPL variables with type conversion, the MT_CplDataConv function block is available.

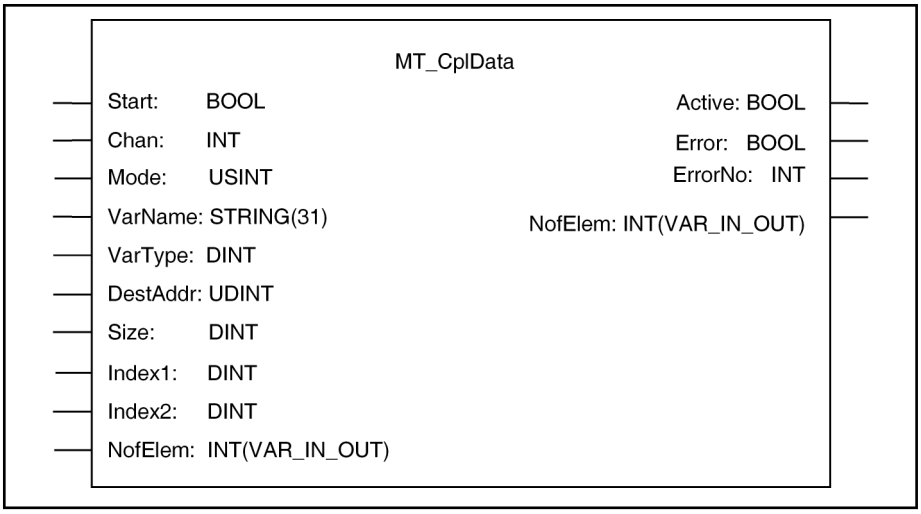


Fig. 12-33: Calling MT_CplData function block

Function block parameters

The function block is called with the following parameters:

Start:	Flag for new request
Chan:	Desired channel number
Mode:	0: Read CPL value 1: Write CPL value
VarName:	Name of CPL variable
VarType:	Type definition of CPL variable: 0 int 1 float 2 double 3 boolean 4 char
DestAddr:	Address of data
Size:	Memory size for the data in bytes
Index1:	Array index 1 (only for arrays)
Index2:	Array index 2 (only for two-dimensional array)
NofElem:	**Read:** Number of requested values **Write:** Number of specified values

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors

ErrorNo: Error code:

Only valid if Error = TRUE

NofElem: **Read:**

Number of read values

Write:

Number of applied values

The function starts when **Start** is set and **Active** is not set. If "Start" is not reset using the PLC sequential program, it is reset when the function is completed.

Active is set until the function is completed. As soon as data is returned, the function is completed and Active is reset.

For the parameter "NofElem", note that it is an input as well as an output parameter: If "0" is specified as input parameter, no data is read or applied. The function block returns without error ID

12.18.2 Configuration example

CPL variable "@Tool location" of type integer is to be read out in channel 1.

The task is triggered with a rising edge at the %IX42.5 input.

The **MT_CplData** function block could be called as follows:

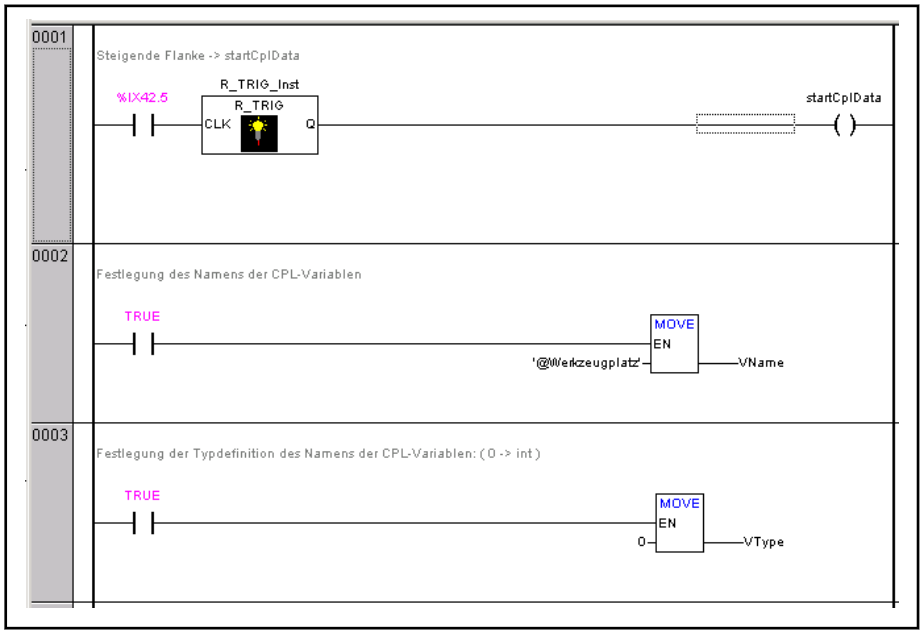


Fig. 12-34: Calling MT_CplData

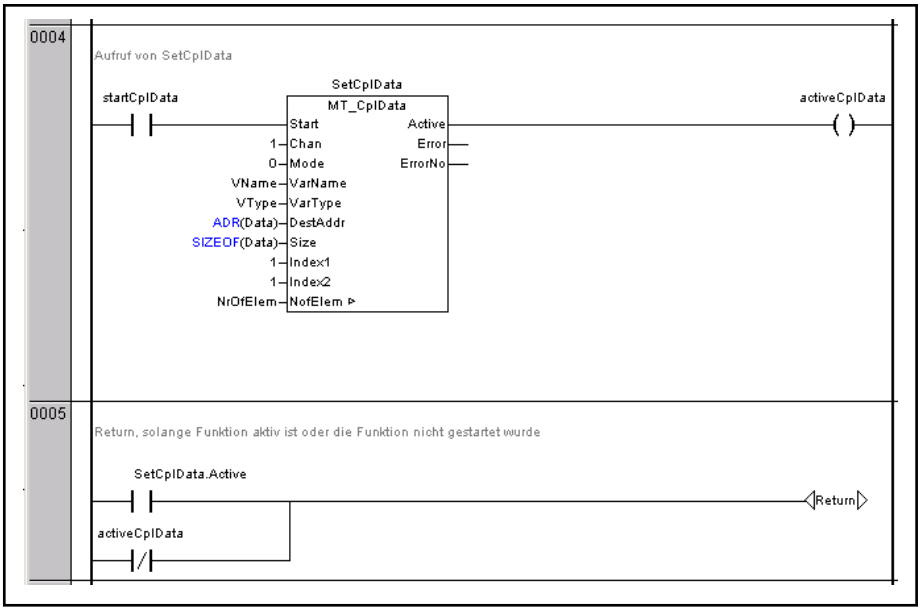


Fig. 12-35: Calling MT_CplData Graphic

12.18.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-3:	Data overflow: Only the number of data the memory can save is returned
-1:	General parameter error

General errors and states as acknowledgement after executing the function:

1:	Variable not found
2:	Different variable type
3:	Array index incorrect
4:	Variable type does not exist

12.19 Reading/writing permanent CPL variables with type conversion (MT_CplDataConv)

12.19.1 Function

This program function block provides the value of a permanent CPL variable valid at the time of the query or it can write to this CPL variable. CPL and PLC variable can be of different data type.

The function block converts between different data types. Any loss of information or value range overflow is not indicated.

This function block can be called absolutely or conditionally.

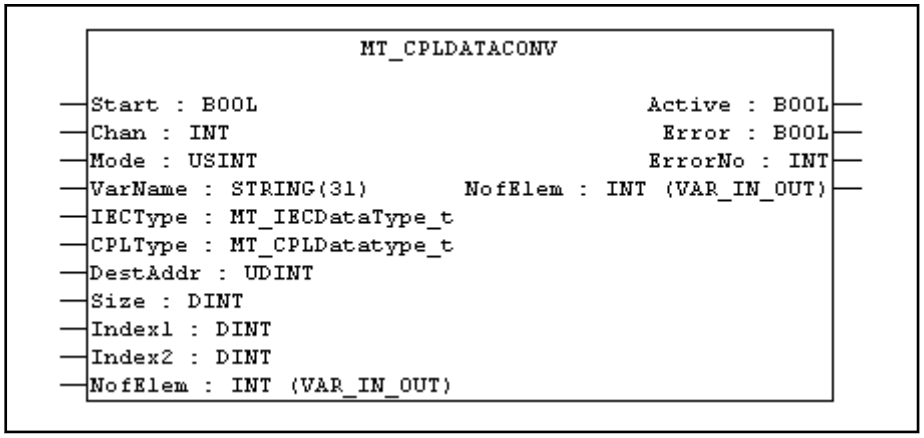


Fig. 12-36: Calling MT_CplDataConv function block

Function block parameters

The function block is called with the following parameters:

Start:	Flag for new request
Chan:	Desired channel number
Mode:	0: Read CPL value 1: Write CPL value
VarName:	Name of CPL variable
IECType:	Type definition of PLC variable: IEC_DINT (0) DINT IEC_REAL (1) REAL IEC_LREAL (2) LREAL IEC_BOOL (3) BOOL IEC_SINT (4) SINT IEC_INT (5) INT IEC_UDINT (6) UDINT IEC_DWORD (7) DWORD IEC_USINT (8) USINT IEC_BYTE (9) BYTE IEC_STRING (10) STRING IEC_UINT (11) UINT IEC_WORD (12) WORD
CPLType:	Type definition of CPL variable: CPL_INT (0) int CPL_REAL (1) float CPL_LREAL (2) double CPL_BOOL (3) Boolean CPL_CHAR (4) char
DestAddr:	Address of data
Size:	Memory size for the data in bytes

Index1:	Array index 1 (only for arrays)
Index2:	Array index 2 (only for two-dimensional array)
NofElem:	**Read:**
	Number of requested values
	Write:
	Number of specified values

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection:
	FALSE means that the function has been processed without errors
ErrorNo:	Error code:
	Only valid if Error = TRUE
NofElem:	**Read:**
	Number of read values
	Write:
	Number of applied values

The function starts when **Start** is set and **Active** is not set. If "Start" is not reset using the PLC sequential program, it is reset when the function is completed.

Active is set until the function is completed. As soon as data is returned, the function is completed and Active is reset.

For the parameter "NofElem", note that it is an input as well as an output parameter: If "0" is specified as input parameter, no data is read or applied. The function block returns without error ID

12.19.2 Configuration example

CPL variable "@Feed_X" of type integer is to be read out in channel 1. On the PLC side, the value is required in real format.

The task is triggered with a rising edge at the %IX42.5 input.

The **MT_CplDataConv** function block could be called as follows:

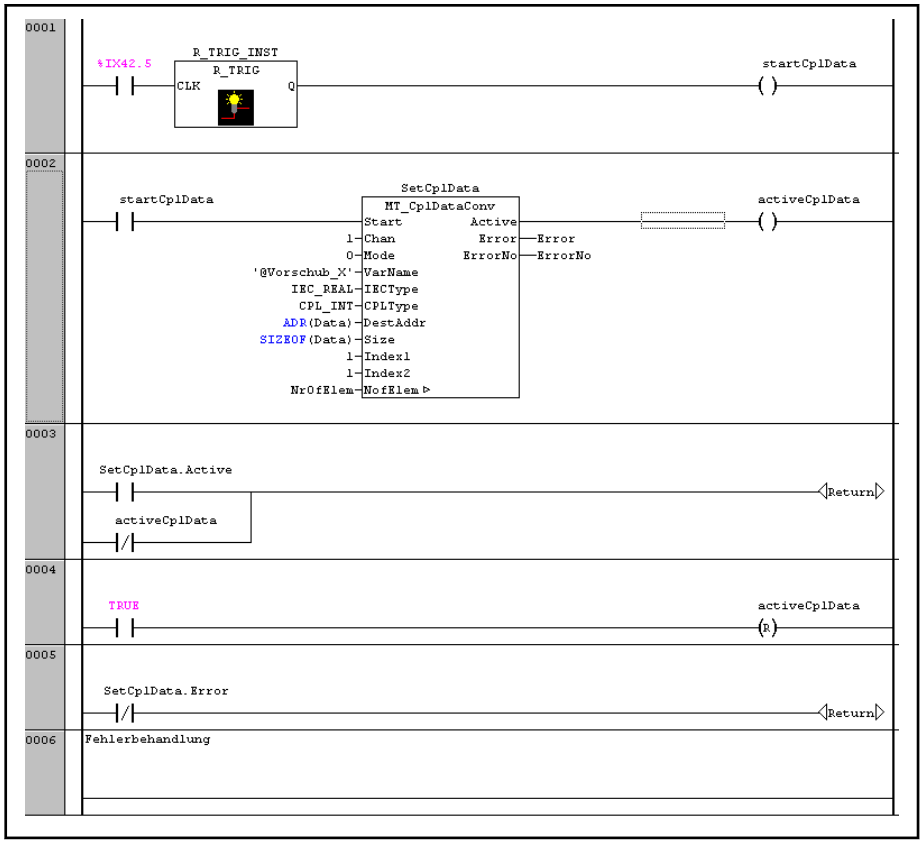


Fig. 12-37: Calling MT_CplDataConv

12.19.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task
-3:	Data overflow: Only the number of data the memory can save is returned
-1:	General parameter error

General errors and states as acknowledgement after executing the function:

1:	Variable not found
2:	Different variable type
3:	Array index incorrect
4:	Variable type does not exist

12.20 Reading System Data (MT_SD_RD)

12.20.1 Function

The **Read system data** function block allows to ready any system data. Entire as well as partial variables can be read. It is selected via the input variable

XPath (e.g. /VAR_1). The data is provided in the format defined on NC side by the scheme files. The respective structures in the PLC program have to be defined to access the files correctly.

Restrictions:

- No complete arrays can be read on the lowest level (e.g. /ArrayOfInts). On the other hand, complete arrays that are subelements of a system variable can be read (e.g. /ArrayOfInts).
- The size of the requested data is limited to 4 KB.

Several instances of the function block can be called in one PLC cycle.

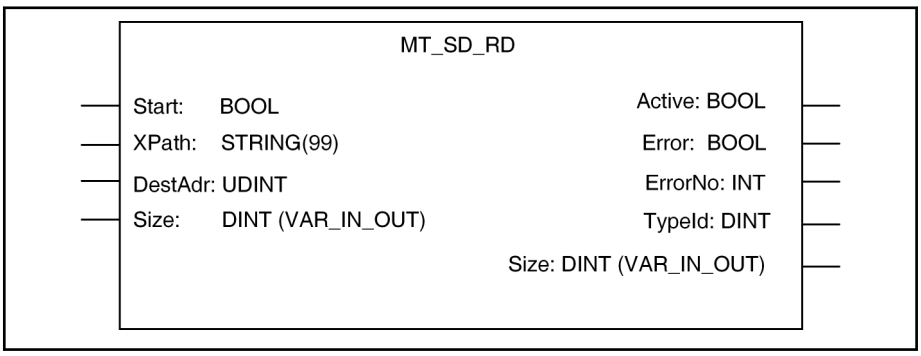


Fig. 12-38: Calling MT_SD_RD function block

Function block parameters

The function block is called with the following parameters:

Start:	Flag for new request A new request is executed if **Start** =TRUE and **Active** = FALSE
XPath:	Identification of system data including Path specifications (e.g. "/Var_1")
DataAdr:	Address of data The data has to be provided in the format they were defined on the NC side by the scheme files
Size:	Max. size of the data range for the requested data in bytes (SIZEOF(<variable>))

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors
ErrorNo:	Error code: Only valid if Error = TRUE
Size:	Size of the requested system data (in bytes)
TypeId:	Type ID of the system date This ID must be defined while writing system data

The function starts when the **Start** bit is set and "Active" is not set. If the "Start" bit is not reset using the PLC sequential program, it is reset when the function has been completed.

Active is set in the status until the function has been completed. As soon as data is returned, the function is completed and **Active** is reset.

12.20.2 Configuration example

The data of system date "/Var_1" is to be read.

The task is started with a rising edge at the **E43.0** input.

The **MT_SD_Rd** function block could be called as follows:

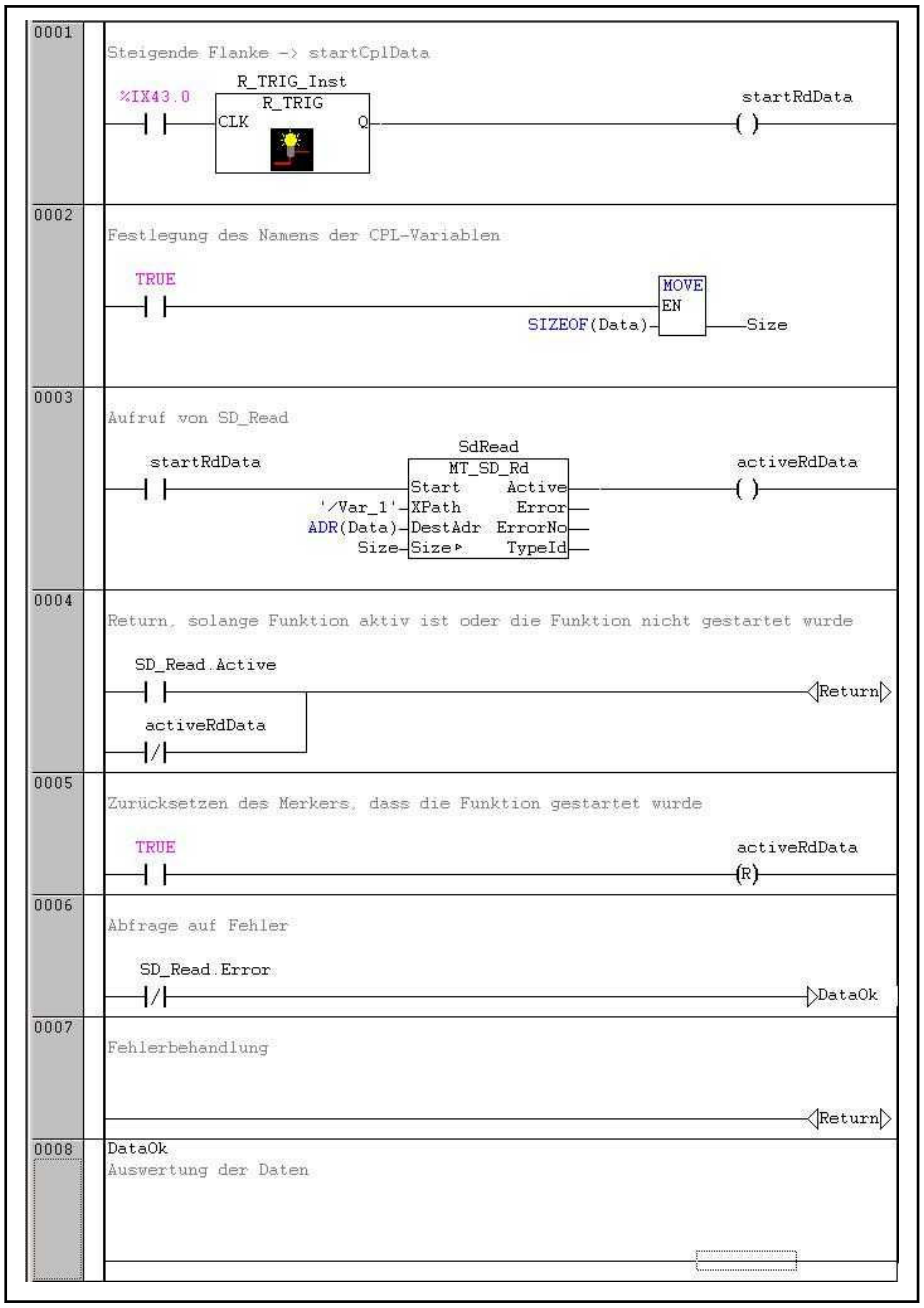


Fig. 12-39: Calling MT_SD_Rd function block

12.20.3 Status messages

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task

General errors and states as acknowledgement after executing the function:

1:	Error in XML library
502:	Internal error of the XML server
507:	Content of the element does not match basic data type acc. to the scheme
508:	Content of the element does not match the facet acc. to the scheme
509:	Internal error while validating
510:	Syntax error in the path string
511:	Segment path is not located in RootPath (internal)
512:	Element position invalid acc. to scheme (path)
513:	An element specified via path cannot be created
514:	The element specified via path cannot be deleted
515:	The element specified via path is not included in the XML files
516:	The element specified via path is not declared in the schema
517:	The last element in path is not identical with the first element in data
518:	The element of the xml file or xml string is not declared in the schema
519:	The data size does not match specified element
520:	Data type does not match specified element
521:	Too much binary files were requested
522:	Unknown data type
523:	Invalid data size
524:	Unknown root element
4001:	Internal error of system data server
4004:	Data buffer too small
4101:	Invalid mode (internal)
4102:	Invalid Xpath

12.21 Writing System Data (MT_SD_WR)

12.21.1 Function

The program function block provides functions to write system data.

Entire and partial variables can be written. It is selected via the input variable **XPath** (e.g. /VAR_1). The data has to be transferred to the format they were defined on the NC side by the scheme files. The size in **Size** also has to match the type size.

The ID of the type, by which the variable is defined on the NC side, has to be defined in **TypeId**. This is returned when reading variables of this type. In case of multiple write accesses, this ID has to be specified only once before the first access. It can then be transferred during each successive access.

Restrictions:

- No complete arrays can be written on the lowest level (e.g. /ArrayOfInts). On the other hand, complete arrays that are subelements of a system variable can be written (e.g. /ArrayOfInts).

- The size of the requested data is limited to 4 KB.

Several instances of the function block can be called in one PLC cycle.

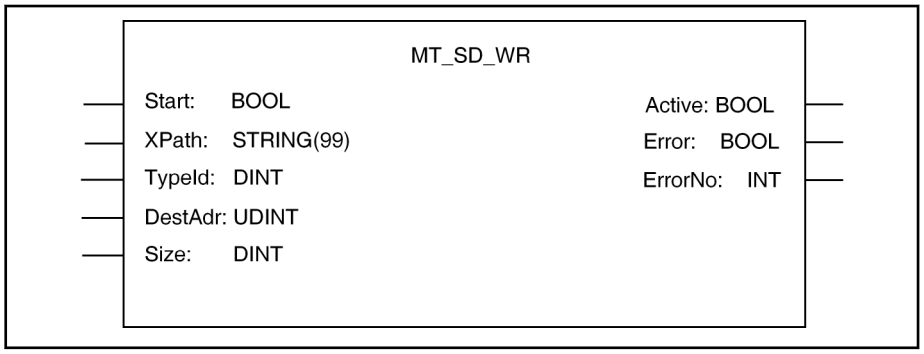


Fig. 12-40: Calling MT_SD_WR function block

Function block parameters

The function block is called with the following parameters:

Start:	Flag that executes a new request. A new request is executed if **Start** =TRUE and **Active** = FALSE
XPath:	Identification of requested system data including Path specifications (e.g. "/Var_1")
TypeId:	Type ID of the system date
DataAdr:	Address of data (ADR(<variable>)) The data has to be provided in the format they were defined on the NC side by the scheme files
Size:	Size of transferred data in bytes In this case, the size has to match the size of the system date

The return parameters are:

Active:	Flag that displays whether the function is still active
Error:	Error detection: FALSE means that the function has been processed without errors
ErrorNo:	Error code: Only valid if Error = TRUE

The function starts when the **Start** bit is set and "Active" is not set. If the "Start" bit is not reset using the PLC sequential program, it is reset when the function has been completed.

Active is set in the status until the function has been completed. As soon as data is returned, the function is completed and **Active** is reset.

12.21.2 Configuration example

Variable "/Var_1.Int_2" is to be written with a value of 1234.

The task is started with a rising edge at the **E43.4** input.

The **MT_SD_Wr** function block could be called as follows:

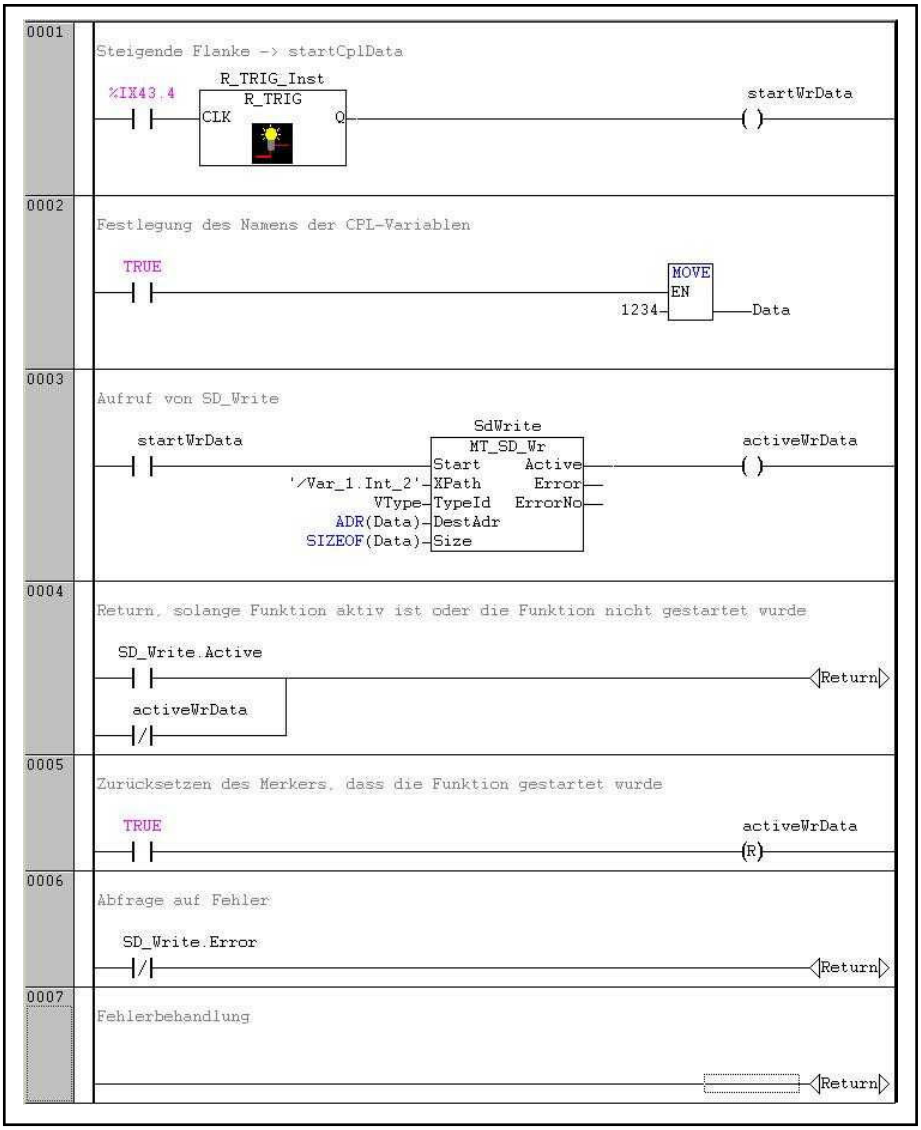


Fig. 12-41: Calling MT_SD_Wr function block

12.21.3 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

-6:	Currently there are no resources available to transmit the task

General errors and states as acknowledgement after executing the function:

1:	Error in XML library
502:	Internal error of the XML server
507:	Content of the element does not match basic data type acc. to the scheme
508:	Content of the element does not match the facet acc. to the scheme

509: Internal error while validating

510: Syntax error in the path string

511: Segment path is not located in RootPath (internal)

512: Element position invalid acc. to scheme (path)

513: An element specified via path cannot be created

514: The element specified via path cannot be deleted

515: The element specified via path is not included in the XML files

516: The element specified via path is not declared in the schema

517: The last element in path is not identical with the first element in data

518: The element of the xml file or xml string is not declared in the schema

519: The data size does not match specified element

520: Data type does not match specified element

521: Too much binary files were requested

522: Unknown data type

523: Invalid data size

524: Unknown root element

4001: Internal error of system data server

4004: Data buffer too small

4101: Invalid mode (internal)

4102: Invalid Xpath

12.22 Reading diagnostic data (MT_DiagCode)

12.22.1 General information

The function block " MT_DiagCode" provides an image of the content of the NC error and message database. The diagnostic messages can be restricted to relevant processing channels and/or relevant message classes via a corresponding input switching. The relevant classes are selected using an OR operation of the individual constants such as **16#00000004 OR 16#00000008** to select drive and interpolator errors.

A diagnostic entry is represented using a structure with information on:

- Date
- Time
- Error class
- Diagnostic number

The user provides the memory for a list of these structures. The function block enters the diagnostic entries in decreasing temporal sequence (latest first) into the list. If the list provided by the user is too small for the errors already occurred, all errors outside the list are not visible for the user. A report on new messages and their number is given at the output by the function block. Note that the diagnostic number is unique across all diagnostic classes except for the MZA messages. MZA messages are provided with the message number projected in the PLC (bit position). NC program and drive diagnostics are represented using one or more collected diagnostic numbers in

the NC. Distinguishing between single NC program or drive diagnostics is therefore not possible.

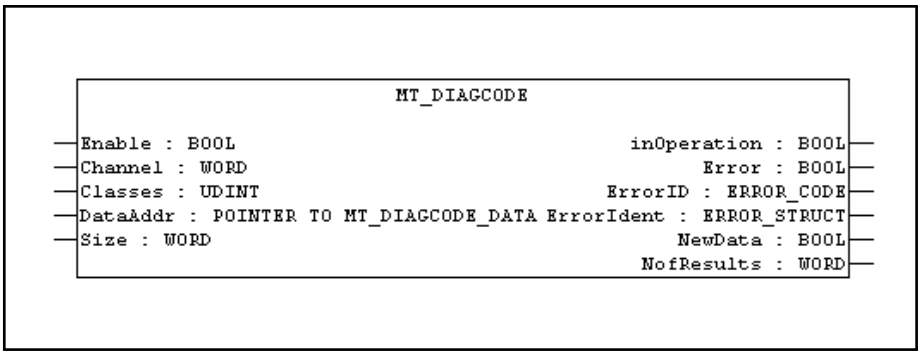


Fig. 12-42: Calling MT_DiagCode function block

12.22.2 Function block parameters

☞ From version 14V04: Parameter channel change

As mask to select several channels, the parameter can only be used for channels 1-12. To read out the information of channel numbers > 12, the channel number and additionally the most significant bit has to be set as parameter. Only data of an individual channel can be requested in case of channel numbers > 12. If all channel-independent information is to be provided when specifying a channel number, bit 14 has to be set in addition to the most significant bit 15. This can be done via an OR operation using 16#8000 or 16#4000.

The function block is called with the following parameters:

Enable:	Enabling input All input parameters are applied at a rising edge at this input.
Channel:	Filter for processing channels: Bit 0 = 1 channel-independent diagnostic messages Bit m = 1 (1<=m<=n) Activates the diagnostics for channel m, n: Number of channels in the system **(max. 12)** **From Version 14V04:** As mask to select several channels, the parameter can only be used for channels 1-12. To read out the information of channel numbers > 12, the channel number and additionally the most significant bit has to be set as parameter. Only data of an individual channel can be requested in case of channel numbers > 12. If all channel-independent information is to be provided when specifying a channel number, bit 14 has to be set in addition to the most significant bit 15. Example: Requesting data of channel 1, 2, 3 and 6 using the channel mask: Channel := 2#1001110 ; Requesting data of channel 27: Channel := 27 OR 16#8000; Requesting data of channel 27 including the channel-independent information: Channel := 27 OR 16#8000 OR 16#4000;

Classes: Filter for message classes:

16#00000001 - Fatal system error

16#00000002 - Non-fatal system error

16#00000004 - Drive error

16#00000008 - Interpolator error

16#00000010 - Hardware error

16#00000020 - ICL error

16#00000040 - Part program error

16#00000080 - CPL error

16#00000100 - GUI error

16#00000200 - DNC error

16#00000400 - Input/output error

16#00000800 - Machine error

16#00001000 - Kernel system warnings

16#00002000 - Periphery warning

16#00004000 - Interface warning

16#00008000 - Runtime warning

16#00010000 - Runtime note

16#00020000 - DNC note

16#00040000 - User note

16#00080000 - General note

16#00100000 - Diagnostic note

16#00200000 - Machine warning

16#00400000 - Machine note

DataAddr: Pointer to one or more structures of the MT_DIAGCODE_DATA type.

The function block fills the data structures with diagnostic information.

The content of the diagnostic structures is only valid if NewData is TRUE

Size: Provided memory size in bytes in the DestAddr

inOperation: Signals via TRUE that the function block monitors the internal diagnostic database for changes.

The output is set to FALSE in case of error (Error=TRUE) or in case of deactivation of the function block via Enable=FALSE (after completion of a possible running reading process)

Error: Signals error states in the function block via TRUE.

More exact diagnostics via "ErrorID" and "ErrorIdent"

ErrorID: Broad error description

ErrorIdent: Detailed error description (for future extensions)

NewData: Signals via TRUE that new diagnostic messages occurred or were deleted considering the filter criteria (channel and classes): the content of the memory range addressed by DataAddr changed

NofResults: Number of diagnostic data blocks in the DataAddr.

Value is only valid if DataAddr = TRUE

Definition of the MT_DIAGCODE_DATA structure:

```
STRUCT
    DateTime :DATE_AND_TIME; //Date and time of the message in UTC
    Chan :UINT; //Channel in which the diagnostics message occurred
    Class :UDINT; //Diagnostics class
    Number :DINT; //Diagnostics number
END_STRUCT
```

12.22.3 Configuration example

All diagnostic messages of channel 1 and all channel-independent messages can be monitored continuously. The memory provided to accept diagnostic information has 32 structure elements. If the content of the diagnostic memory changes, it is copied for further processing.

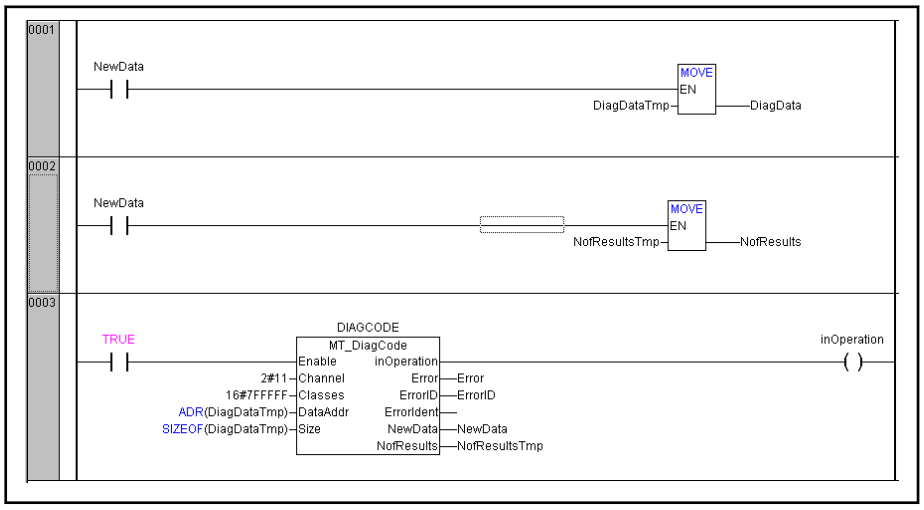


Fig. 12-43: Configuration example

12.22.4 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

6:	Parameter error
-2:	Internal error

12.23 Reading diagnostic texts (MT_DiagText)

12.23.1 General information

The function block MT_DiagText provides an image of the content of the NC error and message database. The diagnostic messages can be restricted to relevant processing channels and/or relevant message classes via a respec-

tive input wiring. The relevant classes are selected using an OR operation of the individual constants such as **16#00000004 OR 16#00000008** to select drive and interpolator errors.

In contrast to the function block MT_DiagCode, the diagnostic text are also provided.

Prerequisite The language-dependent text files have to be loaded as UTF8 format into the directory "text" of the user feprom.

A diagnostic entry is represented using a structure with information on:

- Date, time
- Channel
- Error class
- Diagnostic number
- Diagnostic text

The user provides the memory for a list of these structures, specifying the number of list elements, to the function block. The function block enters the diagnostic entries in decreasing temporal sequence (latest first) into the list. If the list provided by the user is too small for the errors already occurred, all errors outside the list are not visible for the user. A report on the number of new messages and the new messages themselves is given at the output by the function block. Note that the diagnostic number is unique across all diagnostic classes except for the MZA messages. MZA messages are provided with the message number projected in the PLC (bit position). NC program and drive diagnostics are represented using one or more collected diagnostic numbers in the NC. Distinguishing between single NC program or drive diagnostics is therefore not possible.

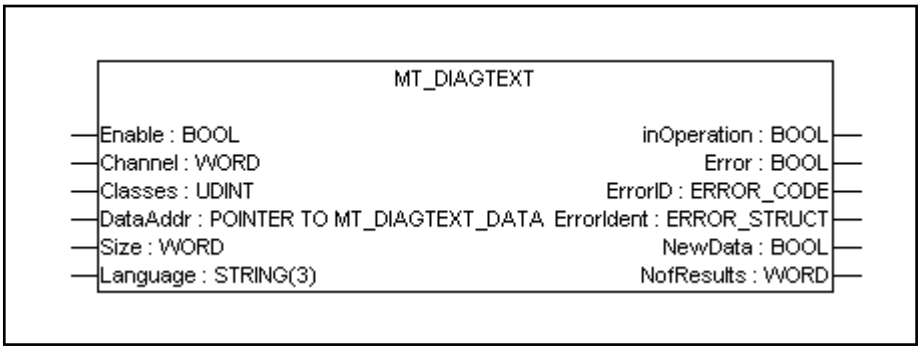


Fig. 12-44: Calling MT_DiagText function block

Restriction A maximum of three instances of the function block can be created for each PLC task.

12.23.2 Function block parameters

☞ From version 14V04: Parameter channel change

As mask to select several channels, the parameter can only be used for channels 1-12. To read out the information of channel numbers > 12, the channel number and additionally the most significant bit has to be set as parameter. Only data of an individual channel can be requested in case of channel numbers > 12. If all channel-independent information is to be provided when specifying a channel number, bit 14 has to be set in addition to the most significant bit 15. This can be done via an OR operation using 16#8000 or 16#4000.

Enable	Activation dialog. All input parameters are applied at a rising edge at this input.
Channel	Filter for processing channels: Bit 0 = 1 Channel-independent diagnostic messages Bit m = 1 (1<=m<=n) Activates the diagnostics for channel m, n: Number of channels in the system **(max. 12)** **From Version 14V04:** As mask to select several channels, the parameter can only be used for channels 1-12. To read out the information of channel numbers > 12, the channel number and additionally the most significant bit has to be set as parameter. Only data of an individual channel can be requested in case of channel numbers > 12. If all channel-independent information is to be provided when specifying a channel number, bit 14 has to be set in addition to the most significant bit 15. Example: Requesting data of channel 1, 2, 3 and 6 using the channel mask: Channel := 2#1001110 ; Requesting data of channel 27: Channel := 27 OR 16#8000; Requesting data of channel 27 including the channel-independent information: Channel := 27 OR 16#8000 OR 16#4000;

Classes	Filter for message classes: 16#00000001 - Fatal system error 16#00000002 - Non-fatal system error 16#00000004 - Drive error 16#00000008 - Interpolator error 16#00000010 - Hardware error 16#00000020 - ICL error 16#00000040 - Part program error 16#00000080 - CPL error 16#00000100 - GUI error 16#00000200 - DNC error 16#00000400 - Input/output error 16#00000800 - Machine error 16#00001000 - Kernel system warning 16#00002000 - Periphery warning 16#00004000 - Interface warning 16#00008000 - Runtime warning 16#00010000 - Runtime note 16#00020000 - DNC note 16#00040000 - User note 16#00080000 - General note 16#00100000 - Diagnostic note 16#00200000 - Machine warning 16#00400000 - Machine note
DataAddr	Pointer to one or more structures of the MT_DIAGTEXT_DATA type. The data structures are filled with diagnostic information by the function block. The content of the diagnostic structures is only valid if NewData is TRUE
Size	Provided memory size in bytes in the DestAddr
Language	Country code for the selected language e.g. "049" for German
inOperation	Signals via TRUE that the function block monitors the internal diagnostic database for changes. The output is set to FALSE in case of error (Error=TRUE) or in case of deactivation of the function block via Enable=FALSE (after completion of a possible running reading process).
Error	Signals error states in the function block via TRUE. More exact diagnostics via "ErrorID" and "ErrorIdent".
ErrorID	Broad error description
ErrorIdent	Detailed error description (for future extensions)

NewData	Signals via TRUE that new diagnostic messages occurred or were deleted considering the filter criteria (channel and classes): the content of the memory range addressed by DataAddr changed.
NofResults	Number of diagnostic data blocks in the DataAddr. Value is only valid if DataAddr = TRUE.

Definition of the MT_DIAGTEXT_DATA structure:

Program:

```
STRUCT
    DateTime :DATE_AND_TIME; //Date and time of the message in UTC
    Chan :UINT; //Channel in which the diagnostics message occurred
    Class :UDINT; //Diagnostics class
    Number :DINT; //Diagnostics number
END_STRUCT
```

12.23.3 Configuration example

Refer to chapter 12.22.3 "Configuration example" on page 195.

12.23.4 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

3:	No more instances of the function block can be created
6:	Parameter error
-2:	Internal error

12.24 Reading Motion Control Data Services (MT_MCoDS)

12.24.1 General information

The MT_MCoDS function block allows read-only access to the internal data services of the CNC. The parameterization of the function block is based on the description of the MCoDS command in the Programming Manual (chapter: CPL functions/NCS coupling).

The resulting values are written into a memory provided by the user. The following type mapping is to be considered:

CPL data types	PLC data types
Double	LREAL
Real	LREAL
Integer	DINT
Character	STRING

Tab. 12-2: Type mapping for the function block MT_MCoDS

When using this function block, note that access to internal data services of the CNC cannot be ensured at every point in time. In these cases, the function block cannot provide valid data. The "Error" output is set to TRUE and the result memory is filled with zeros.

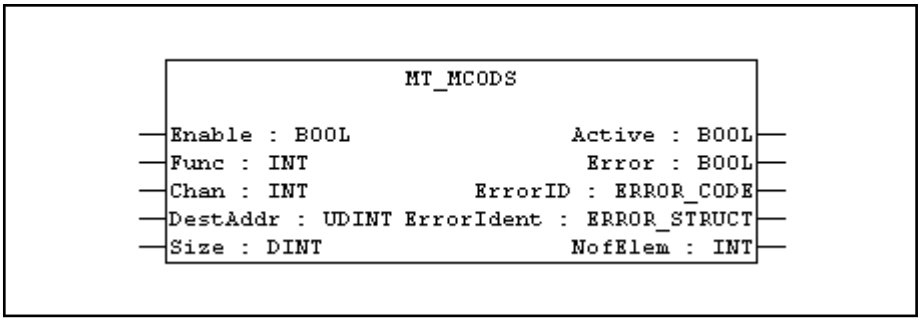


Fig. 12-45: Structure of the MT_MCoDS function block

12.24.2 Function block parameters

The function block is called with the following parameters:

Enable	Enabling input
Func	Function number and requested data service (refer to the MCODS description in the Programming Manual) The functions 42, 55 56, 57, 62, 94 and 95 are not supported.
Chan	Channel specification on which the function is to apply
DataAddr	Pointer to the resulting data range
Size	Memory size in bytes provided in "DestAddr"
Active	Flag that displays whether the function is still active
Error	Signals error states in the function block via TRUE. More exact diagnostics via "ErrorID" and "ErrorIdent"
ErrorID	Broad error description
ErrorIdent	Detailed error description (for future extensions)
NofElem	Number of returned elements

12.24.3 Configuration example

The following example shows the continuous determination of the command positions of all feed and auxiliary axes in an ascending, channel-independent sequence. A field of LREAL values (AxisPos) is provided for the results.

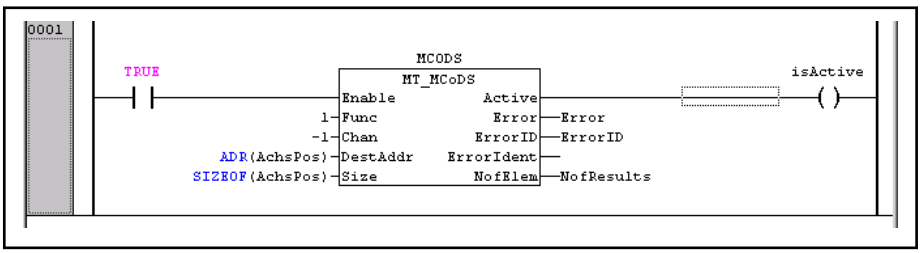


Fig. 12-46: Configuration example for the MT_MCoDS function block

12.24.4 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

6:	Parameter error
-2:	Internal error

12.25 Automatic Sercos III I/O exchange (MT_S3AutoAdapt)

12.25.1 General information

Exchanging an individual Sercos III I/O device in a system currently producing is possible at any time without using interface tools if the machine manufacturer enables the function for an "automatic Sercos III I/O exchange".

The function for the automatic I/O exchange can be switched on and off at any time in the PLC user program via the function block MT_S3AutoAdapt.

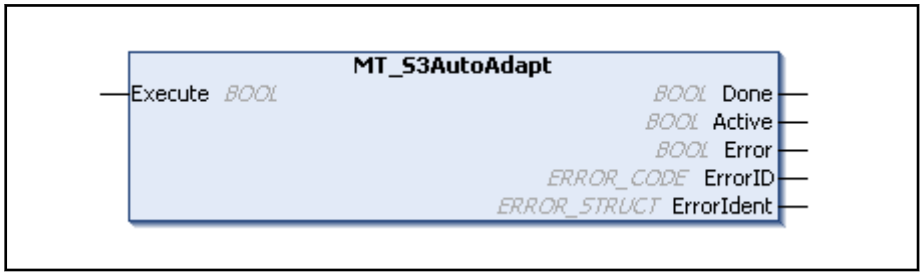


Fig. 12-47: Calling the MT_S3AutoAdapt function block

12.25.2 Function block parameters

The function block is called with the following parameters:

Execute	Flag for new request
Active	Flag that displays whether the function is still active
Error	Error detection: FALSE means that the function has been processed without errors
ErrorID	Error code: only valid if Error = TRUE
ErrorIdent	Detailed error description (for future extensions)

12.25.3 Configuration example

Enabling the function with a rising edge at the %IX41.3 input.

The **MT_S3AutoAdapt** function block can be called as follows:

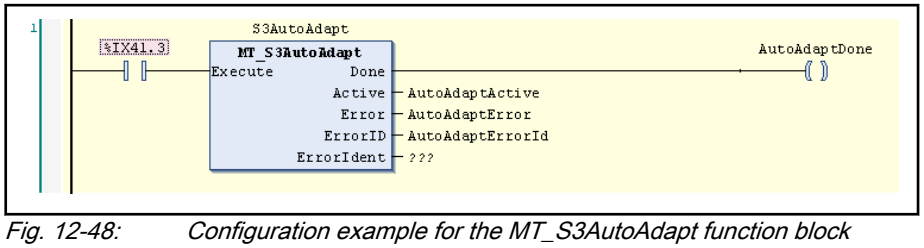


Fig. 12-48: Configuration example for the MT_S3AutoAdapt function block

12.25.4 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states when calling the function with incorrect parameterization or NC-internal error:

0:	No error: Either no address adjustment or the automatic address adaptation was performed without error.
506:	More than two I/O devices were replaced
507:	Automatic address adaptation could not be executed Further diagnostics with the IndraWorks interface
508:	Automatic address adjustment is not supported

12.26 Reading out the Sercos ring topology (MT_RingTop)

12.26.1 General information

This function block returns the Actual topology of the Sercos ring and information about the online status of the Sercos devices.

Additionally, the point of interruption is displayed in case of an interrupted ring.

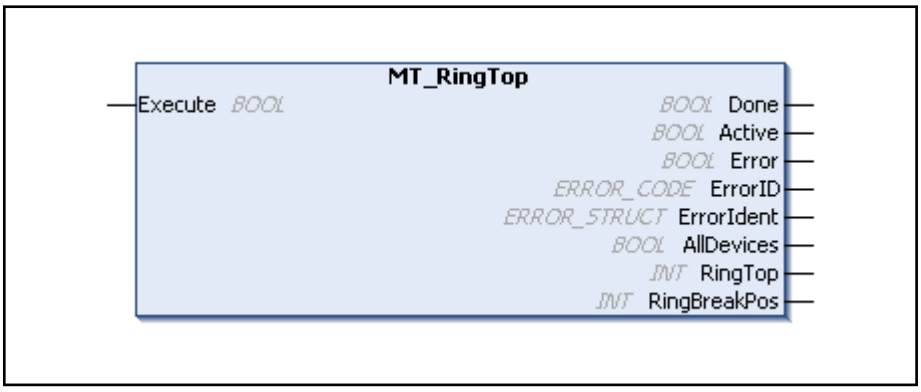


Fig. 12-49: Calling MT_RingTop function block

12.26.2 Function block parameters

The function block is called with the following parameters:

Execute	Flag for new request
Done	Displays that the task was completed successfully
Active	Flag that displays whether the function block is still active
Error	Error detection: FALSE means that the function has been processed without errors
ErrorID	Error code: only valid if Error = TRUE
ErrorIdent	Detailed error description (for future extensions)
AllDevices	TRUE - All devices are active FALSE - One or more devices failed

RingTop	Current ring topology: 0 - Unknown topology The Sercos communication is not active: 1 - A line at Port 1 is active 2 - A line at Port 2 is active 3 - Two lines are active 4 - A closed ring is active
RingBreak-Pos	Position of interruption of ring (last device at Port 1): -1 - A closed ring is active (RingTop = 4) 0 - The ring between Port 1 and the first device is interrupted or a line at Port 2 is closed Otherwise - The ring between Port 2 and this device is interrupted or a line is connected to Port 1

12.26.3 Configuration example

Enabling the function with a rising edge at the %IX41.3 input.

The **MT_RingTop** function block can be called as follows:

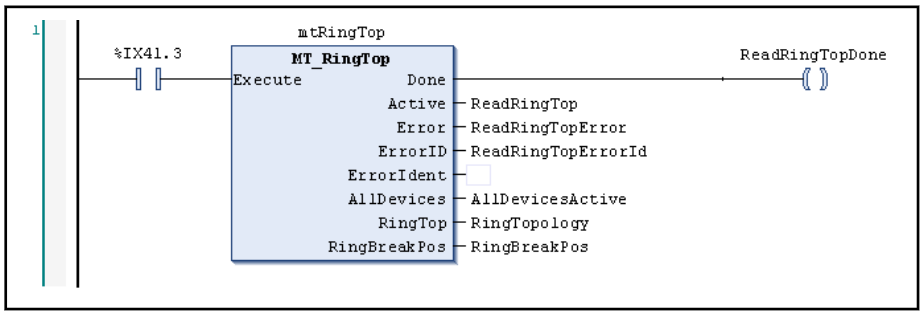


Fig. 12-50: Configuration example for the function block MT_RingTop

12.26.4 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set and an error number is output in "ErrorID".

General errors and states:

-6:	Currently there are no resources available to transmit the task

12.27 Log entry in the action recorder (MT_LogData)

12.27.1 General information

Use the function block "MT_LogData" to write texts (LogText) and contents of PLC variables (VarAddr, VarSize) into the logbook of the action recorder. For later post-editing, an ID (LogTextID) can be specified with the entry. A file name (AddFiles) can be used to assign multiple files stored on the PLC to the log entry. The action recorder copies the respective files to the area of the action recorder on the mount directory. In the action recorder display, this data can also be displayed. Via a bit mask (NcDataSel), the PLC can specify which NC data on the log entry is applied to the logbook. The "ChannelSel" input allows channel-dependent writing of NC data. The log entry type (EventType) is used to classify the machine events in the display of the action re-

corder. Using the "Component" input, the corresponding machine event can be assigned to a component by the PLC. This is especially relevant for critical events and crashes.

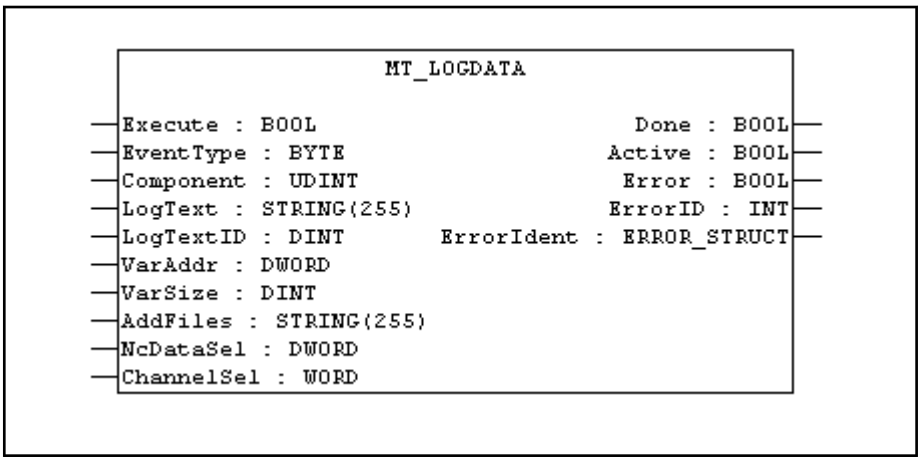


Fig. 12-51: Calling the MT_LogData function block

12.27.2 Function block parameters

The function block is called with the following parameters:

Execute	Activation dialog. All input parameters are applied at a rising edge at this input. If the function block is still active (Active=TRUE), no new tasks are accepted
EventType	Type of logbook entry: MT_LOGDATA_NORMAL (0) MT_LOGDATA_SPECIAL (1) MT_LOGDATA_CRITICAL (2) MT_LOGDATA_CRASH (3) (Changes/extensions possible using the language file "MTXacr_BaseLogData.xxx“)
Component	Freely definable ID of the reporting component: (Changes/extensions using the language file "MTXacr_BaseLogData.xxx“) Example: 1: Spindle 2: Tool changer 3: Workpiece clamp
LogText	Text for the log entry (empty string if not used)
LogTextID	Text ID for post-editing using the language file "MTXacr_AddLogData.xxx“ 0: No post-editing 1: "My Text %d" 2: "My Nice Text" 3: …

VarAddr	Address of the variable whose content is inserted into the text selected by LogTextID. If LogTextID is 0, this parameter is ignored **ATTENTION!** Since the transferred values are interpreted using the format string in the text, the type of the variables has to match the sizes of the format specifications in the text. **For the IndraLogic 2G, a structure has to be defined additionally as packed!**
VarSize	Total size of the variables in bytes
AddFiles	Name and path of the additional files the action recorder copies to the mount directory and integrates them into the ACR management. More than one file can be specified. It is separated by a ";"
NcDataSel	Bit mask to select NC data blocked with his entry: (Multiple selection is possible via OR operation) MT_LOGDATA_NODATA (0): No recording of NC data MT_LOGDATA_AXPOS (1): Recording axis positions MT_LOGDATA_PRGLVL (2): Program level information for all channels (path and file name of the active NC programs as well as the corresponding current position across all active planes) MT_LOGDATA_MODGFUNC (4): Recording modal G-functions MT_LOGDATA_ACTTOOL (8): Recording active tool (active tool and preselected tool with place, sector, number of edges, name and tool number as well as the active edge [L1, L2, L3, R, ORI, Theta, Phi]) MT_LOGDATA_TABLES (16): File name of the active correction table (ZeroOffset , Placement , D-correction table) MT_LOGDATA_AUXFUNC (32): Auxiliary functions
Channel-Sel	Selecting channels for NcData MT_LOGDATA_ALLCHAN : Recording NcData in all channels MT_LOGDATA_CHAN01 : Recording NcData in channel 1 MT_LOGDATA_CHAN02: Recording NcData in channel 2 : MT_LOGDATA_CHAN12 : Recording NcData in channel 12 (**Note**: If axis-specific ones are selected, the data of all axes is applied to the log file
Active	Signals via TRUE that the function block processes the task
Done	Displays that the command was completed successfully
Error	Signals error states in the function block via TRUE. More exact diagnostics are possible via ErrorID.
ErrorID	Error code (only valid if Error = TRUE)
ErrorIdent	Not used

12.27.3 Configuration examples

A log entry of type NORMAL is transferred to the action recorder in the configuration example.

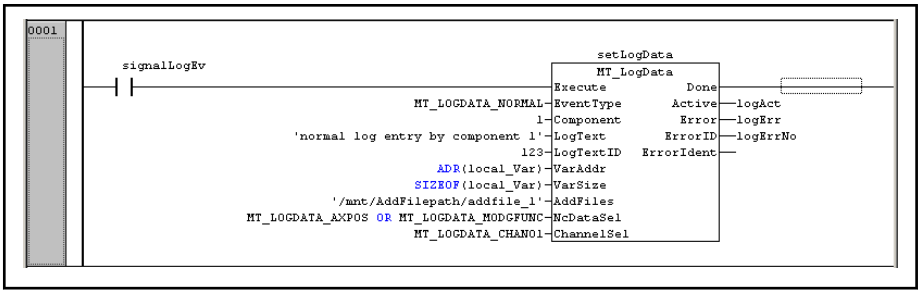


Fig. 12-52: Configuration example for the function block MT_LogData

12.27.4 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states:

6:	Parameter error
-2:	Internal error

Function-specific errors and states:

1:	Internal action recorder error
2:	Action recorder not active
3:	Invalid parameter
4:	Invalid Ncs message
5:	Log entries not active by PLC
6:	Range for variables too high (max. 512)
7:	Text for log entry too high (max. 255 + closing '$00'
8:	Additional file names too high (max. 255 + closing '$00'

12.28 Reading System Data in Segments (MT_SD_Rd_Seg)

12.28.1 General information

The program block offers the possibility to read system data, which are also larger than 4K bytes. Both entire variables and partial variables can be read. It is selected via the input variable XPath (e.g. MyVarArr_1). The data is provided in the format defined on NC side by the scheme files. For a correct access to the data, the structures have to be defined in the PLC program using the pragma {attribute 'pack_mode':='1'}.

Restrictions:

No complete arrays can be read on the lowest level. On the other hand, complete arrays that are subelements of a system variable can be read (e.g. / ArrayOfInts).

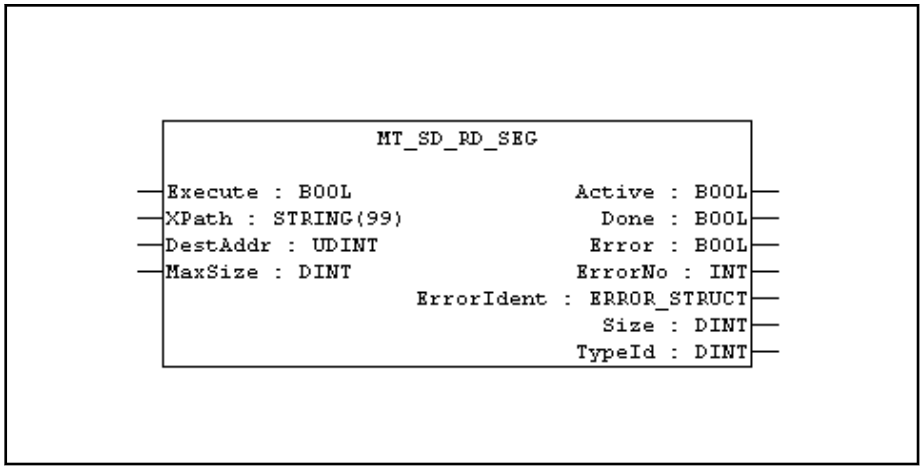


Fig. 12-53: Calling MT_SD_Rd_Seg function block

12.28.2 Function block parameters

The function block is called with the following parameters:

Execute	Activation dialog. All input parameters are applied at a rising edge at this input. If the function block is still active (Active=TRUE), no new tasks are accepted
XPath	Identification of system data including Path specifications (e.g. "MyVar-Arr_1")
DestAddr	Address of data The variable has to be defined in the format defined on NC by the scheme files
MaxSize	Total size of the variables in bytes
Active	Signals via TRUE that the function block processes the task
Done	Displays that the command was completed successfully
Error	Signals error states in the function block via TRUE. More exact diagnostics are possible via ErrorID.
ErrorNo	Error code (only valid if Error = TRUE)
ErrorIdent	Not used

12.28.3 Configuration example

In the configuration example, the system variable **MyVarArr_1** is read and the array element 10 is applied to the local variable **loc_Var** after receiving the data.

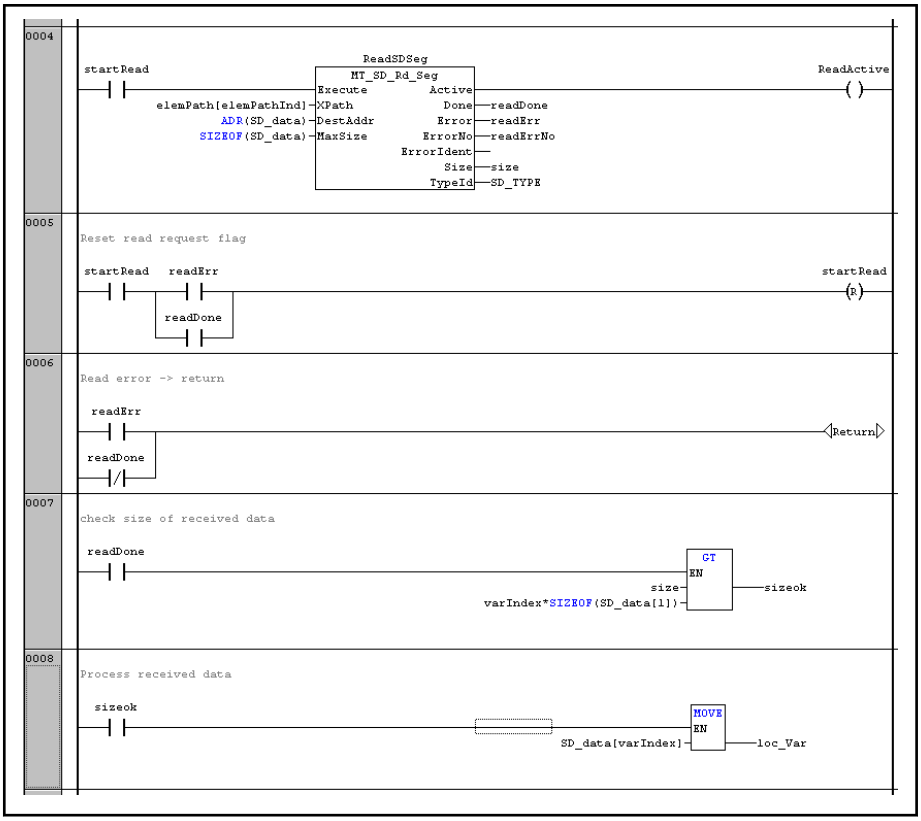


Fig. 12-54: Configuration example for the MT_SD_Rd_Seg function block

12.28.4 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states:

-6:	Currently there are no resources available to transmit the task

Function-specific errors and states:

1:	Error in XML library
507:	Content of the element does not match basic data type acc. to the scheme
508:	Content of the element does not match the facet acc. to the scheme
509:	Internal error while validating
510:	Syntax error in the path string
511:	Segment path is not located in RootPath (internal)
512:	Element position invalid acc. to scheme (path)
513:	An element specified via path cannot be created
514:	The element specified via path cannot be deleted
515:	The element specified via path is not included in the XML files
516:	The element specified via path is not declared in the schema

517:	The last element in path is not identical with the first element in data
518:	The element of the xml file or xml string is not declared in the schema
520:	Data type does not match specified element
521:	Too much binary files were requested
522:	Unknown data type
523:	Invalid data size
524:	Unknown root element
4001:	Internal error of system data server
4004:	Data buffer too small
4101:	Invalid mode (internal)
4102:	Invalid Xpath

12.29 Writing System Data in Segments (MT_SD_Wr_Seg)

12.29.1 General information

The program block offers the possibility to write system data, which are also larger than 4K bytes. Both entire variables and partial variables can be written. It is selected via the input variable XPath (e.g. MyVarArr_1). The data has to be transferred to the format they were defined on NC side by the scheme files. The size in "Size" also has to match the type size in the system data. For a correct access to the data, the structures have to be defined in the PLC program using the pragma {attribute 'pack_mode':='1'}. The ID of the type, by which the variable is defined on NC side, has to be defined in "TypeId." This is returned when reading variables of this type. In case of multiple write accesses, this ID has to be specified only once before the first access. It can then be transferred during each successive access.

Restrictions:

No complete arrays can be read on the lowest level. On the other hand, complete arrays that are subelements of a system variable can be read (e.g. /ArrayOfInts).

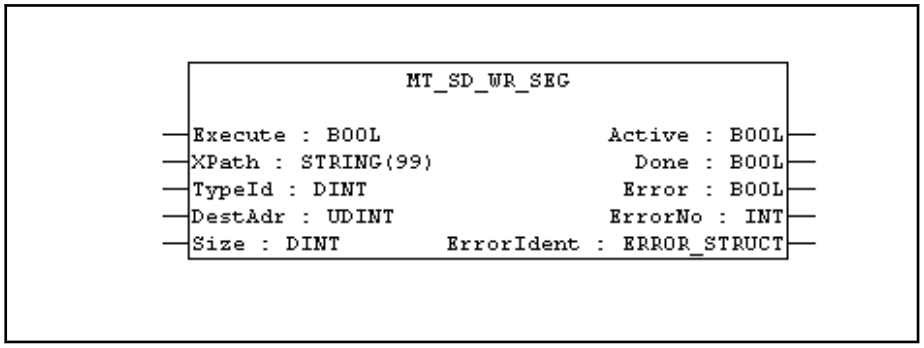


Fig. 12-55: Calling MT_SD_Wr_Seg function block

12.29.2 Function block parameters

The function block is called with the following parameters:

Execute	Activation dialog. All input parameters are applied at a rising edge at this input. If the function block is still active (Active=TRUE), no new tasks are accepted
XPath	Identification of system data including Path specifications (e.g. "MyVar-Arr_1")
DestAddr	Address of data The variable has to be defined in the format defined on NC by the scheme files
MaxSize	Total size of the variables in bytes
Active	Signals via TRUE that the function block processes the task
Done	Displays that the command was completed successfully
Error	Signals error states in the function block via TRUE. More exact diagnostics are possible via ErrorID.
ErrorNo	Error code (only valid if Error = TRUE)
ErrorIdent	Not used

12.29.3 Configuration example

In the configuration example, the system variable **MyVarArr_1** is written completely. The "TypeId" of the system data was determined once before by reading the variable. In this case, the size of **MyVarArr_1** has to match the size of the system date.

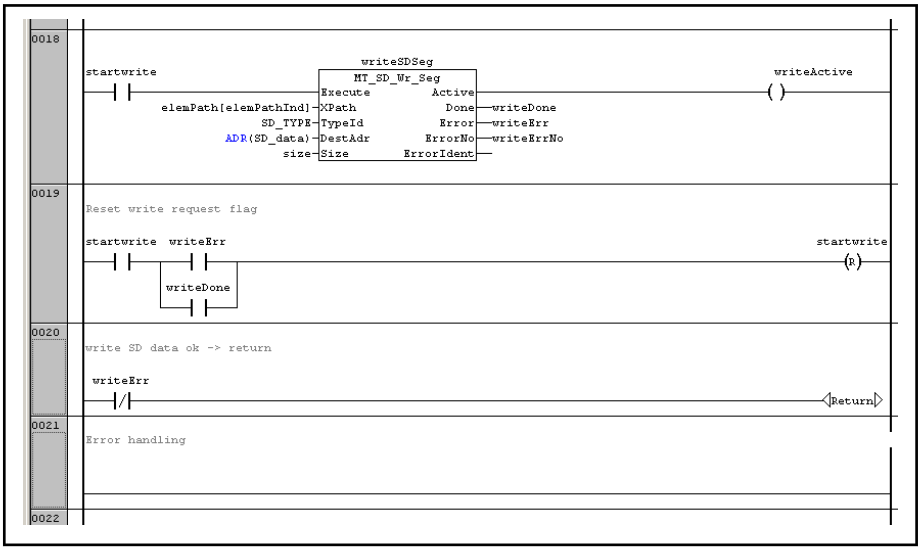


Fig. 12-56: Configuration example for the MT_SD_Wr_Seg function block

12.29.4 Status messages

If an error or state occurred which prevents proper execution of the function after calling it, the error bit "Error" is set. Furthermore, the type of error or state is represented in the "ErrorNo" as number:

General errors and states:

-6:	Currently there are no resources available to transmit the task

Function-specific errors and states:

1:	Error in XML library
507:	Content of the element does not match basic data type acc. to the scheme
508:	Content of the element does not match the facet acc. to the scheme
509:	Internal error while validating
510:	Syntax error in the path string
511:	Segment path is not located in RootPath (internal)
512:	Element position invalid acc. to scheme (path)
513:	An element specified via path cannot be created
514:	The element specified via path cannot be deleted
515:	The element specified via path is not included in the XML files
516:	The element specified via path is not declared in the schema
517:	The last element in path is not identical with the first element in data
518:	The element of the xml file or xml string is not declared in the schema
520:	Data type does not match specified element
521:	Too much binary files were requested
522:	Unknown data type
523:	Invalid data size
524:	Unknown root element
4001:	Internal error of system data server
4004:	Data buffer too small
4101:	Invalid mode (internal)
4102:	Invalid Xpath

12.30 Reading fast action input data (MT_FaDataRead)

12.30.1 General information

This function is used to read the cyclically transferred data from the interpolator to the "fast PLC task". At the beginning of the "fast PLC task", the input data has to be read using this function. After reading in the data, start the actual program processing.

The size of the structure for the input data has to be the same as the input data configured for this fast PLC task on the NC

☞ Since the data has to be transferred as packed on NC side, the structure for type definition on PLC side also has to be defined as packed with **{attribute 'pack_mode' := '1'}**.

For a detailed description, refer to the documentation **MTX Functional Description - Special Functions** in chapter **Fast actions MTX fa**

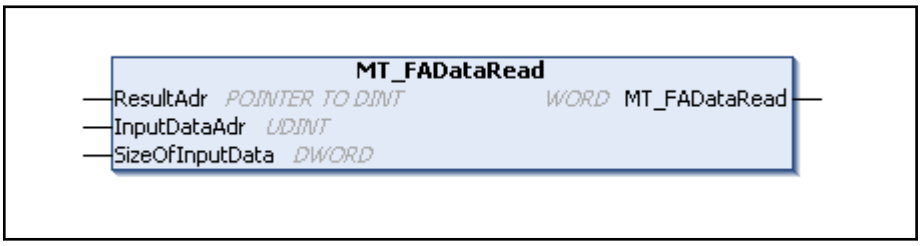


Fig. 12-57: Calling MT_FaDataRead function block

12.30.2 Function block parameters

The function block is called with the following parameters:

Function value	The function value includes the result of the function This is the result of the library function and not the result of an NC-sided fast action (see below)
ResultAdr	Reserved for future extensions (pointer to "Return" value of the NC-sided fast action). It is currently not used
InputDataAdr	Pointer to data transferred from the interpolator to the "fast PLC task"
SizeOfInputData	Total size of the data range of the input data in bytes

12.30.3 Configuration example

One fast PLC task is configured with two fast actions in the configuration example. Apart from the call of the function MT_FaDataRead, the example also includes the call of the respective function to write output data:

1. Action: The NC provides the command positions of two axes of type LREAL (2*8 bytes)

2. Action: The PLC requests the NC to write a CPL variable of type DINT (4 bytes)

The type for input and output data is defined in faInp_t or faOutp_t respectively.

Task logic: If one of the command values exceeds a limit, this value is returned to the output data (CPL variable) of the second fast action

```
1  {attribute 'pack_mode' := '1'}
2  TYPE faInp_t :
3  STRUCT
4      sollPos : ARRAY[0..1] OF LREAL  ; (* 2 required positions of the type LREAL *)
5  END_STRUCT
6  END_TYPE
7
```

Fig. 12-58: Structure for input data

```
{attribute 'pack_mode' := '1'}
TYPE faOut_t :
STRUCT
    CplVarVal  : DINT; (* default value of the CPL-variable *)
END_STRUCT
END_TYPE
```

Fig. 12-59: Structure for output data

```
PROGRAM FAction
VAR

    NcResult            : DINT;                 (* Result of the NC *)
    FaInputData         : faInp_t ;             (* Structure of the Input Values *)
    FaOutputData        : faOut_t ;             (* Structure of the Input Values *)
    fbErr               : WORD;                 (* Result of the POU-Call *)
    maxVal              : LREAL := 123456.789 ;
    err: BOOL;
    greater: BOOL;
END_VAR
```

Fig. 12-60: Declaration part of the program

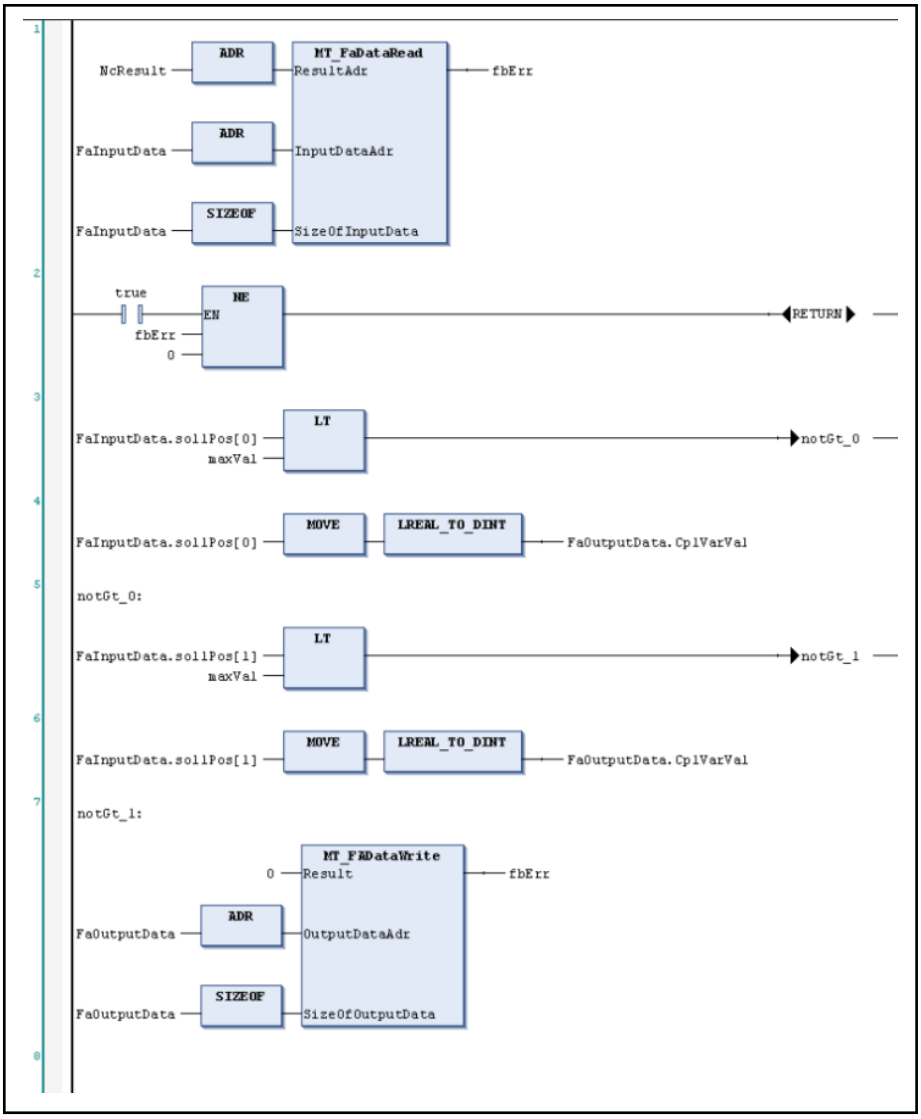


Fig. 12-61: Configuration example for the functions MT_FaDataRead and MT_FaDataWrite

12.30.4 Status messages

The function value can assume the following values:

0x0000:	Data transfer OK
0x000A:	The function was not executed, since it was not called from a "fast Plc task"
0x000B:	Configuration error: No input data is configured for the task or the size to be transferred in **SizeOfInputData** is 0
0x000C:	The size of the configured input data does not match the size transmitted in **SizeOfInputData**
0x000D:	Target address in **InputDataAdr** is a zero pointer
0x7FFE:	Internal error

12.31 Writing fast action output data (MT_FaDataWrite)

12.31.1 General information

This function is used to write the cyclically transferred data from the "fast PLC task" to the interpolator. At the end of the "fast PLC task", the output data has to be transferred to the interpolator using this function.

The size of the structure for the output data has to be the same as the output data configured for this Fast PLC task on the NC

☞ Since the data has to be transferred as packed on NC side, the structure for type definition on PLC side also has to be defined as packed with **{attribute 'pack_mode' := '1'}**.

For a detailed description, refer to the manual **Functional Description 13VRS Special Functions** in chapter **Fast actions MTX fa**

☞ This function block may only be called from a "fast PLC task" (the task has to be of type **Externally event-triggered** and the external event has to be **EVENT_FASTACTION_x**. x stand for index 1 to 4)

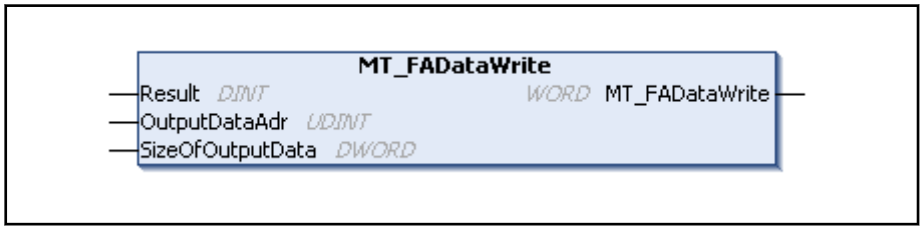


Fig. 12-62: Calling the MT_FaDataWrite function block

12.31.2 Function block parameters

The function block is called with the following parameters:

Function value	The function value includes the result of the library function
OutputDataAdr	Pointer to the data transferred from the "fast Plc task" to the interpolator
SizeOfOutputData	Total size of the data range of the output data in bytes
Result	"Return" value of the "fast Plc task" to the interpolator

12.31.3 Status messages

The function value can return the following values:

0x0000:	Data transfer OK
0x000A:	The function was not executed, since it was not called from a "fast Plc task"
0x000B:	Configuration error: No input data is configured for the task or the size to be transferred in **SizeOfInputData** is 0

0x000C:	The size of the configured input data does not match the size transmitted in **SizeOfInputData**
0x000D:	Target address in **InputDataAdr** is a zero pointer
0x7FFE:	Internal error

12.31.4 Configuration example

One fast PLC task is configured with two fast actions in the configuration example. Apart from the call of the function MT_FaDataWrite, the example also includes the call of the respective function to read input data:

1. Action: The NC provides the command positions of two axes of type LREAL (2*8 bytes)

2. Action: The PLC requests the NC to write a CPL variable of type DINT (4 bytes)

The type for input and output data is defined in faInp_t or faOutp_t respectively.

Task logic: If one of the command values exceeds a limit, this value is returned to the CPL variable of the second fast action

```
{attribute 'pack_mode' := '1'}
TYPE faInp_t :
STRUCT
    sollPos : ARRAY[0..1] OF LREAL  ; (* 2 required positions of the type LREAL *)
END_STRUCT
END_TYPE
```

Fig. 12-63: Structure for input data

```
{attribute 'pack_mode' := '1'}
TYPE faOut_t :
STRUCT
    CplVarVal  : DINT; (* default value of the CPL-variable *)
END_STRUCT
END_TYPE
```

Fig. 12-64: Structure for output data

```
PROGRAM FAction
VAR

    NcResult            : DINT;             (* Result of the NC *)
    FaInputData         : faInp_t ;         (* Structure of the Input Values *)
    FaOutputData        : faOut_t ;         (* Structure of the Input Values *)
    fbErr               : WORD;             (* Result of the POU-Call *)
    maxVal              : LREAL := 123456.789 ;
    err: BOOL;
    greater: BOOL;
END_VAR
```

Fig. 12-65: Declaration part of the program

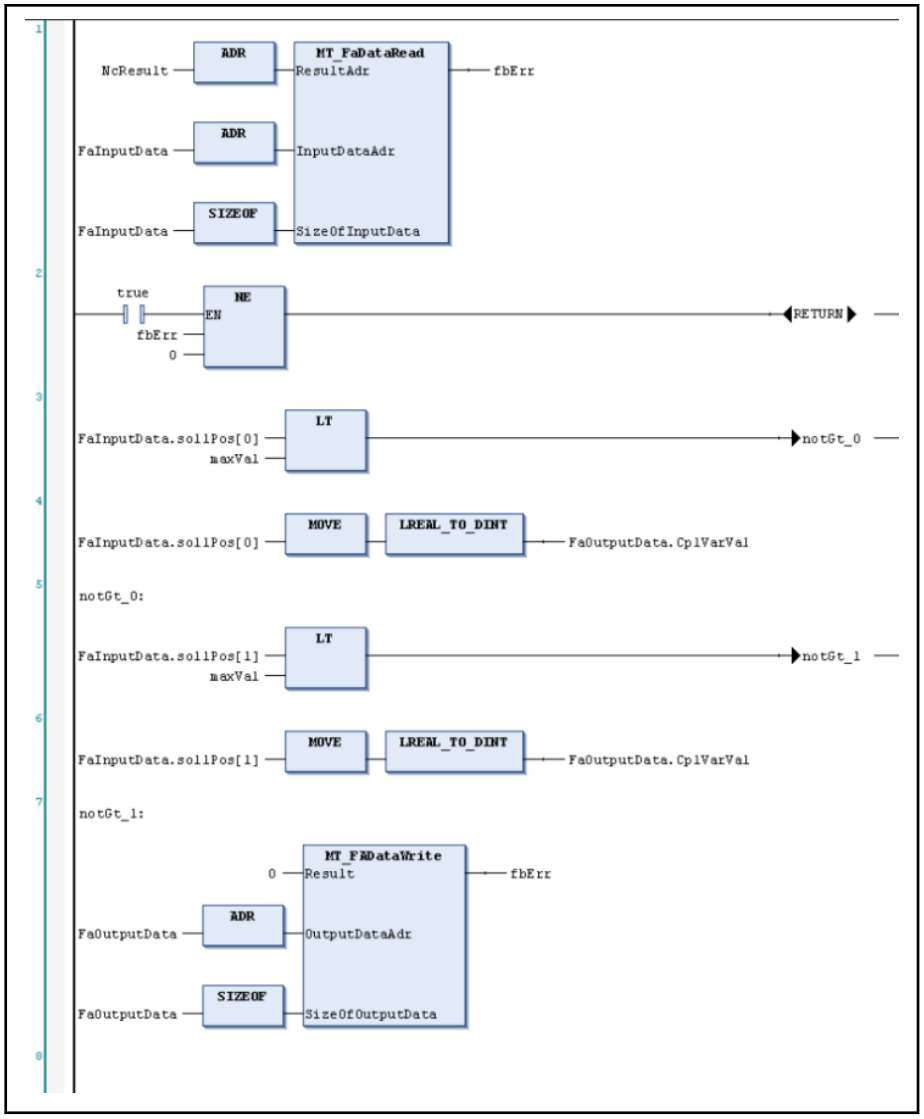


Fig. 12-66: Configuration example for the functions MT_FaDataRead and MT_FaDataWrite

13 Configuring PLC-specific data in IndraWorks

13.1 Configuring machine state display (MZA)

13.1.1 General information

The machine state display enables the diagnostics of machine errors, warnings and notes and thus supports the fast remedy of malfunctions in the operational procedure at the machine.

The MZA diagnostics are displayed in the MTX-HMI in diagnostic header and in the OP9 "Diagnostics". MZA diagnostics can be recorded in the logbook.

The corresponding messages are defined in plain text in the MZA file "mzatexte.xxx" and activated by setting the corresponding signal at the MZA bit interface in the PLC program.

13.1.2 Parameterizing the machine state display (MZA)

MZA signals are cyclically exchanged between PLC and NC. Set the time interval in the parameter "MachStatDiagCycTime". The time should be set to e.g. 300 ms to keep communication load of the system low.

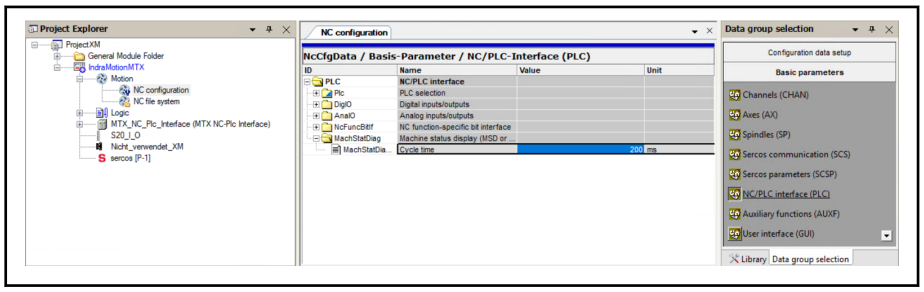


Fig. 13-1: MZA cycle time

The cycle time "0" switches off the MZA diagnostics.

13.1.3 Configuring MZA bit interface

Depending on the number of machine errors, warnings and notes, use the following bit interface data types with a different number of signals:

- "qMZA" with up to 2048 signals or
- "qMZAext" with up to 8192 signals.

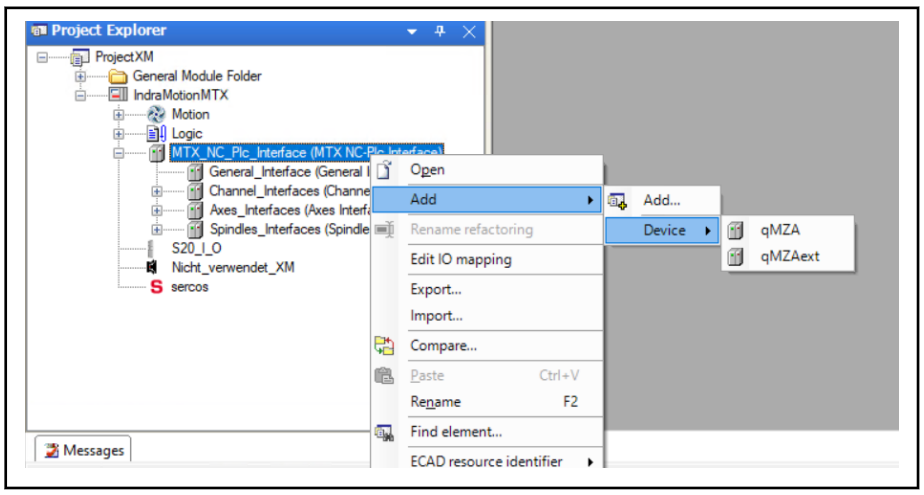


Fig. 13-2: Inserting MZA interface signals

The following settings can be made in the MZA "Properties" dialog (see following figure):

- The identifier for the MZA bit interface can be freely selected. Therefore, right click on the identifier of MZA node below the "MTX_NC_Plc_Interface" node. The "Rename" command can now be selected in the context menu. It is recommended to apply the suggested identifier "qMZA" or "qMZAext".
- The memory address from which the selected data type in the bit interface begins can be freely selected as long as this does not overlap with other addresses in the bit interface.

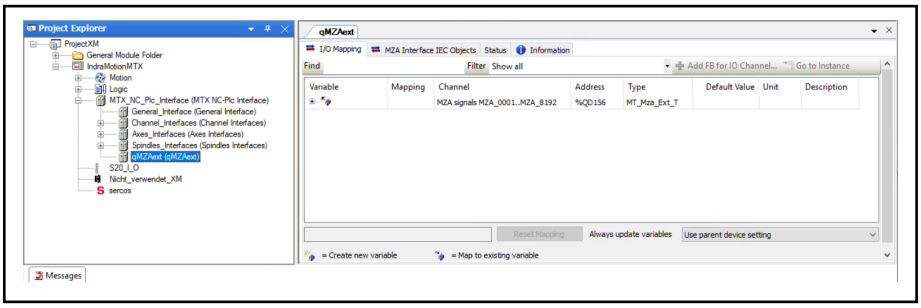


Fig. 13-3: MZA "Properties" dialog

13.1.4 MZA file structure

The MZA file has to be stored in the root directory "/" or the user FEPROM "/usrfep" of the control under the name **"mzatexte.xxx"**. The file name extension "xxx" stands for the used language (e.g. 049 for German).

The MZA text is divided into main text, auxiliary text and remedy text. It is separated with a double number sign "##". Long texts can be continued with the "_" character at line end in the next line.

Example:

MZA text

mzatexte.049

...

(E123, E-STOP button is pressed at the station! ## _

The E-STOP button is still pressed at the station! ## _

If the E-STOP does not apply anymore, _

the power can be switched on!)

(M124, coolant missing! ## The coolant level has fallen below the minimum level _

fallen below! ## _

Fill up coolant if necessary.)

...

☞ Close each MZA definition - even the one at file end - after the right parenthesis with line feed. Otherwise, the message texts are not applied.

The syntax of MZA definition in the file "mzatexte.xxx" is displayed as follows:

(<Diagnostic class><MSD number>,<Main text>##<Help text>##<Recovery text>)

<Diagnostic class>	• **E** for machine error • **W** for machine warning • **M** for machine note
<MSD number>	Bit in the structure qMZA [1..2048] or qMZA_Ext [1..8192] and simultaneous message number in the diagnostics.
<Main text>	Mostly single-line message text also shown in the status line.
<Help text>	Auxiliary text for detailed description of the diagnostics.
<Recovery text>	Description to eliminate of the occurred malfunction.

The texts which cannot be displayed in the ASCII character set can be entered in the UTF8 character coding. An MZA file in UTF8 format has to have the hexadecimal 3 byte file identification "EFBBBF" at the beginning of file. The texts are transmitted with a maximum length of 1000 characters. When using the UTF8 format, the text length reduces by up to 2/3 of the displayable characters with regard to the coding effort.

13.1.5 Enabling messages

The MZA messages are activated in the PLC by setting the requested bit signals.

Example:

Structured text

```
qMZAext.MZA_0004:=TRUE;
```

Resetting the MZA messages has to be programmed in the PLC as well.

Example:

Structured text

```
qMZAext.MZA_0004:=FALSE;
```

13.1.6 Supplementing notes

The control-internal error database cannot store an unlimited number of errors.

*The **maximum** following messages can be stored from the MZAs:*

- 16 errors
- 16 warnings and
- 16 notes

If there are more messages (for example more than 16 machine errors), only the errors with the highest priority are saved in the internal error database. The priority is specified by the message number. Small numbers mean high priority.

If messages are deleted, messages with a lower priority can "succeed" to the database. At the point of "succeeding", it is unknown when the respective message occurred, since the time stamp is not saved.

"Succeeding" messages therefore get the time stamp 1.1.1970 00:00:00.000.

*"Succeeding" only occurs if either **more** than*

- 16 machine errors or
- 16 machine warnings or
- 16 machine notes are present at the same time.

The range of the monitored MZA signals is specified in the "mzatexte.xxx" file. Thus, only PLC-MZA signals between qMZA(ext).MZA_0001 and qMZA(_Ext).MZA_nnnn are analyzed. "nnnn" results either from the highest error number (Ennnn,...), warning number (Wnnnn,...) or note number (Mnnnn,...) entered into the "mzatexte.xxx" file. Signals outside this range are ignored. Thus, transmission time for non-required signals is saved. Signals within the range but without the corresponding text are displayed automatically as machine warning (<no.>).

14 MT_BaseAdditional.library

14.1 Introduction and overview

The functions and function blocks of this library supplement and simplify the basic functionality provided with the MT_Base library.

Overview on channel functions		
Category	**Function/function block**	**Brief description**
General information	MT_ChNumberValidation, page 240	Checks whether the specified channel number exists in the NC configuration
	MT_ChGetState, page 242	Determine current channel
Feed	MT_ChSetFeedOverride, page 254	Change the channel feed
	MT_GetFeedOverride, page 255	Determining the active channel feed
NC program	MT_ChSetReentryBreakPoint, page 251	NC program, saving breakpoint
	MT_ChGetActiveNcProgram, page 244	Determining the name of the active NC program
	MT_ChNcProgProgress, page 14.3.5	Determining current NC program progress
Operation mode	MT_ChOPMode, page 268	Determining and changing the operation modes of a channel (combines MT_ChGetOPMode and MT_ChSetOPMode)
	MT_ChGetOPMode, page 256	Determines the active main and secondary operation mode of a channel
	MT_ChSetOPMode, page 263	Changes the main and secondary operation modes of a channel
	MT_ChSetMainOPMode, page 259	Changes the main operation modes of a channel

Tab. 14-1: Overview on channel functions

Overview on axis functions		
Category	**Function/function block**	**Brief description**
General information	MT_DrvGetSafeState, page 274	Determines the active safety engineering operating state of a drive (also for spindle)
	MT_AxGetChIndex, page 278	Check channel assignment of an axis
	MT_AxHandwheelACS, page 278	Operates handwheel for an axis in the axis coordinate system
	MT_AxProbeMonitor, page 280	Monitoring of a probe axis

Asynchronous axes	MT_AxAsyncSetOverride, page 301	Change feed for asynchronous axes
	MT_AxAsyncOPMode, page298	Determines and changes the operation mode of an asynchronous axis (combines MT_AxAsyncGetOpMode and MT_AxAsyncSetOpMode)
	MT_AxAsyncGetOPMode, page 294	Determines the active operation mode of an asynchronous axis
	MT_AxAsyncSetOPMode, page 295	Changes the operation mode of an asynchronous axis
Jog distance	MT_AxJogMode, page 289	Determines and changes the active jog distance of an axis (combines MT_AxGetJogMode and MT_AxSetJogMode)
	MT_AxGetJogMode, page 283	Determines the active jog distance for an axis
	MT_AxSetJogMode, page 285	Changes the active jog distance for an axis
Power & diagnostics	MT_AxPower, page 308	Determines the drive status and sets/resets the controller enable (combines MT_AxSetPowerOn, MT_AxSetPowerOff and MT_AxGetPowerStateExt)
	MT_AxSetPowerOn, page 305	Sets the controller enable for an axis
	MT_AxSetPowerOff, page 306	Resets the controller enable for an axis
	MT_AxGetPowerState, page 302	Determines the drive status of an axis
	MT_AxGetPowerStateExt, page 303	Determines the drive status of an axis including the detailed information about errors and warnings

Tab. 14-2: Overview on axis functions

Overview on spindle functions		
Category	**Function/function block**	**Brief description**
General information	MT_SpSetOverride, page 312	Change feed for a spindle
Jog velocity	MT_SpJogSpeed, page 317	Determines and changes the active jog velocity of a spindle (combines (MT_SpGetJogSpeed and MT_SpSetJogSpeed)
	MT_SpGetJogSpeed, page 313	Determines the current jog velocity of a spindle
	MT_SpSetJogSpeed, page 314	Changes the active jog velocity of a spindle

Power & diagnostics	MT_SpPower, page 327	Determines the drive status and sets/ resets the controller enable (combines MT_SpSetPowerOn, MT_SpSetPowerOff and MT_SpGetPowerStateExt)
	MT_SpSetPowerOn, page 323	Sets controller enable for a spindle
	MT_SpSetPowerOff, page 325	Resets the controller enable of a spindle
	MT_SpGetPowerState, page 320	Determines the drive status of a spindle
	MT_SpGetPowerStateExt, page 322	Determines the drive status of a spindle including the detailed information about errors and warning

Tab. 14-3: Overview on spindle functions

Overview on tool management functions		
Category	**Function/function block**	**Brief description**
Tool life	MT_TmToolLife, page 330	Determines the tool service life

Tab. 14-4: Overview on tool management functions

Overview on miscellaneous functions		
Category	**Function/function block**	**Brief description**
Other axes	MT_DistanceAccumulator, page 336	Calculates the distance from position data
	MT_TimeAccumulator, page 339	Calculate the lapsed time
	MT_GetRandomNumber, page 341	Generates an integer random number

Tab. 14-5: Overview on miscellaneous functions

14.2 Standard interfaces at function blocks

14.2.1 Motivation

The majority of the function blocks disposes of an input for the activation and an output for displaying the correct processing.

In addition, an output is often necessary which indicates the processing time.

Additional outputs are defined for the indication of errors.

A uniform naming convention as well as an identical standard interface behavior, increases understanding, shortens the time needed for familiarization and reduces the workload of the support.

14.2.2 Function block types

Using function blocks, it is possible to encapsulate complex tasks so they can be used again and to address them via defined interfaces. For this, the processing can be either state-controlled or edge-controlled. Furthermore, a difference is made between tasks which can be processed final and tasks which require permanent intervention once they are started.

State-controlled If a function block always repeats its task after switch-on after it has reached a defined state, this function block processes "state-controlled".

Edge-controlled If a function block fulfills its task exactly once after being switched on, this function block processes "edge-controlled".

Final processing If a block can be finally process its order and afterwards stop to work, this is called "final processing".

Permanent processing If a block can never conclude its order definitively but continuous intervention is necessary, this is defined as "permanent processing".

In combination, there are four function block types possible:

Control	Processing	Example
State-controlled (Enable)	Final (Done)	Continuous reading of a parameter If reading is completed, the function block has completed its task. The function block repeats this task as long as the control input (Enable) remains set, e.g. MB_ReadParameter
State-controlled (Enable)	Permanent (In….)	Terminable control If the control is active, it processes until the control input (Enable) is reset and thus disabled the control
Edge-controlled (Execute)	Final (Done)	One-time writing of a parameter If writing is completed, the function block has completed its task. A new edge is required at the control input (Execute) to start writing again, e.g. MB_Write-Parameter
Edge-controlled (Execute)	Permanent (In….)	Non-terminable control Once the control is started via the control input (Execute), it cannot be disabled via the interfaces of this function block, e.g. MC_MoveVelocity

Tab. 14-6: Overview on function block types

14.2.3 Block types and their interfaces

Inputs In order to mark whether a function block processes state-controlled or edge-controlled, two different variable names are used for the inputs for the activation of the function block.

- Enable = state-controlled
- Execute = edge-controlled

Outputs To mark whether a function block processes finally or permanently, different variable names are used for the outputs indicating the processing state.



- Done = Final processing
- In….. = Permanent processing (e.g., InOperation, InSync, InVelocity, InGear, InTorque…)

The optional "Active" output can be used with final and permanent processing function blocks. However, the behavior differs!

All outputs for the error identification (Error, ErrorID, ErrorIdent) can be found with final and permanent processing function blocks. Both show the same behavior.

Standardized identifiers

Identifier	Description
Inputs	
Enable	Enable input of state-controlled function blocks
Execute	Enable input of edge-controlled function blocks. Other function blocks can interrupt this function block
ExecuteLock	Enable input of edge-controlled function blocks. As long as this input is TRUE, the function block cannot be interrupted by others
Outputs	
Active	Output marking the processing time
Done	Processing completed successfully and the data outputs are valid
In……	The output signals that the function block is processing its task and that the data outputs are valid
Shutdown (optional)	The output signals that the function block is currently reaching a defined final state (e.g., it stops axes, releases resources…). The function block has to be called until "Shutdown" becomes FALSE
CommandAborted	Output indicates that the function block was interrupted (e.g. by another function block)
Error	Process completed with error
ErrorID	Output for rough error classification
ErrorIdent	Output for detailed error classification

Tab. 14-7: Overview on standardized identifiers

14.2.4 Inputs and outputs of state-controlled function blocks, final processing

Control/ processing	I/O	Variable name	Description
State-controlled/final	I	Enable	The input variables are registered with a positive edge of "Enable". New input values are only applied at another positive edge at "Enable". "Enable" has to be TRUE during the complete function block processing! If "Enable" is disabled, processing is interrupted and the "Done", "Active", "CommandAborted" and "Error" outputs are set to FALSE. If required, "Shutdown" signals that further signals are necessary to make the function block to reach its final state defined
	O	Done	If "Done" is TRUE, the function block has successfully completed its job and is in a final state. Data outputs are now valid. "Active", "Error" and "CommandAborted" are FALSE! "Done" remains TRUE for exactly one PLC cycle. If, afterwards, "Enable" is still TRUE, the function block starts its processing again
	O	Active	If "Active" is TRUE, the function block is working on its actual task and is in an intermediate state. Possible preprocessings are not flagged with this output! "Done", "Error" and "CommandAborted" are FALSE!
	O	Shutdown (optional)	If a "Shutdown" output is available, the function block requires multiple cycles of computation time to disable "Enable" to reach a defined final state and to release the resources used. The function block has to be called until "Shutdown" becomes FALSE!
	O	CommandAborted	If "CommandAborted" is TRUE, the function block was interrupted and is in a final state. "Done", "Active" and "Error" are FALSE! "CommandAborted" remains TRUE until the control input "Enable" is disabled. An additional reset input is therefore not necessary. If a "Shutdown" output is available, it becomes simultaneously TRUE with "CommandAborted". In this case, the function block has to be called until "Shutdown" becomes FALSE
	O	Error	If "Error" is TRUE, the function block was aborted due to an error and is in a final state. "Error" remains TRUE until the "Enable" control input is deactivated. An additional reset input is therefore not necessary. If a "Shutdown" output is available, it becomes simultaneously TRUE with "Error". In this case, the function block has to be called until "Shutdown" becomes FALSE

Tab. 14-8: I/O of state-controlled function blocks, final processing

☞ If it is required that certain inputs are not only applied at the edge at "Enable", but cyclically during the function block processing, it is to be documented explicitly and not to be specified via the variable name!

☞ For function blocks with "Shutdown", define the behavior of the function block when "Enable" is set to TRUE while "Shutdown" remains set (retrigger).

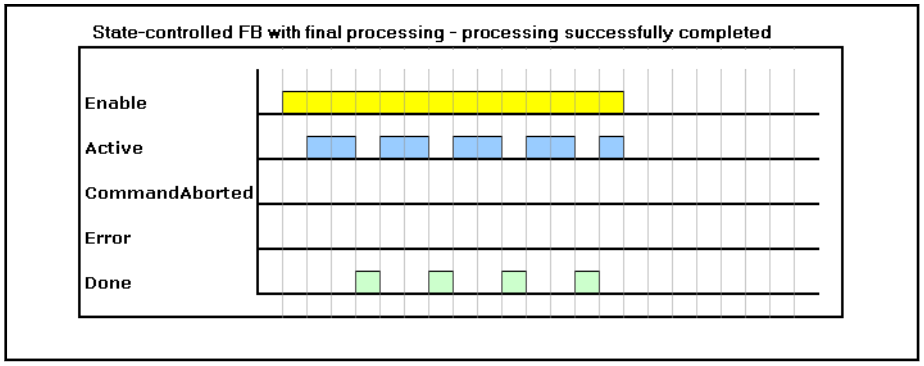


Fig. 14-1: I/O of state-controlled function blocks, final processing without "Shutdown". Processing successfully completed

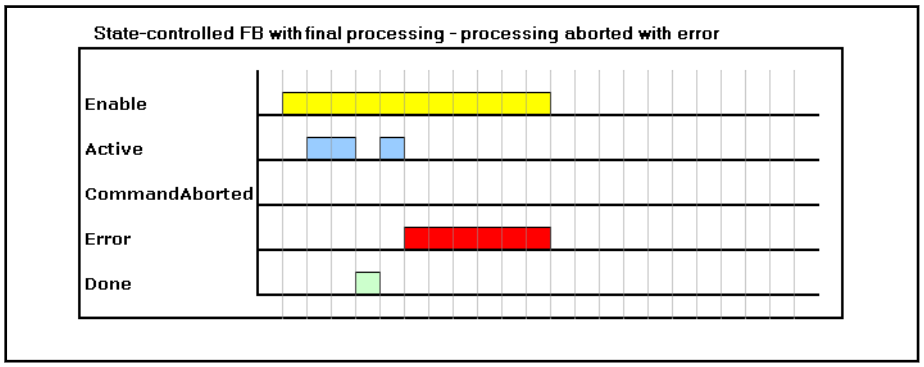


Fig. 14-2: I/O of state-controlled function blocks, final processing without "Shutdown". Processing completed with error

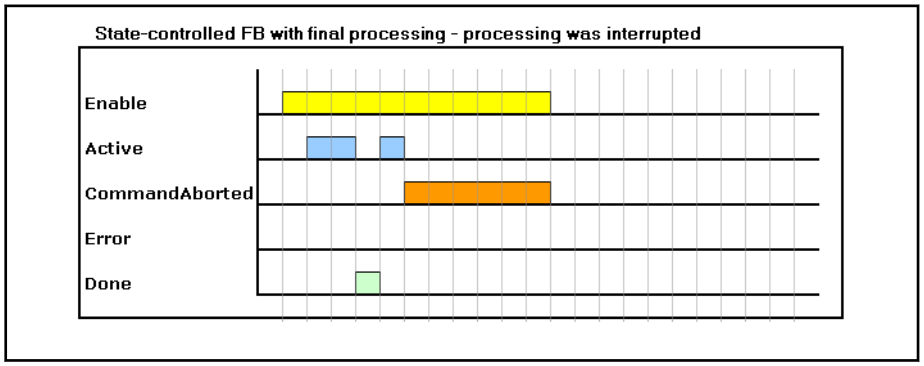


Fig. 14-3: I/O of state-controlled function blocks, final processing without "Shutdown". Processing interrupted

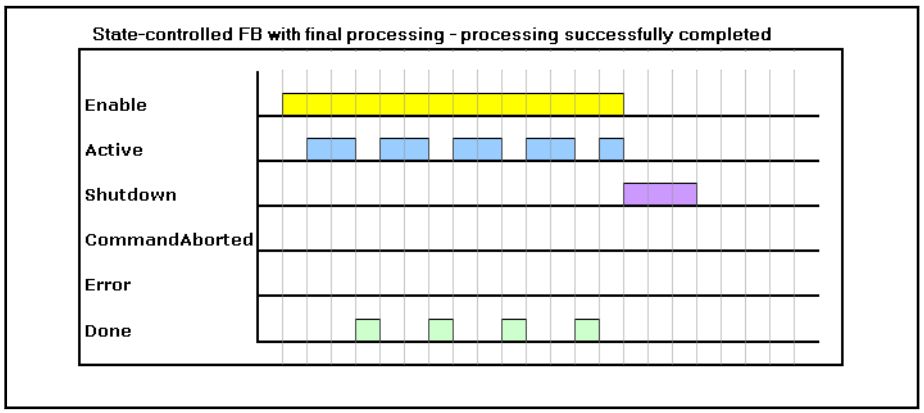


Fig. 14-4: I/O of state-controlled function blocks, final processing with "Shutdown". Processing successfully completed

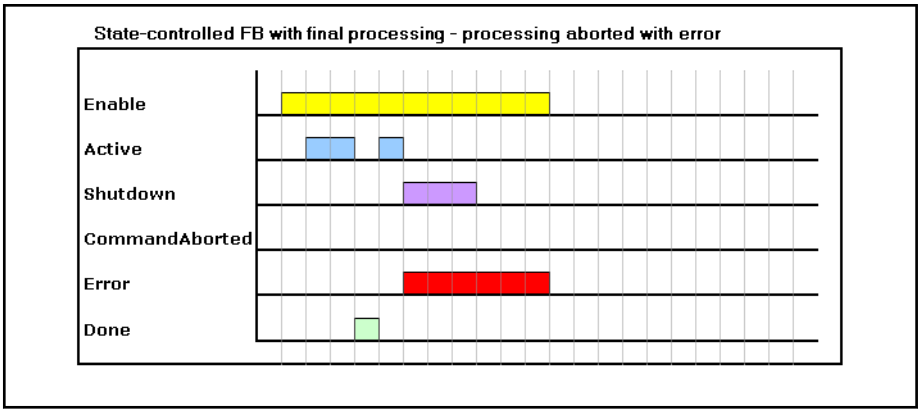


Fig. 14-5: I/O of state-controlled function blocks, final processing with "Shutdown". Processing completed with error

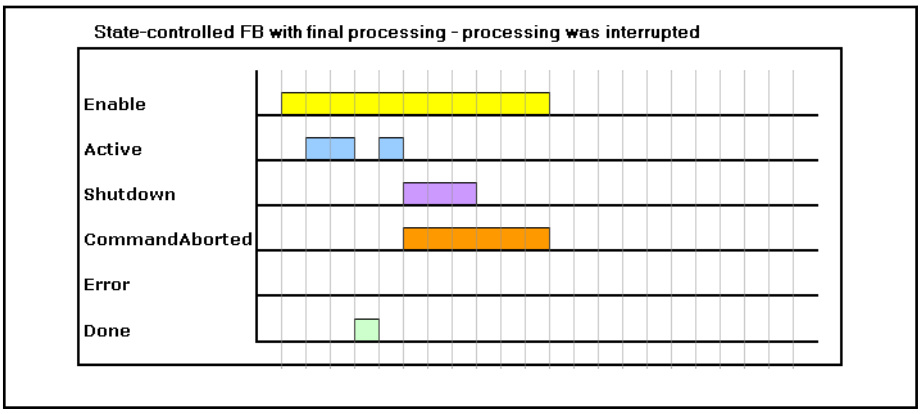


Fig. 14-6: I/O of state-controlled function blocks, final processing with "Shutdown". Processing was interrupted

14.2.5 Inputs and outputs of edge-controlled function blocks, final processing

Control/ processing	I/O	Name	Description
Edge-controlled/final	I	Execute	The input variables are registered with a positive edge of "Execute". New input values are only applied at another positive edge at "Execute". The change of edges at "Execute" suffices to start the block. For further processing, the state of "Execute" is not important. With a new edge change during the processing, the previous task is discarded, the inputs are applied again and the task is continued with the new values (retriggering, subsequent triggering). If a "Shutdown" output is available, retriggering is perhaps not supported. The function block documentation provides clarity on this issue
	O	Done	If "Done" is TRUE, the function block has successfully completed its job and is in a final state. Data outputs are now valid. "Active", "Error" and "CommandAborted" are FALSE! If the control input "Execute" is FALSE when the order is completed, "Done" remains TRUE for exactly one PLC cycle. If "Execute" is TRUE, "Done" remains TRUE until "Execute" is disabled. If a "Shutdown" output is available, it becomes simultaneously TRUE with "Done". In this case, the function block has to be called until "Shutdown" becomes FALSE
	O	Active	If "Active" is TRUE, the function block is working on its actual task and is in an intermediate state. Possible preprocessings are not flagged with this output! "Done", "Error", "Shutdown" and "CommandAborted" are FALSE!
	O	Shutdown (optional)	If a "Shutdown" output is available, the function block requires multiple cycles of computation time after the completion of the task (signaled via "Done") to reach a defined final state and to release the resources used. The function block has to be called until "Shutdown" becomes FALSE!
	O	CommandAborted	If "CommandAborted" is TRUE, the function block was interrupted and is in a final state. "Done", "Active" and "Error" are FALSE! If the control input "Execute" is FALSE at the time when the order is interrupted, "CommandAborted" remains TRUE for exactly one PLC cycle. If "Execute" is TRUE, "CommandAborted" remains TRUE until "Execute" is disabled. If a "Shutdown" output is available, it becomes simultaneously TRUE with "CommandAborted". In this case, the function block has to be called until "Shutdown" becomes FALSE
	O	Error	If "Error" is TRUE, the function block was interrupted due to an error and is now in a final state. If the control input "Execute" is FALSE when the error occurs, "Error" remains TRUE for exactly one PLC cycle. If "Execute" is TRUE, "Error" remains TRUE until "Execute" is disabled. If a "Shutdown" output is available, it becomes simultaneously TRUE with "Error". In this case, the function block has to be called until "Shutdown" becomes FALSE

Tab. 14-9: I/O of state-controlled function blocks, final processing

☞ If certain inputs are not only applied with the 0/1 edge at "Execute", but cyclically during the function block processing, it is to be documented explicitly and not to be specified via the variable name!

For function blocks with "Shutdown", define the behavior of the function block when a new edge is detected at "Execute" while "Shutdown" remains set (retrigger).

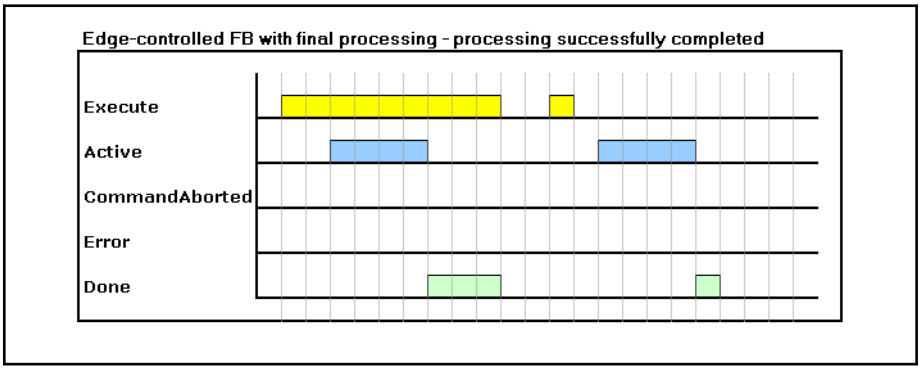


Fig. 14-7: I/O of edge-controlled function blocks, final processing without "Shutdown". Processing successfully completed

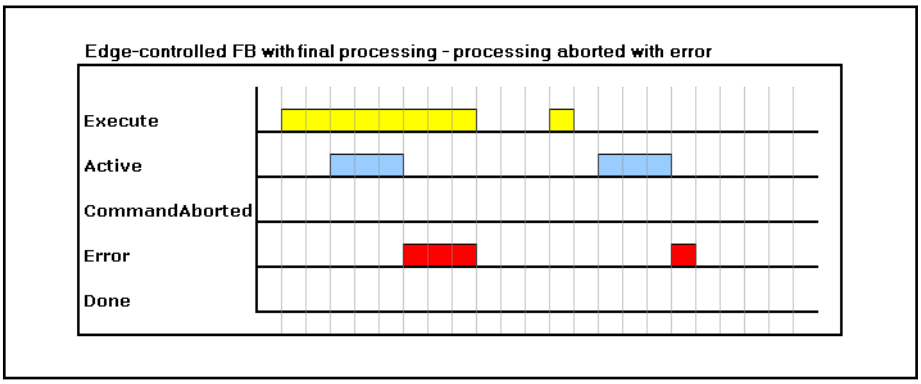


Fig. 14-8: I/O of edge-controlled function blocks, final processing without "Shutdown". Processing completed with error

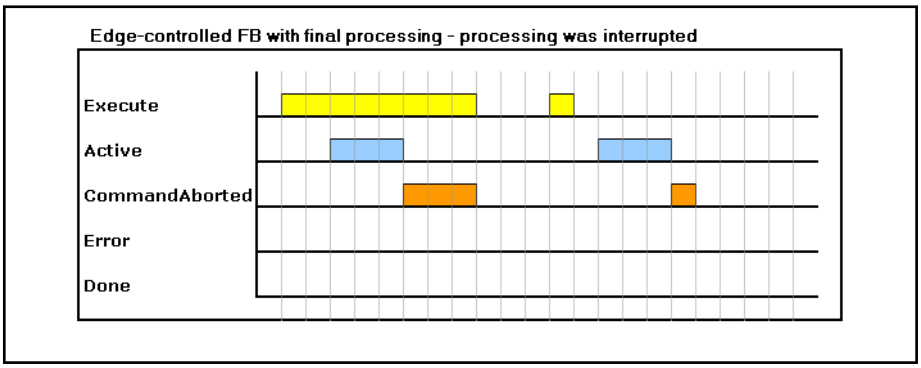


Fig. 14-9: I/O of edge-controlled function blocks, final processing without "Shutdown". Processing interrupted

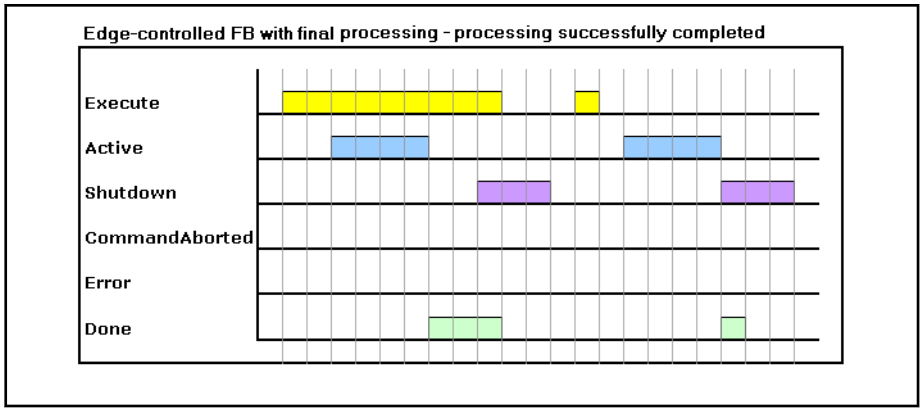


Fig. 14-10: I/O of edge-controlled function blocks, final processing with "Shutdown". Processing successfully completed

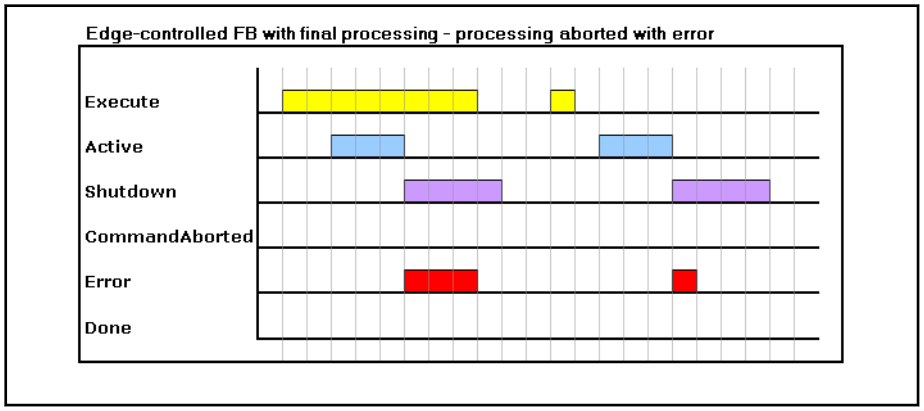


Fig. 14-11: I/O of edge-controlled function blocks, final processing with "Shutdown". Processing completed with error

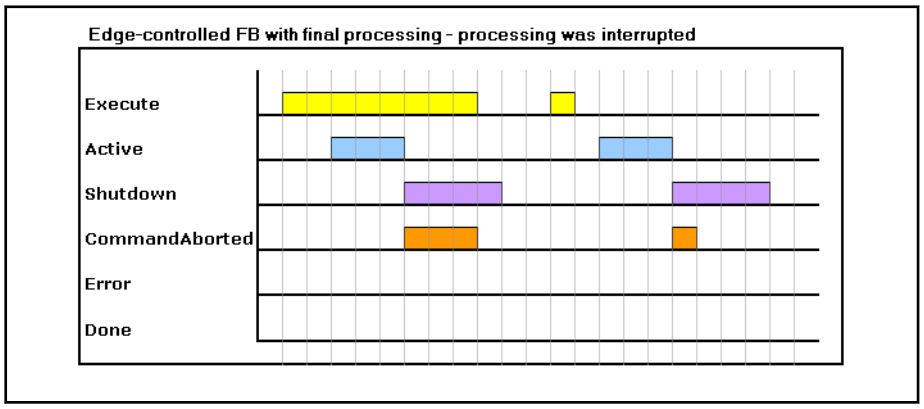


Fig. 14-12: I/O of edge-controlled function blocks, final processing with "Shutdown". Processing interrupted

14.2.6 Inputs and outputs of state-controlled function clocks, permanent processing

Control/ processing	I/O	Variable name	Description
State-controlled/ permanent	I	Enable	The input variables are registered with a positive edge of "Enable". New input values are only applied at another positive edge at "Enable". "Enable" has to be TRUE during the complete function block processing! If "Enable" is disabled, processing is interrupted and the "In….", "Active", "CommandAborted" and "Error" outputs are set to FALSE. If required, "Shutdown" signals that further signals are necessary to make the function block to reach its final state defined
	O	In….	If "In…." TRUE, the function block has reached its goal, but is still in operation to "preserve" what has been achieved and is thus in a permanent final state. Data outputs are now valid. Since the function block remains in intervention, "Active" also has to remain TRUE. "Error" and "CommandAborted" are FALSE! As long as "Enable" is TRUE, "In...." also remains TRUE
	O	Active	If Active is TRUE, the function block processes its actual task. Possible preprocessing are not marked with this output! Since the function block remains permanently in operation, "Active" remains TRUE until the function block is switched off via the control input "Enable" or is terminated by "Error" or "CommandAborted". As long as "Active" is TRUE, "Error" or "CommandAborted" must be FALSE!
	O	Shutdown (optional)	If a "Shutdown" output is available, the function block requires multiple cycles of computation time to disable "Enable" to reach a defined final state and to release the resources used. The function block has to be called until "Shutdown" becomes FALSE!
	O	CommandAborted	If "CommandAborted" is TRUE, the function block was interrupted and is in a final state. "In...," "Active" and "Error" are FALSE! "CommandAborted" remains TRUE until the control input "Enable" is disabled. An additional reset input is therefore not necessary. If a "Shutdown" output is available, it becomes simultaneously TRUE with "CommandAborted". In this case, the function block has to be called until "Shutdown" becomes FALSE
	O	Error	If "Error" is TRUE, the function block was aborted due to an error and is in a final state. "Error" remains TRUE until the "Enable" control input is deactivated. An additional reset input is therefore not necessary. If a "Shutdown" output is available, it becomes simultaneously TRUE with "Error". In this case, the function block has to be called until "Shutdown" becomes FALSE

Tab. 14-10: I/O of state-controlled function blocks, final processing

☞ If it is required that certain inputs are not only applied at the edge at "Enable", but cyclically during the function block processing, it is to be documented explicitly and not to be specified via the variable name!

☞ For function blocks with "Shutdown", define the behavior of the function block when "Enable" is set again to TRUE while "Shutdown" remains set (retrigger).

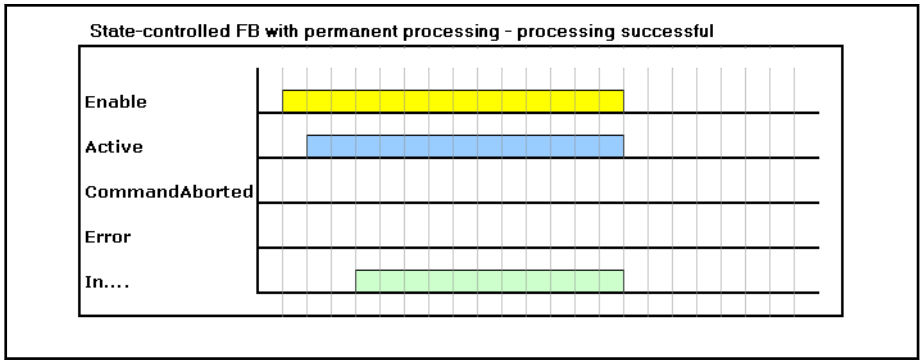


Fig. 14-13: I/O of state-controlled function blocks, permanent processing without "Shutdown". Processing successful

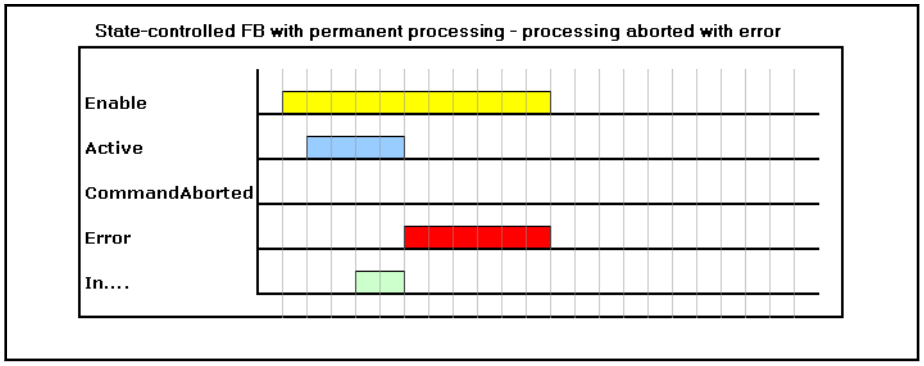


Fig. 14-14: I/O of state-controlled function blocks, permanent processing without "Shutdown". Processing aborted with errors

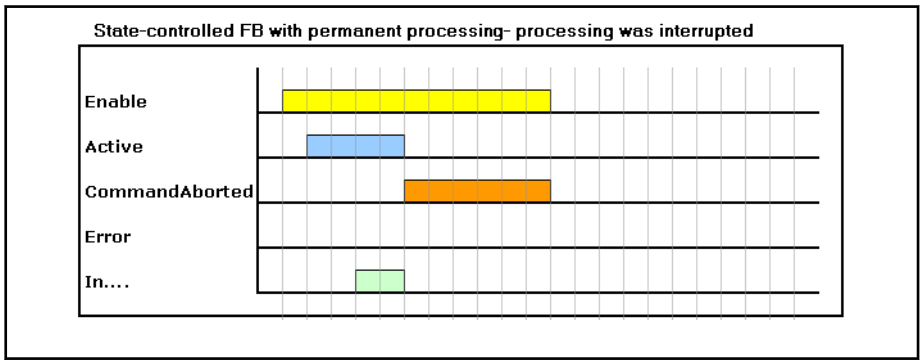


Fig. 14-15: I/O of state-controlled function blocks, permanent processing without "Shutdown". Processing interrupted

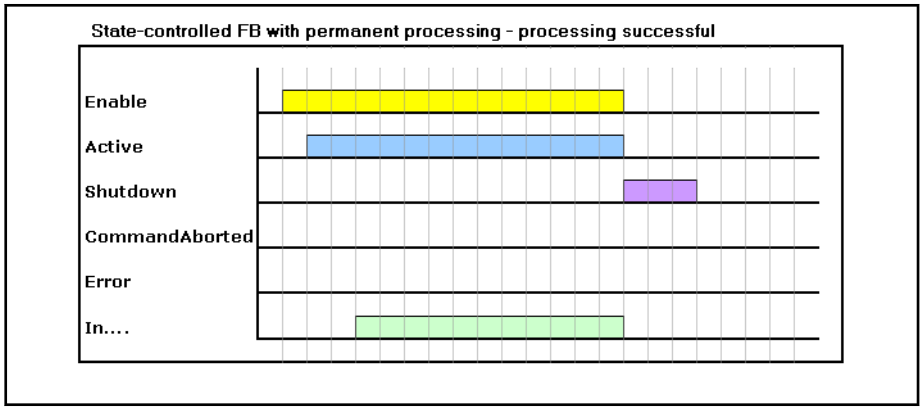


Fig. 14-16: I/O of state-controlled function blocks, permanent processing with "Shutdown". Processing successful

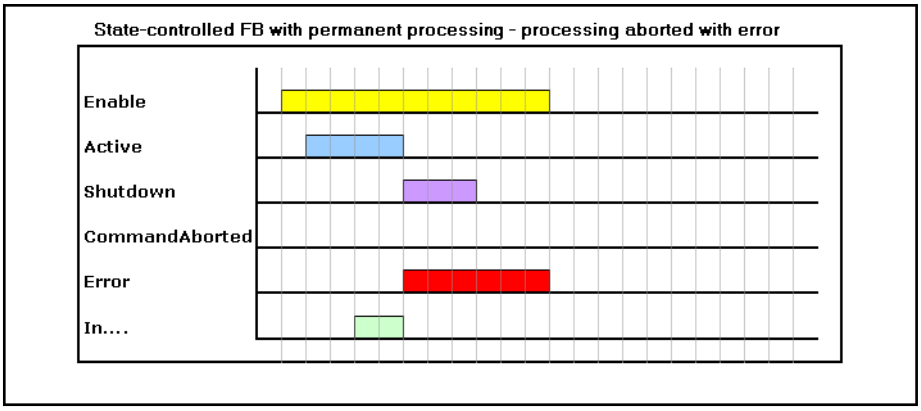


Fig. 14-17: I/O of state-controlled function blocks, permanent processing with "Shutdown". Processing aborted with errors

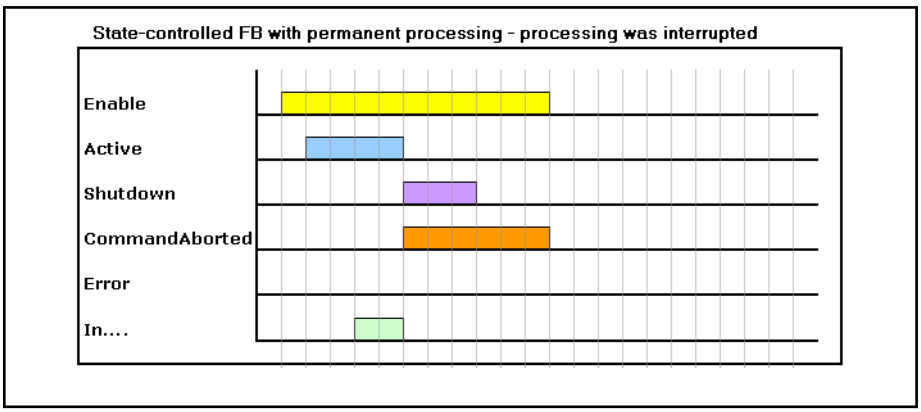


Fig. 14-18: I/O of state-controlled function blocks, permanent processing with "Shutdown". Processing interrupted

14.2.7 Inputs and outputs of edge-triggered function blocks, permanent processing

Control/ processing	I/O	Variable name	Description
Edge-controlled/ permanent	i	Execute	The input variables are registered with a positive edge of "Execute". New input values are only applied at another positive edge at "Execute". The change of edges at "Execute" suffices to start the block. For further processing, the state of "Execute" is not important. By changing the edge again during processing, the previous task is discarded, the inputs are applied again and the task is continued with the new values (retriggering, subsequent triggering). If the function block provided with a "Shutdown" output, retriggering could not be allowed
	O	In….	If "In…." TRUE, the function block has reached its goal, but is still in operation to "preserve" what has been achieved and is thus in a permanent final state. Data outputs are now valid. Since the function block remains in intervention, "Active" also has to remain TRUE. This even applies when "In...." decreases again. "Error" an "CommandAborted" are FALSE! If the control input "Execute" is FALSE when the order is completed, "In...." remains TRUE for exactly one PLC cycle. If "Execute" is TRUE, "In…." remains TRUE until "Execute" is disabled
	O	Active	If "Active" is TRUE, the function block processes its actual task. Possible preprocessing are not marked with this output! Since the function block remains in intervention permanently, "Active" remains TRUE until the block is completed by "Error" or "CommandAborted". As long as "Active" is TRUE, "Error" or "CommandAborted" as well as "Shutdown" has to be FALSE!
	O	Shutdown (optional)	If a "Shutdown" output is available, the function block requires multiple cycles of computation time to reach a defined final state and to release the resources used after the function block has canceled its task ("CommandAborted" or "Error" = TRUE). The function block has to be called until "Shutdown" becomes FALSE!
	O	CommandAborted	If "CommandAborted" is TRUE, the function block was interrupted and is in a final state. "In...," "Active" and "Error" are FALSE! If the control input "Execute" is FALSE at the time when the order is interrupted, "CommandAborted" remains TRUE for exactly one PLC cycle. If "Execute" is TRUE, "CommandAborted" remains TRUE until "Execute" is disabled. If a "Shutdown" output is available, it becomes simultaneously TRUE with "CommandAborted". In this case, the function block has to be called until "Shutdown" becomes FALSE
	O	Error	If "Error" is TRUE, the function block was interrupted due to an error and is now in a final state. If the control input "Execute" is FALSE when the error occurs, "Error" remains TRUE for exactly one PLC cycle. If "Execute" is TRUE, "Error" remains TRUE until "Execute" is disabled. If a "Shutdown" output is available, it becomes simultaneously TRUE with "Error". In this case, the function block has to be called until "Shutdown" becomes FALSE

Tab. 14-11: I/O of edge-controlled function blocks, permanent processing

If certain inputs are not only applied with the 0/1 edge at "Execute", but cyclically during the function block processing, it is to be documented explicitly and not to be specified via the variable name!

For function blocks with "Shutdown", define the behavior of the function block when a new edge is detected at "Execute" while "Shutdown" remains set (retrigger).

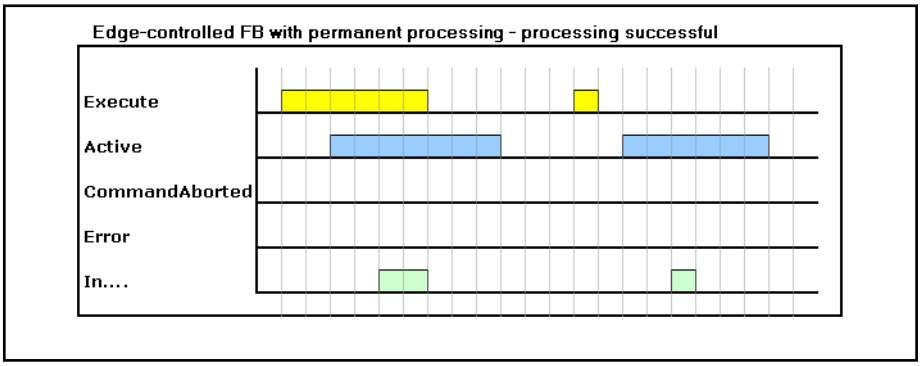


Fig. 14-19: I/O of edge-controlled function blocks, permanent processing without "Shutdown". Processing successful

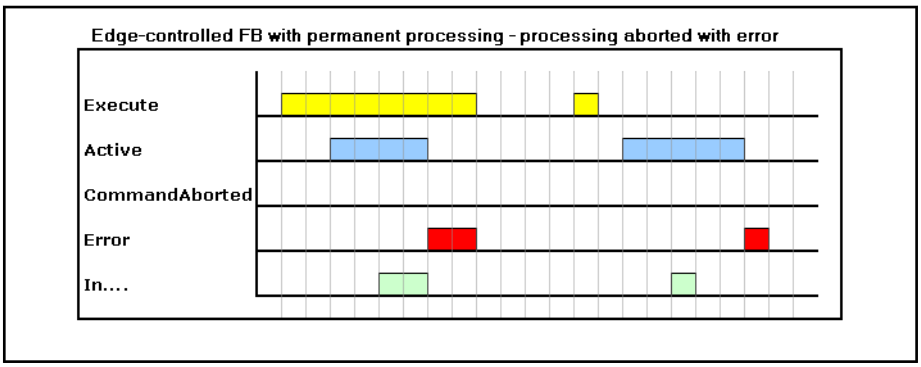


Fig. 14-20: I/O of edge-controlled function blocks, permanent processing without "Shutdown". Processing aborted with errors

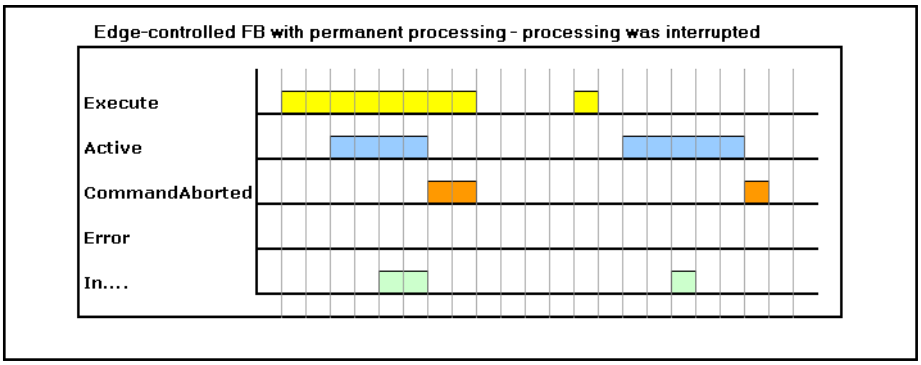


Fig. 14-21: I/O of edge-controlled function blocks, permanent processing without "Shutdown". Processing interrupted

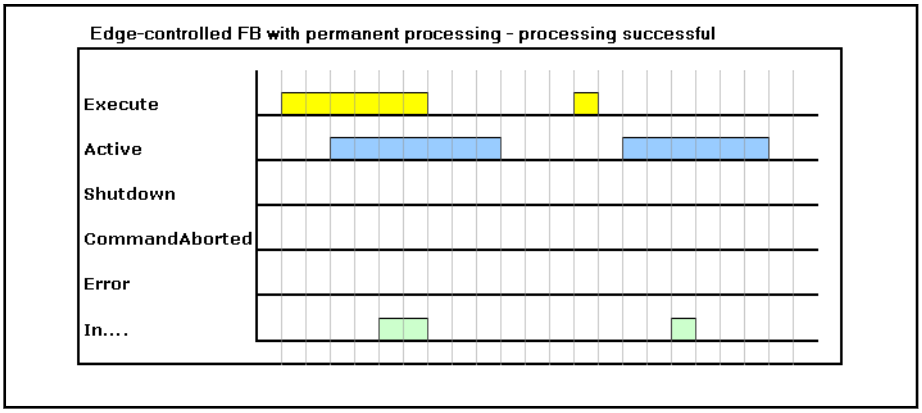


Fig. 14-22: I/O of edge-controlled function blocks, permanent processing with "Shutdown". Processing successful

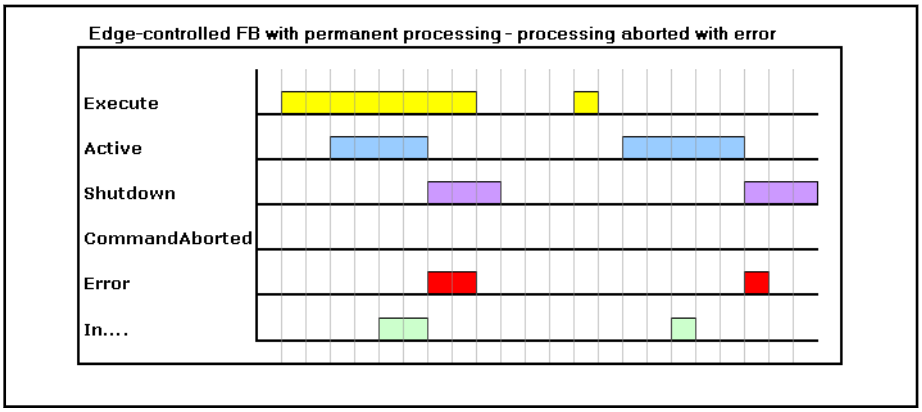


Fig. 14-23: I/O of edge-controlled function blocks, permanent processing with "Shutdown". Processing aborted with errors

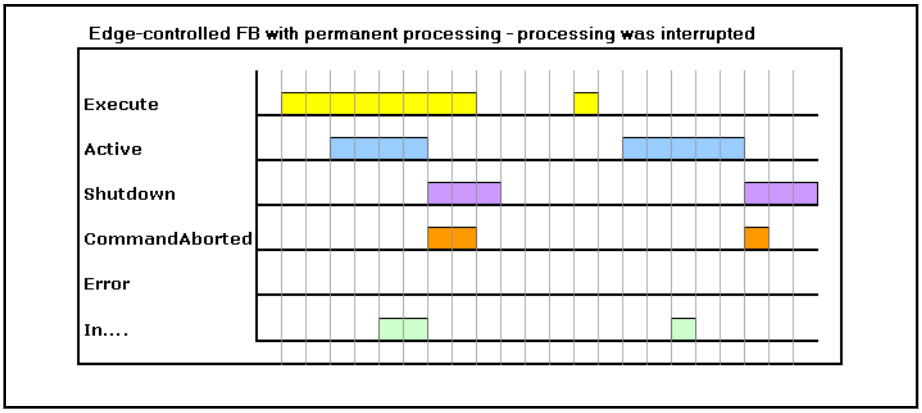


Fig. 14-24: I/O of edge-controlled function blocks, permanent processing with "Shutdown". Processing interrupted

14.3 NC channel

14.3.1 Overview

Overview on channel functions		
Category	**Function/function block**	**Brief description**
General information	MT_ChNumberValidation, page 240	Checks whether the specified channel number exists in the NC configuration
	MT_ChGetState, page 242	Determine current channel

Feed	MT_ChSetFeedOverride, page 254	Change the channel feed
	MT_GetFeedOverride, page 255	Determining the active channel feed
NC program	MT_ChSetReentryBreak-Point, page 251	NC program, saving breakpoint
	MT_ChGetActiveNcPro-gram, page 244	Determining the name of the active NC program
	MT_ChNcProgProgress, page 14.3.5	Determining current NC program progress
Operation mode	MT_ChOPMode, page 268	Determining and changing the operation modes of a channel (combines MT_ChGe-tOPMode and MT_ChSetOPMode)
	MT_ChGetOPMode, page 256	Determines the active main and secondary operation mode of a channel
	MT_ChSetOPMode, page 263	Changes the main and secondary opera-tion modes of a channel
	MT_ChSetMainOPMode, page 259	Changes the main operation modes of a channel

Tab. 14-12: Overview on channel functions

14.3.2 MT_ChNumberValidation, Checking validity of channel number

Brief description

The function blocks checks whether the specified channel number exists in the NC configuration.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-13: Library names according to the target systems

Interface description

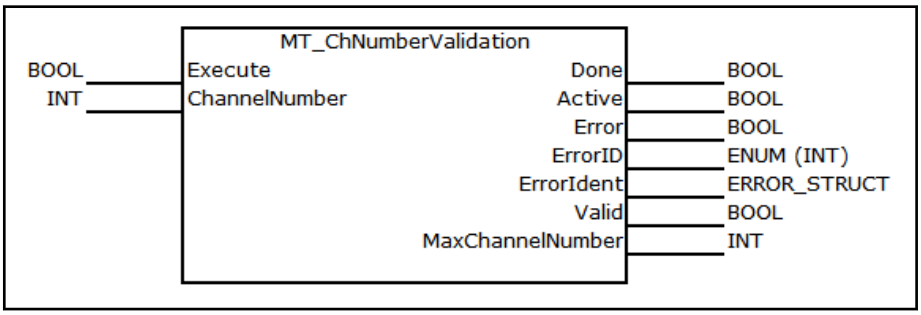


Fig. 14-25: MT_ChNumberValidation function block

	Name	Type	Description
VAR_INPUT	Execute	BOOL	Start check
	ChannelNumber	INT	Channel number to be checked

	Name	Type	Description
VAR_OUTPUT	Done	BOOL	Processing successfully completed
	Active	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "Error" = TRUE)
	Valid	BOOL	TRUE = channel number valid, FALSE = channel number invalid, value of "Valid" only valid as long as "Done"=TRUE
	MaxChannelNumber	INT	Maximum number in the NC-configured channels, value only valid as long as "Done" = TRUE

Tab. 14-14: Inputs/outputs of the MT_ChNumberValidation function block

Signal-time diagram

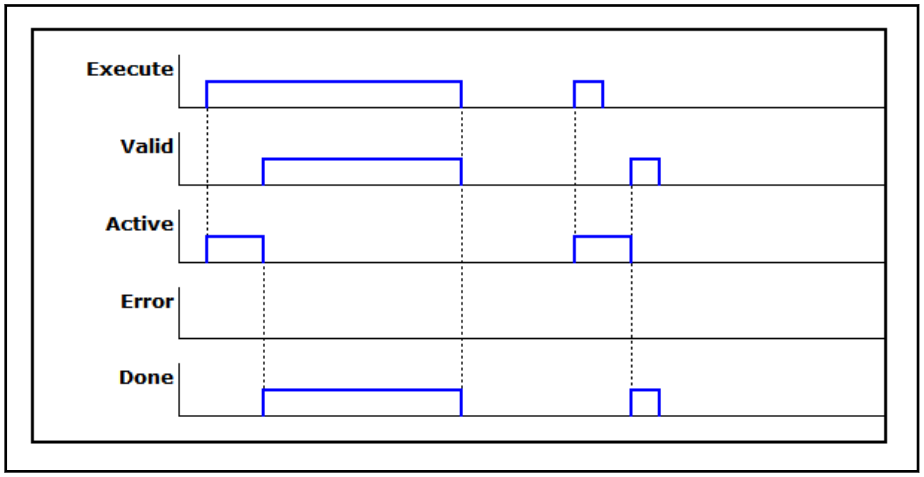


Fig. 14-26: Signal-time diagram MT_ChNumberValidation

Functional description



This function block works edge-controlled. That means that the function block executes the check exactly once after the activation.

A signal change from zero to one (0 → 1) at "Execute" enables the function block and backs up the state of the inputs at this point in time. When the "Done" output is TRUE, the channel number check is successfully completed. If "Done" is TRUE, the outputs "Valid" and "MaxChannelNumber" are valid.

If the "Valid" output is TRUE, the value at the "ChannelNumber" input is a valid channel number available in the NC configuration (MaCoDa: 904000001). If there is no correspondence in the NC configuration, the "Valid" output is FALSE.

The output "MaxChannelNumber" shows the number of channels configured in the NC.

It is displayed at the "Error" output if the check could not be made. As long as "Error" is TRUE, there is a valid error code under "ErrorIdent". Before checking again and deleting errors, "Execute" has to be set to FALSE. Resetting "Execute" deletes "Error" and "Done". If "Active" is TRUE, signal changes at

the inputs have no effect. The outputs "Done", "Active" and "Error" exclude each other. That means that only one is TRUE at a time.

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_ChNumberValidation"

Error code (hex)	Description
0x00000000	No error code available
0x00090415	MT_CfgData error. The value in Addtional2 contains the error code for MT_CfgData

Tab. 14-15: "MT_ChNumberValidation" error, from "ErrorIdent.Additional1", valid if "Error"==TRUE

Implementation example

In this example, it is checked whether number 3 is a valid channel number. The result shows that 3 is a valid channel number and that 5 would be the maximum valid channel number.

```
PROGRAM prgTestChNumberValidation
VAR
    fbChNumberValidation    :MT_ChNumberValidation;
    bExecute                :BOOL:=FALSE;   (*execute setting operation mode*)
    iChNumber               :UINT:=1;       (*channel number that should be validated*)
    bDone                   :BOOL:=FALSE;   (*process successful finished*)
    bActive                 :BOOL:=FALSE;   (*process active*)
    bError                  :BOOL:=FALSE;   (*process incorrect aborted, 'enErrorID' and 'stErrorIdent' valid*)
    enErrorID               :ERROR_CODE;    (*basic error information, only valid when 'bError'==TRUE*)
    stErrorIdent            :ERROR_STRUCT;  (*detailed error information, only valid when 'bError'==TRUE*)
    bValid                  :BOOL:=FALSE;   (*TRUE=channel number ist valid, FALSE=channel number is invalid*)
    iMaxChNumber            :UINT:=0;       (*maximum permissible channel number according to the NC-configuration*)
END_VAR
```

Fig. 14-27: Declaration of MT_ChNumberValidation

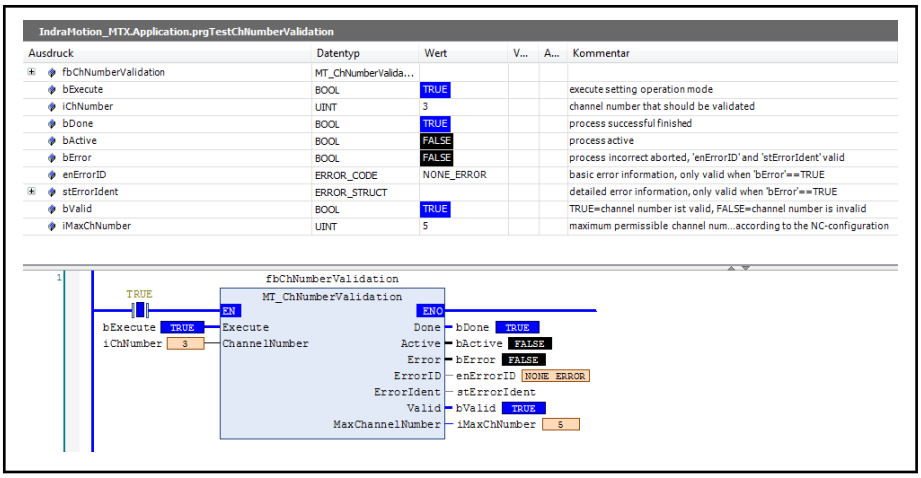


Fig. 14-28: Implementing MT_ChNumberValidation

14.3.3 MT_ChGetState, Determine current channel status

Brief description

Function to determine the current channel status.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-16: Library names according to the target systems

Interface description

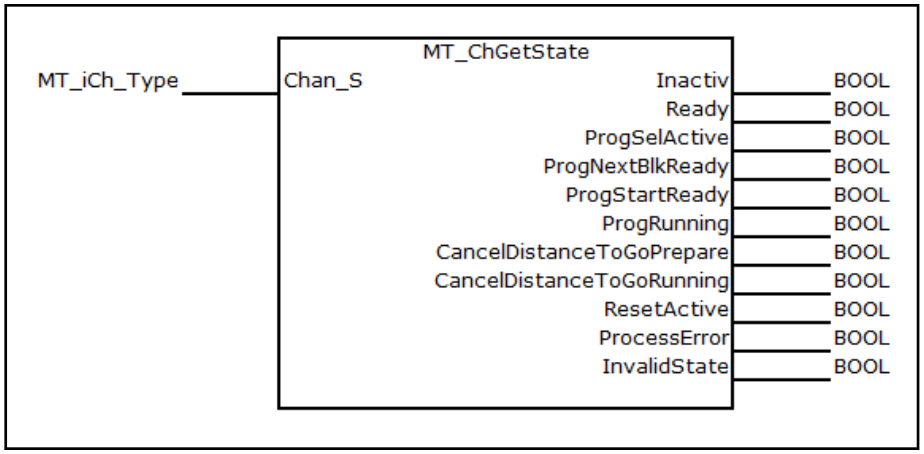


Fig. 14-29: Function block MT_ChGetState

	Name	Type	Description
VAR_INPUT	Chan_S	MT_iCH_TYPE	Channel interface: Channel-status signals
VAR_OUTPUT	Inactive	BOOL	Operation mode is not active and a process can be selected
	Ready	BOOL	Operation mode is ready and the process can be started
	ProgSelActive	BOOL	A selected program is prepared (linked)
	ProgNextBlkReady	BOOL	All blocks at the buffered NC block specification are processed. Waiting for the next specification
	ProgStartReady	BOOL	Operation mode is ready and the process is at program beginning and can be started
	ProgRunning	BOOL	Operation mode is active and executes a program or an NC block
	CancelingDistanceToGoPrepare	BOOL	"Cancel distance to go" was triggered. Cleanup continues
	CancelingDistanceToGoRunning	BOOL	"Cancel distance to go" was triggered, operation mode is active and processes existing buffers again
	ResetActive	BOOL	Control reset is being executed by the channel. No tasks may be accepted anymore until the state switches to "inactive"
	ProcessError	BOOL	An error occurred in the operation mode. This error can be eliminated only with "Control reset" or "Program deselection"
	InvalidState	BOOL	Invalid channel status, the active bit coding of "iCh_State_00..04" is unknown

Tab. 14-17: Inputs/outputs function block MT_ChGetState

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

The current channel status is transmitted as bit-coded via the (MT_iCh_Type) channel interface using the signals "iCh_State_00 .. 04" from the NC to the PLC. This function decodes the information and sets the relevant outputs of the function to display the channel status. Only one status is valid.

Troubleshooting

No error handling required for this function.

Implementation example

The operation mode of the first channel is determined in the example program.

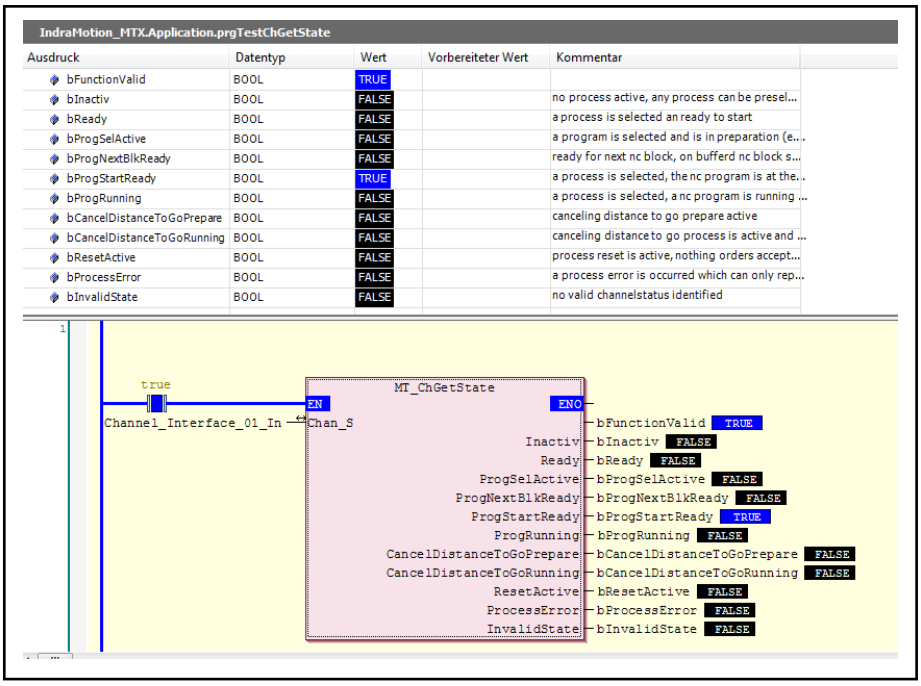


Fig. 14-30: Implementation example MT_ChGetState

14.3.4 MT_ChGetActiveProgram, Determining active NC program

Brief description

Function block to determine the active NC program.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-18: Library names according to the target systems

Interface description

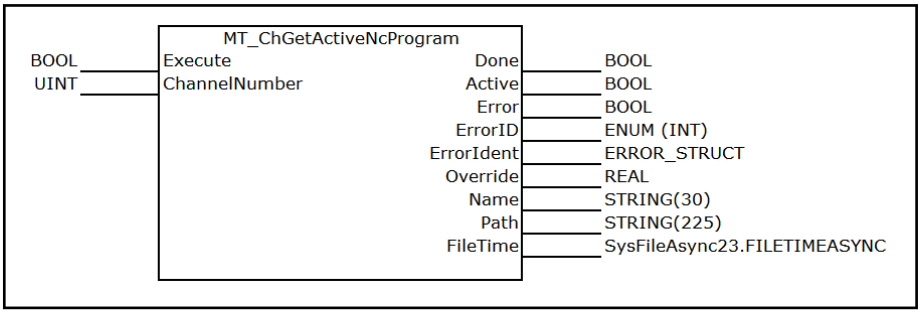


Fig. 14-31: MT_ChGetActiveNcProgram function block

	Name	Type	Description
VAR_INPUT	Execute	BOOL	Start reading the active NC program
	ChannelNumber	UINT	Channel number

	Name	Type	Description
VAR_OUTPUT	Done	BOOL	Processing successfully completed
	Active	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "Error" = TRUE)
	Name	STRING(30)	File name of the active NC program
	Path	STRING(225)	Storage path of the file of the active NC program
	FileTime.dtCreation	DATE_AND_TIME	Date and time when the file (active NC program) was created
	FileTime.dtLastAccess	DATE_AND_TIME	Date and time when the file (active NC program) was accessed last
	FileTime.dtLastModification	DATE_AND_TIME	Date and time when the file (active NC program) was changed last

Tab. 14-19: Inputs/outputs of the MT_ChGetActiveNcProgram function block

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

The function block determines the NC program currently selected in the channel. As result, the file name (Name), the storage path (Path), the creation date and time, the last access and last change (FileTime) is output.

In the "Done" output becomes TRUE, the determined values at the outputs "Name", "Path" and "FileTime" are valid.

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_ChGetActiveNcProgram"

Additional 1 (Hex)	Additional 2 (Hex)	Description
0x00000000	0x00000000	No error code available
16#00060401	MaxChannelNumber	The specified channel number is invalid. Additional 2 contains the highest channel number permitted from the NC configuration.
0x00090426	ErrorCode MT_MCoDS	Error during the internal use of the function block MT_MCoDS. Additional2 contains error code of this function block
0x00091401	ErrorCode SysFileAsync23.SysFileReadAsync	Error during the internal use of "SysFileAsync23.SysFileReadAsync" Additional 2 contains the error code of this function block

Tab. 14-20: "MT_ChGetActiveNcProgram" error codes, only valid if the output "Error" == TRUE

Implementation example

```
PROGRAM prgTestChGetActiveProgram
VAR
    fbChGetActiveProgram    :MT_ChGetActiveProgram;
    bExecute                :BOOL:=FALSE;           (*execute setting operation mode*)
    iChNumber               :UINT:=1;               (*channel number that should be validated*)
    bDone                   :BOOL:=FALSE;           (*process successful finished*)
    bActive                 :BOOL:=FALSE;           (*process active*)
    bError                  :BOOL:=FALSE;           (*process incorrect aborted, 'enErrorID' and 'stErrorIdent' valid*)
    enErrorID               :ERROR_CODE;            (*basic error information, only valid when 'bError'==TRUE*)
    stErrorIdent            :ERROR_STRUCT;          (*detailed error information, only valid when 'bError'==TRUE*)
    strName                 :STRING(30);            (*nc program name*)
    strPath                 :STRING(225);           (*path to the nc program*)
    stFileTime              :SysFileAsync23.FILETIMEASYNC;  (*nc program file status (creation, last access, last modification*)
END_VAR
```

Fig. 14-32: Declaration

IndraMotion_MTX.Application.prgTestChGetActiveProgram

Expression	Type	Value	Comment
fbChGetActiveProgram	MT_ChGetActiveProgram		
bExecute	BOOL	TRUE	execute setting operation mode
iChNumber	UINT	1	channel number that should be validated
bDone	BOOL	TRUE	process successful finished
bActive	BOOL	FALSE	process active
bError	BOOL	FALSE	process incorrect aborted, 'enErrorID' and 'stErrorIdent' valid
enErrorID	ERROR_CODE	NONE_ERROR	basic error information, only valid when 'bError'==TRUE
stErrorIdent	ERROR_STRUCT		detailed error information, only valid when 'bError'==TRUE
strName	STRING(30)	'PYRAMIDE.npg'	nc program name
strPath	STRING(225)	'/mnt/Pyramide/'	path to the nc program
stFileTime	SysFileAsync23.FILETIMEAS...		nc program file status (creation, last access, last modification
dtCreation	DATE_AND_TIME	DT#2018-8-28-11:37:58	
dtLastAccess	DATE_AND_TIME	DT#2018-8-28-11:37:58	
dtLastModification	DATE_AND_TIME	DT#2018-8-28-11:37:58	

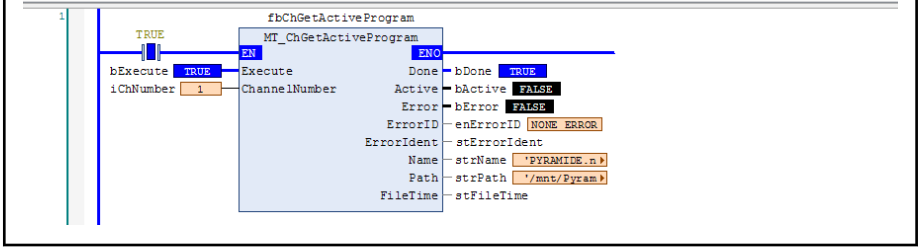


Fig. 14-33: Implementation

14.3.5 MT_ChNcProgProgress, Determining current program progress

Brief description

Function block to determine the current progress of an NC program.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-21: Library names according to the target systems

Interface description

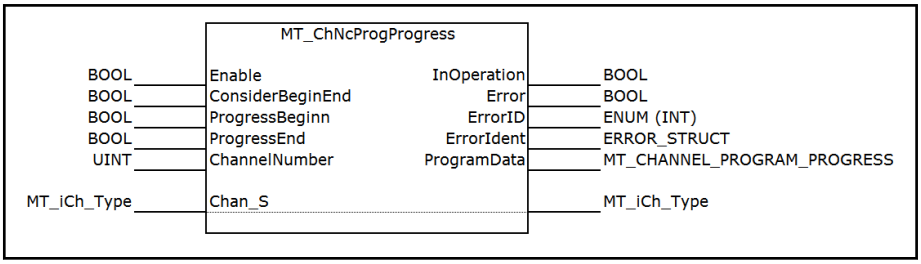


Fig. 14-34: MT_ChNcProgProgress function block

	Name	Type	Description
VAR_INPUT	Enable	BOOL	Start reading the active NC program
	ConsiderBeginEnd	BOOL	Considering external events for program start/stop
	ProgressBeginn	BOOL	External event; relevant program part was started for the progress
	ProgressEnd	BOOL	External event; relevant program part ended for the progress
	ChannelNumber	UINT	Channel number
VAR_OUTPUT	Done	BOOL	Processing successfully completed
	Active	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "Error" = TRUE)
	ProgramData.Name	STRING(30)	File name of the active NC program
	ProgramData.Path	STRING(225)	Storage path of the file of the active NC program
	ProgramData.FileTime	DATE_AND_TIME	Date and time of the file (active NC program)
	ProgramData.ProcessingTime	UDINT	Current processing time in seconds; The time the program already run
	ProgramData.RemainingTime	UDINT	Current remaining time in seconds. Time that the program will probably run with the current override
	ProgramData.TotalTimeAct	UDINT	Current total runtime in seconds. Time that the complete program will probably run with the current override
	ProgramData.TotalTimeFile	UDINT	Total runtime in seconds read from the NC program (ProgRunTimeTotal)
	ProgramData.TotalTimeFileValid	BOOL	Total runtime read from the NC program (ProgRunTimeTotal) is valid
	ProgramData.TotalTimeMeasured	UDINT	Total runtime in seconds measured by the PLC if the override is 100%
	ProgramData.TotalTimeMeasuredValid	BOOL	Measured total time is valid
	ProgramData.Progress	REAL	Program progress in percent
	ProgramData.ForecastAccuracy	UINT	Forecast accuracy 1 = Good, 2 = Medium, 3 = Bad BYTE0 = Accuracy affected by override change BYTE1 = Accuracy affected by operation mode change

Tab. 14-22: Inputs/outputs of the MT_ChGetActiveNcProgram function block

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

This function block provides information on the program progress for one channel. They can be linked to the display in the HMI header.

Basics:

The total runtime of the NC program (determined if the override is 100% without interruption) is required to compute the progress! If this information is known, it can be specified in the first line of the NC program as comment using the keyword "ProgRunTimeTotal". (e.g. ;ProgRunTimeTotal:=450s). The time has to be specified in seconds! The program progress is based on this specification. The value specified in the NC program is retrieved by the function block. It is located in the output structure in the "ProgramData.TotalTimeFile" element.

If the function block detects an uninterruptible run of the NC program at an override of 100%, this runtime is measured and subsequently used for each further program run as computation basis for the program progress. This applies as well if a time was specified in the first line of the NC program. The program runtime measured by the PLC is displayed in the output structure "ProgramData.TotalTimeMeasured" and is only valid if the value "ProgramData.TotalTimeMeasuredValid" is TRUE.

Cache:

Normally, this measuring result is lost when deselecting the NC program. However, the cache can be enabled containing all information on an NC program. The size of the array "MT_NC_PROGRAM_PROGRESS_CACHE" has to be increased to 1 in the Global Variable List "pMtChannel". (library manager, MT_BaseAdditional library)

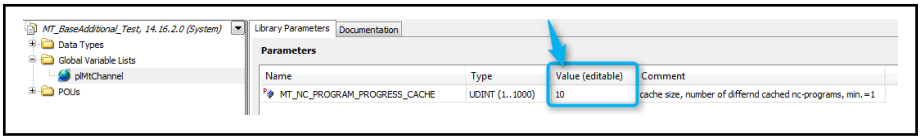


Fig. 14-35: A cache of 10 for example can save data of up to 10 different NC programs. If the cache is full, the NC program used the earliest, is removed from the cache and replaced by the newly selected one.

Cache requires memory! 381 bytes are required for each cache space.

Use:

When the function block was enabled by setting the "Enable" input, the active program is continuously determined from the channel specified at the "ChannelNumber" input. As soon as the NC program was started, the current total runtime (ProgramData.TotalTimeAct), the remaining time (ProgramData.RemainingTime) as well as the progress (ProgramData.Progress) are updated permanently in the output structure.

The function block detects program start and end.

If additional criteria for the start and the end of the program progress display are to be considered, use the inputs "ProgressBegin" and "ProgressEnd". The function block can consider these inputs only if the input "ConsiderBeginEnd" is TRUE as well when enabling the function block via "Enable".

Thus, it is possible to consider only a part of the NC program for the progress display by linking respective M-functions with the function block inputs.

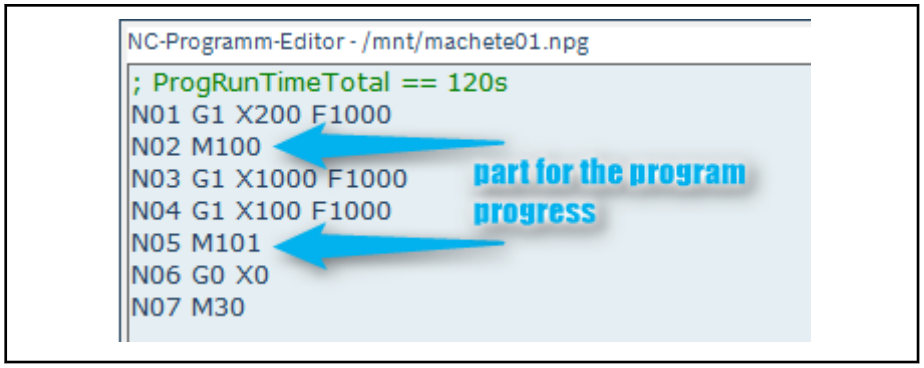


Fig. 14-36: Example: NC program with M-function, considering a progress section for the progress display

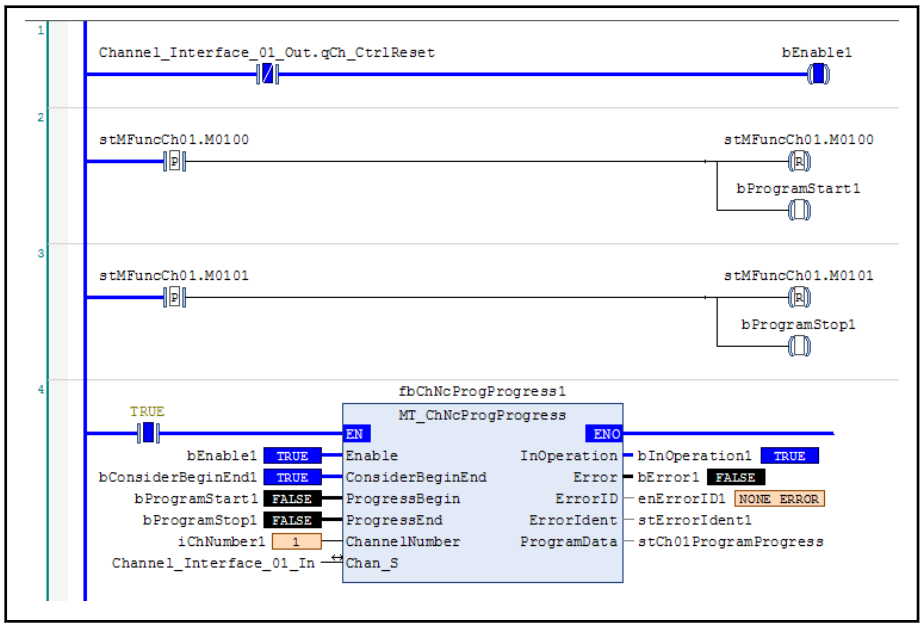


Fig. 14-37: Example: Implementation of the function block considering the M-functions

HMI, display in the header of IndraWorks Operation

There is a progress display in the header of IndraWorks Operation The function block provides data for this display. The variable connected to the function block output "ProgramData" has to have the following name "stCh**XX**ProgramProgramProgress" (XX has to be replaced by the channel number, e.g. stCh01ProgramProgress). Furthermore, this variable has to be provided with the namespace "HmiHeader" and included into the symbol configuration.

```
19      {attribute 'namespace' := 'HmiHeader'}
20      {attribute 'symbol' := 'read'}
21      stCh01ProgramProgress       :MT_CHANNEL_PROGRAM_PROGRESS; //result progress calculation
```

Fig. 14-38: Declaration for channel 01 with display in IndraWorks Operation

For more details on the display in the header, refer to the HMI documentation "Multitouch".

Accuracy

The computed values are only based on time specifications. The override is included in the computations, but changed accelerations (or similar values) are not included. Thus, data computed by the function block becomes more

inexact the more it deviates from the state from which the program runtime was determined originally (100% override, operation mode “Automatic continuous block”, no interruption). To show detected inaccuracies, the function block predicts the accuracy of the computed values via the "ProgramData.ForecastAccuracy" output. As mainly changes at the override and the operation mode affect the accuracy of the progress display, an accuracy forecast of this output is mapped based on the override in LowByte and operation mode changes in the HighByte.

Significance of the values in HighByte and LowByte:

- 1 = high accuracy of the progress display
- 2 = medium accuracy of the progress display
- 3 = low accuracy of the progress display

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_ChNcProgProgress"

Additional1 (hex)	Additional2 (hex)	Description
0x00000000	0x00000000	No error code available
16#00060401	MaxChannelNumber	The specified channel number is invalid. Additional2 contains the highest channel number permitted from the NC configuration.
0x00090426	ErrorCode MT_MCoDS	Error during the internal use of the function block MT_MCoDS. Additional2 contains error code of this function block
0x00091401	ErrorCode SysFileAsync23.SysFileReadAsync	Error during the internal use of "SysFileAsync23.SysFileReadAsync" Additional2 contains the error code of this function block

Tab. 14-23: "MT_ChNcProgProgress" error codes, only valid if the output "Error" == TRUE

Implementation example

```
PROGRAM prgTestChNcProgProgress
VAR
    //CHANNEL 01 progress bar FB
    //-------------------------------------------------------------------------------
    fbChNcProgProgress1         : MT_ChNcProgProgress;
    bEnable1                    : BOOL:=FALSE;  //enable program progress
    bConsiderBeginEnd1          : BOOL:=FALSE;  //TRUE==fb is waiting for inputs "ProgressBegin/-End", FALSE==fb consider only channel states
    bProgramStart1              : BOOL:=FALSE;  //command that nc-program was started
    bProgramStop1               : BOOL:=FALSE;  //command that nc-program was stoped
    iChNumber1                  : UINT:=1;      //channel number
    bInOperation1               : BOOL:=FALSE;  //progress calculation in work
    bError1                     : BOOL:=FALSE;  //process incorrect aborted, 'enErrorID' and 'stErrorIdent' valid
    enErrorID1                  : ERROR_CODE;   //basic error information, only valid when 'bError'==TRUE
    stErrorIdent1               : ERROR_STRUCT; //detailed error information, only valid when 'bError'==TRUE
    {attribute 'namespace' := 'HmiHeader'}
    {attribute 'symbol' := 'read'}
    stCh01ProgramProgress       :MT_CHANNEL_PROGRAM_PROGRESS; //result progress calculation

    //CHANNEL 02 progress bar FB
    //-------------------------------------------------------------------------------
    fbChNcProgProgress2         : MT_ChNcProgProgress;
```

Fig. 14-39: Declaration

IndraMotion_MTX.Application.prgTestChNcProgProgress

Ausdruck	Datentyp	Wert	Kommentar
fbChNcProgProgress1	MT_ChNcProgProgress		CHANNEL 01 progress bar FB
bEnable1	BOOL	TRUE	enable programprogress
bConsiderBeginEnd1	BOOL	FALSE	
bProgramStart1	BOOL	FALSE	command that nc-program was started
bProgramStop1	BOOL	FALSE	command that nc-program was stoped
iChNumber1	UINT	1	channel number
bInOperation1	BOOL	TRUE	progress calculation in work
bError1	BOOL	FALSE	process incorrect aborted, 'enErrorID' and 'stErrorIdent' valid
enErrorID1	ERROR_CODE	NONE_ERROR	basic error information, only valid when 'bError'==TRUE
stErrorIdent1	ERROR_STRUCT		detailed error information, only valid when 'bError'==TRUE
stCh01ProgramProgress	MT_CHANNEL_PROGRAM_P...		result progress calculation
FileName	STRING(30)	'machete10.npg'	nc-program file name
FilePath	STRING(225)	'/mnt/'	nc-program path to the file
FileTime	SysFileAsync23.FILETIMEAS...		nc-program timestamp (creation, last access, last modificatic
ProcessingTime	UDINT	10	actual program processing time in s
RemainingTime	UDINT	49	actual program remaining time in s
TotalTimeAct	UDINT	59	actual program total time in s
TotalTimeFile	UDINT	65	active program total time, read from file
TotalTimeFileValid	BOOL	TRUE	active program total time, read from file are valid
TotalTimeMeasured	UDINT	59	active program total time, measured from plc
TotalTimeMeasuredValid	BOOL	TRUE	active program total time, measured from plc are valid
Progress	REAL	16.94915	actual program progress
ForecastAccuracy	UINT	257	LowByte Override == 1=high, 2=...l, 3=low / HighByte OpM
fbChNcProgProgress2	MT_ChNcProgProgress		CHANNEL 02 progress bar FB

Fig. 14-40:

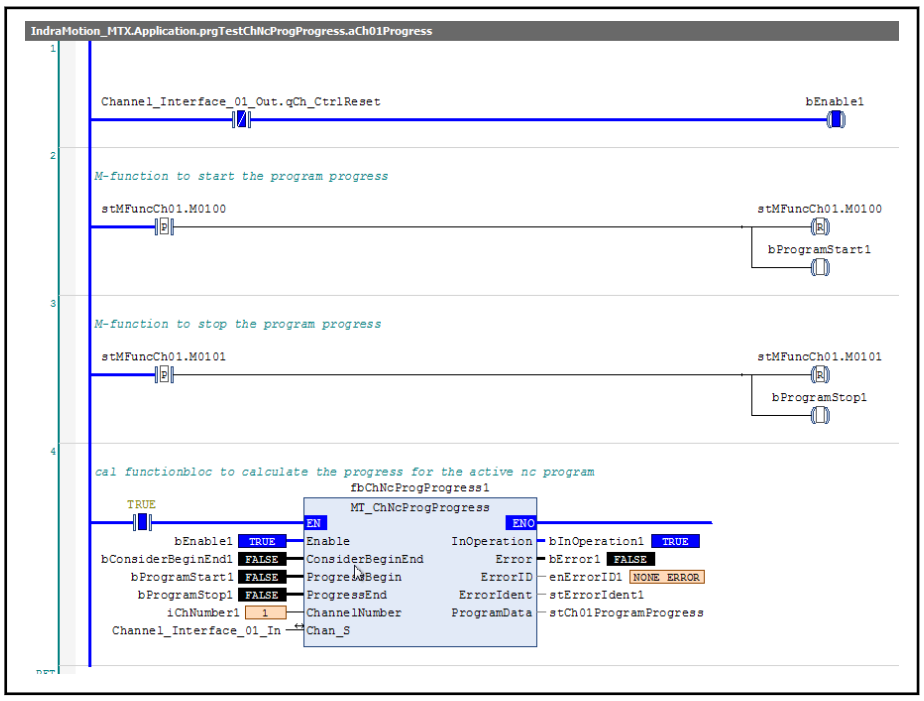


Fig. 14-41: Implementation

14.3.6 MT_ChSetReentryBreakPoint, Saving breakpoint for re-entry

Brief description

The function block enables the saving of the breakpoint in the NC. It can be used for a later re-entry.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-24: Library names according to the target systems

Interface description

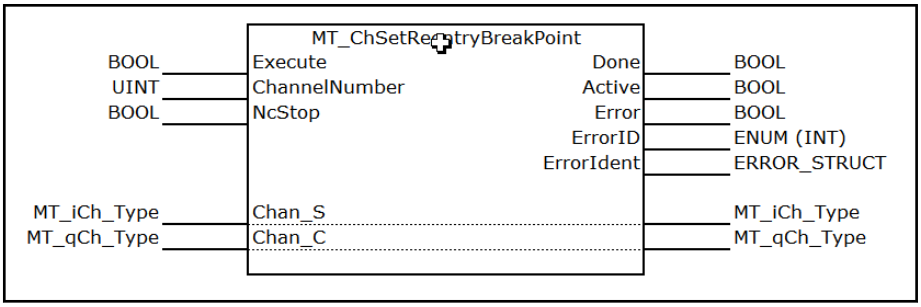


Fig. 14-42: MT_ChSetReentryBreakPoint function block

	Name	Type	Description
VAR_INPUT	Execute	BOOL	Start “Saving breakpoint”
	ChannelNumber	INT	Channel number
	NcStop	BOOL	Execute NcStop execute before “Saving breakpoint” starts
VAR_OUTPUT	Done	BOOL	Processing successfully completed
	Active	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "Error" = TRUE)
VAR_IN_OUT	Chan_S	MT_iCh_Type	Channel interface: Channel-status signals
	Chan_C	MT_qCh_Type	Channel interface: Channel-control signals

Tab. 14-25: Inputs/outputs of the MT_ChSetReentryBreakPoint function block

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description



This function block works edge-controlled. That means that the function block executes the check exactly once after the activation.

The breakpoint of an NC program is saved automatically upon each channel reset as long as the value of the system variable "SD.SysRun[Ch].ReEntryStore" is 1. Use this function block if the breakpoint of an NC should not be saved upon each channel reset, but only in case of certain events (e.g. power failure).

A signal change from zero to one (0 -> 1) at "Execute" enables the function block and checks whether the saving of the breakpoint has already been enabled ("SD.SysRun[Ch].ReEntryStore=1"). If not, it is enabled, the channel is reset by setting the channel control signal "qCh_CtrlReset" and then disabled again. If the saving of the breakpoint has already been active, only a channel reset is initiated.

If the “NcStop” input is set, the function block executes an NC stop before saving the breakpoint. The channel control signals “Block transfer inhibit” (channel status signal: iCh_TransferLockAct) and “Feed hold” (channel status signal: iCh_FeedHoldAct) are set.

When the "Done" output is TRUE, saving the breakpoint is successfully completed. Otherwise, it is displayed by the "Error" output. As long as "Error" is

TRUE, there is a valid error code under "ErrorIdent". Before executing again and deleting errors, Set "Execute" to FALSE. Resetting "Execute" deletes "Error" and "Done". If "Active" is TRUE, signal changes at the inputs have no effect. The inputs "Chan_S" and "Chan_C" (VAR_IN_OUT) are an exception. They are read cyclically as long as the "Active" output is TRUE.

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_ChSetReentryBreakPoint"

Additional1 (hex)	Additional2 (hex)	Description
0x00000000	0x00000000	No error code available
0x00040201	0x0000000C	Changing the interface signal qCh_TransferLock failed. The interfaces might have been overwritten outside of the function block
0x00020201	0x0000000D	There was no feedback on the interface signal "iCh_FeedHoldAct" within 500 ms. The interface for another channel might be connected to the function block.
0x00020201	0x00000011	There was no feedback on the interface signal "iCh_Reset" within 500 ms. The interface for another channel might be connected to the function block.
16#00060401	MaxChannelNumber	The specified channel number is invalid. Additional2 contains the highest channel number permitted from the NC configuration.
0x00090415	ErrorNumber MT_CfgData	MT_CfgData error. The value in Addtional2 contains the error code for MT_CfgData
0x00090422	ErrorNumber MT_SD_RD	MT_SD_RD error. The value in Addtional2 contains the error code for MT_SD_RD
0x00090423	ErrorNumber MT_SD_WR	MT_SD_WR error. The value in Addtional2 contains the error code for MT_SD_WR

Tab. 14-26: "MT_ChSetReentryBreakPoint" error, from "ErrorIdent.Additional1", valid if "Error"==TRUE

Implementation example

```
PROGRAM prgTestChSetReentryBreakPoint
VAR
    fbChSetReentryBreakPoint: MT_ChSetReentryBreakPoint;
    bExecute                :BOOL:=FALSE;          (*execute FB*)
    bNcStop                 :BOOL:=FALSE;          (*TRUE == the FB triggers a Nc-Stop before set the reentry breakpoint*)
    uiChNumber              :UINT:=1;              (*Number of the channel to for which you want set the reentry breakpoint*)
    bDone                   :BOOL:=FALSE;          (*process successful finished*)
    bActive                 :BOOL:=FALSE;          (*process active*)
    bError                  :BOOL:=FALSE;          (*process incorrect aborted, 'enErrorID' and 'stErrorIdent' valid*)
    enErrorID               :ERROR_CODE;           (*basic error information, only valid when 'bError'==TRUE*)
    stErrorIdent            :ERROR_STRUCT;         (*detailed error information, only valid when 'bError'==TRUE*)
END VAR
```

Fig. 14-43: Declaration of MT_ChSetReentryBreakPoint

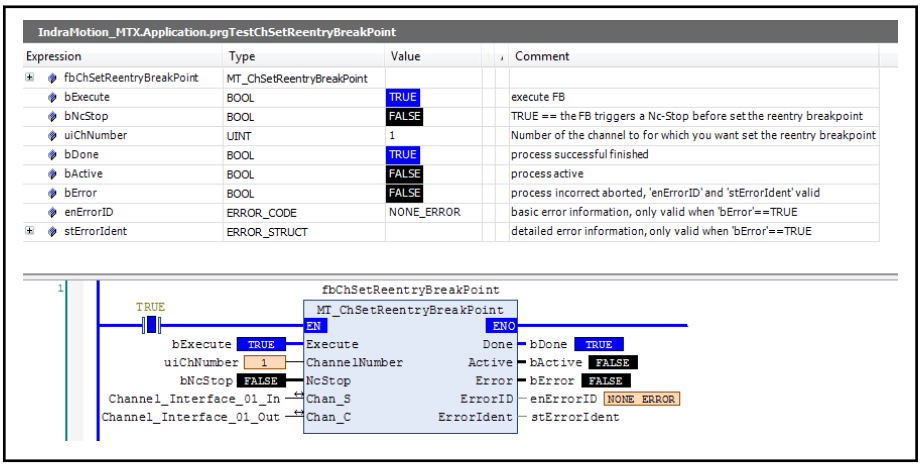


Fig. 14-44: Implementing MT_ChSetReentryBreakPoint

14.3.7 MT_ChSetFeedOverride, Change channel feed

Brief description

Function to set a feed value for a channel.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-27: Library names according to the target systems

Interface description

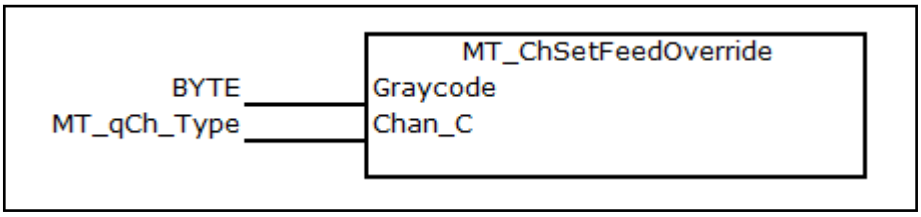


Fig. 14-45: MT_ChSetFeedOverride function block

	Name	Type	Description
VAR_INPUT	Graycode	BYTE	Override value as gray code
VAR_IN_OUT	Chan_C	MT_qCh_Type	Channel interface: Channel-control signals

Tab. 14-28: Inputs/outputs function block MT_ChSetFeedOverride

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

The channel feed value has to transmitted from the PLC to the NC via its channel interface (MT_qCh_Type) using the "qCh_Override" signal. With this function, the value at the "Graycode" input is converted and transmitted to the channel interface.

Troubleshooting

No error handling required for this function.

Implementation example

Example program for changing the override for the first channel.

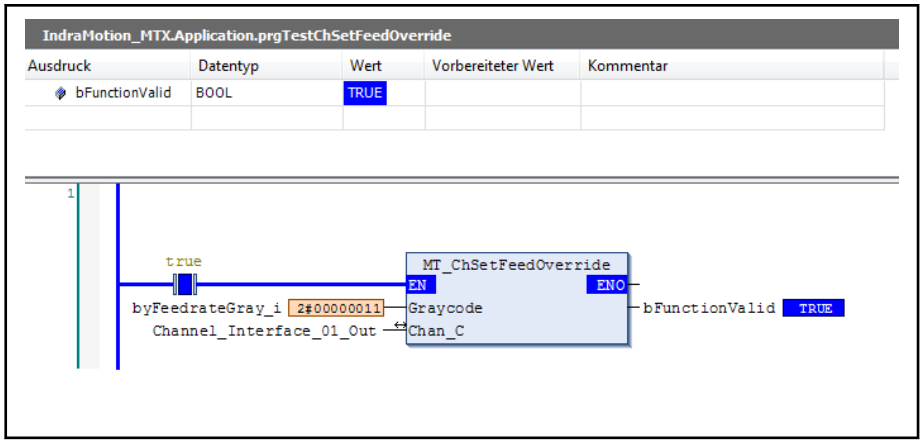


Fig. 14-46: Implementation MT_ChSetFeedOverride

14.3.8 MT_ChGetFeedOverride, Determining active feed override

Brief description

Function to determine the active feed override.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-29: Library names according to the target systems

Interface description

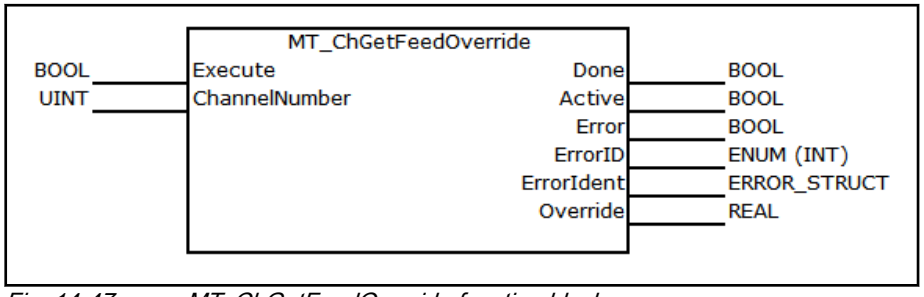


Fig. 14-47: MT_ChGetFeedOverride function block

	Name	Type	Description
VAR_INPUT	Execute	BOOL	Execute reading the feed override
	ChannelNumber	UINT	Channel number
VAR_OUTPUT	Done	BOOL	Processing successfully completed
	Active	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "Error" = TRUE)
	Override	REAL	Current override for the specified channel

Tab. 14-30: Inputs/outputs function block MT_ChGetFeedOverride

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

The function block determines the override for the channel specified at the "ChannelNumber" input by reading the function MCODS 7.

When the "Done" output is TRUE, the determined value at the "Override" output is valid.

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_ChGetFeedOverride"

Error code (hex)	Description
0x00000000	No error code available
0x00060401	Invalid input assignment. An invalid channel number was selected
0x00090426	Error during the internal use of the function block MT_MCoDS. Additional2 contains the error code of the MT_MCoDS

Tab. 14-31: "MT_GetFeedOverride" error, from "ErrorIdent.Additional1", valid if "bError" == TRUE

Implementation example

The operation mode of the first channel is determined in the example program.

```
PROGRAM prgTestChGetActiveOverride
VAR
    fbChGetActiveProgram    :MT_ChGetFeedOverride;
    bExecute                :BOOL:=FALSE;   (*execute setting operation mode*)
    iChNumber               :UINT:=1;       (*channel number that should be validated*)
    bDone                   :BOOL:=FALSE;   (*process successful finished*)
    bActive                 :BOOL:=FALSE;   (*process active*)
    bError                  :BOOL:=FALSE;   (*process incorrect aborted, 'enErrorID' and 'stErrorIdent' valid*)
    enErrorID               :ERROR_CODE;    (*basic error information, only valid when 'bError'==TRUE*)
    stErrorIdent            :ERROR_STRUCT;  (*detailed error information, only valid when 'bError'==TRUE*)
    rOverride               :REAL:=0.0;
END_VAR
```

Fig. 14-48: Declaration

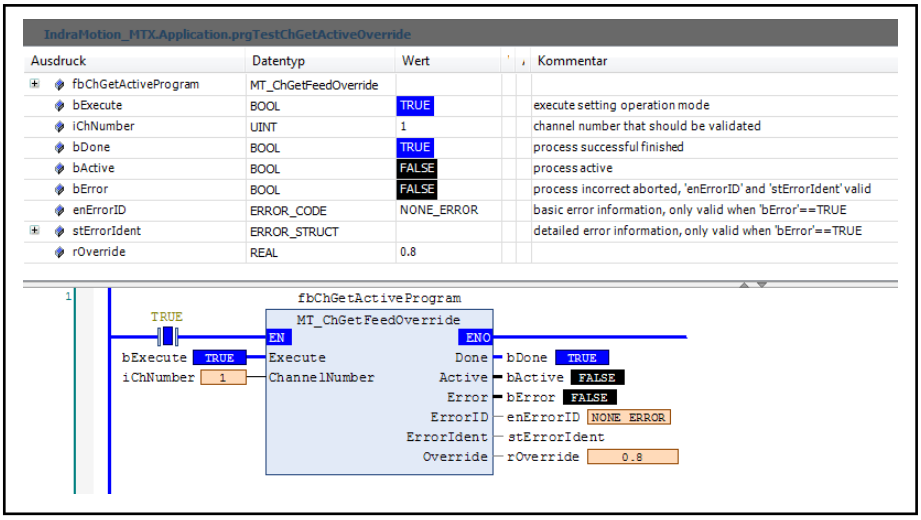


Fig. 14-49: Implementation

14.3.9 MT_ChGetOPMode, Determine active operation mode

Brief description

Function block to determine the active main and secondary operation modes.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-32: Library names according to the target systems

Interface description

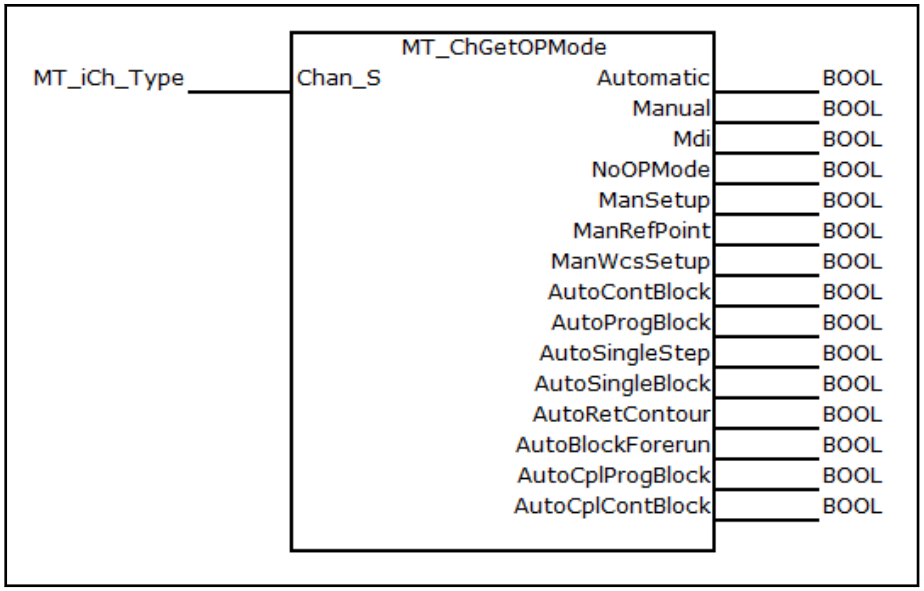


Fig. 14-50: Function block MT_ChGetOPMode

	Name	Type	Description
VAR_INPUT	Chan_S	MT_iCH_TYPE	Channel interface: Channel-status signals
VAR_OUTPUT	Automatic	BOOL	Primary mode: Automatic mode
	Manual	BOOL	Primary mode: Manual mode
	Mdi	BOOL	Primary mode: MDI - Manual data input; NC block
	NoOPMode	BOOL	No operation mode
	ManSetup	BOOL	Secondary mode: Manual setup of jog mode
	ManRefPoint	BOOL	Secondary mode: Setup approach reference point
	ManWcsSetup	BOOL	Secondary mode: Manual setup: workpiece coordinates
	AutoContBlock	BOOL	Secondary mode: Continuous block
	AutoProgBlock	BOOL	Secondary mode: Program block
	AutoSingleStep	BOOL	Secondary mode: Single step
	AutoSingleBlock	BOOL	Secondary mode: Single block
	AutoRetContour	BOOL	Secondary mode: Return to contour
	AutoBlockForerun	BOOL	Secondary mode: Block pre-run
	AutoCplProgBlock	BOOL	Secondary mode: CPL debugger: program block
	AutoCplContBlock	BOOL	Secondary mode: CPL debugger: continuous block

Tab. 14-33: Inputs/outputs function block MT_ChGetOPMode

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

The MTX uses three main operation modes and eleven secondary operation modes for each channel. All states a channel can assume with regard to the operation mode are bit-coded via the channel interface (MT_iCh_Type) by the signals "iCh_OpMode_00 .. 03" from the NC to the PLC. Only one main operation mode with one secondary mode (if available) is active at a time. If no operation mode is selected, this state is also valid. The following table shows all main operation modes with the assignment of the secondary operation

modes and the respective signal outputs of the function block which are TRUE if the channel is in the respective state.

Main and secondary operation mode

Primary mode	Secondary mode	Function block output signal
Automatic mode	Continuous block	AutoContBlock, Automatic
Automatic mode	Program block	AutoProgBlock, Automatic
Automatic mode	Single step	AutoSingleStep, Automatic
Automatic mode	Single block	AutoSingleBlock, Automatic
Automatic mode	Return to contour	AutoContBlock, Automatic
Automatic mode	Block pre-run	AutoBlockForerun, Automatic
Automatic mode	CPL debugger: program block	AutoCplProgBlock, Automatic
Automatic mode	CPL debugger: continuous block	AutoCplContBlock, Automatic
Manual mode	Manual setup of jog mode	ManSetup, manual
Manual mode	Setup approach reference point	ManRefPoint, Manual
Manual mode	Manual setup: workpiece coordinates	ManWcsSetup, Manual
MDI	None	Mdi
None	None	NoOPMode

Tab. 14-34: Overview on main and secondary operation modes with the respective interface signals

Troubleshooting No error handling required for this function.

Implementation example The operation mode of the first channel is determined in the example program.

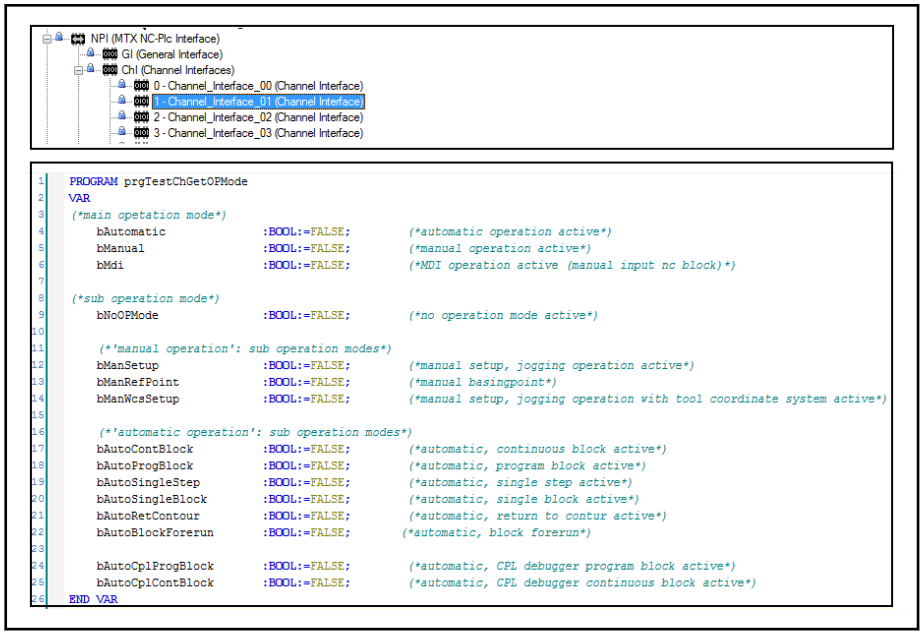


Fig. 14-51: Declaration MT_ChGetOPMode

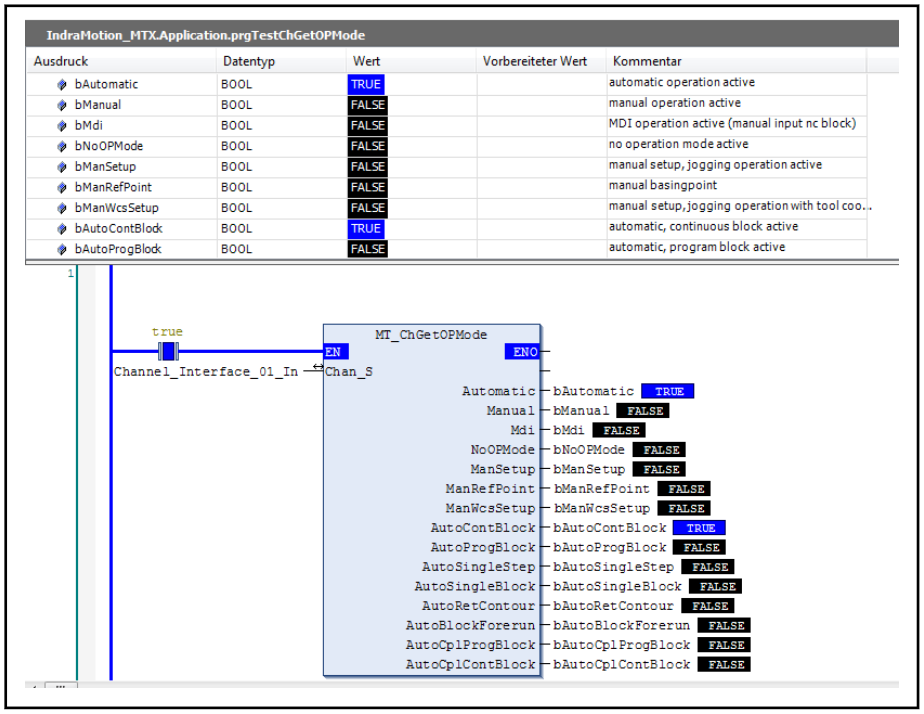


Fig. 14-52: Implementation MT_ChGetOPMode

14.3.10 MT_ChSetMainOPMode, Changing the NC main operation modes

Brief description Function block to change the main operation modes.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-35: Library names according to the target systems

Interface description

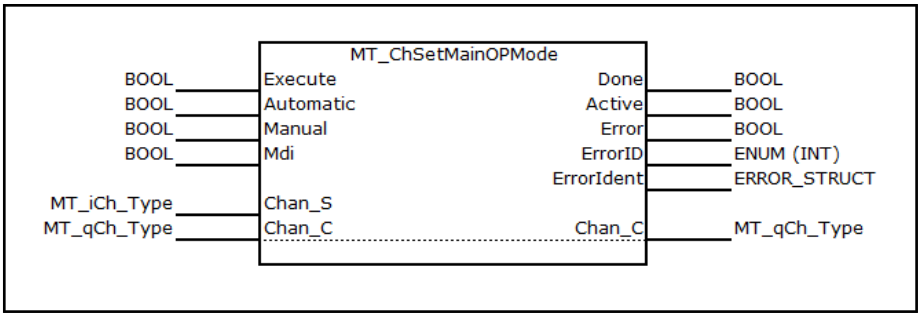


Fig. 14-53: Function block MT_ChSetMainOPMode

	Name	Type	Description
VAR_INPUT	Execute	BOOL	Setting the operation mode execution
	Automatic	BOOL	Primary mode: Automatic mode
	Manual	BOOL	Primary mode: Manual mode
	Mdi	BOOL	Primary mode: MDI - Manual data input; NC block
	Chan_S	MT_iCh_Type	Channel interface: Channel-status signals
VAR_OUTPUT	Done	BOOL	Processing successfully completed
	Active	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "Error" = TRUE)
VAR_IN_OUT	Chan_C	MT_qCh_Type	Channel interface: Channel-control signals

Tab. 14-36: Inputs/outputs function block MT_ChSetMainOPMode

Signal-time diagram

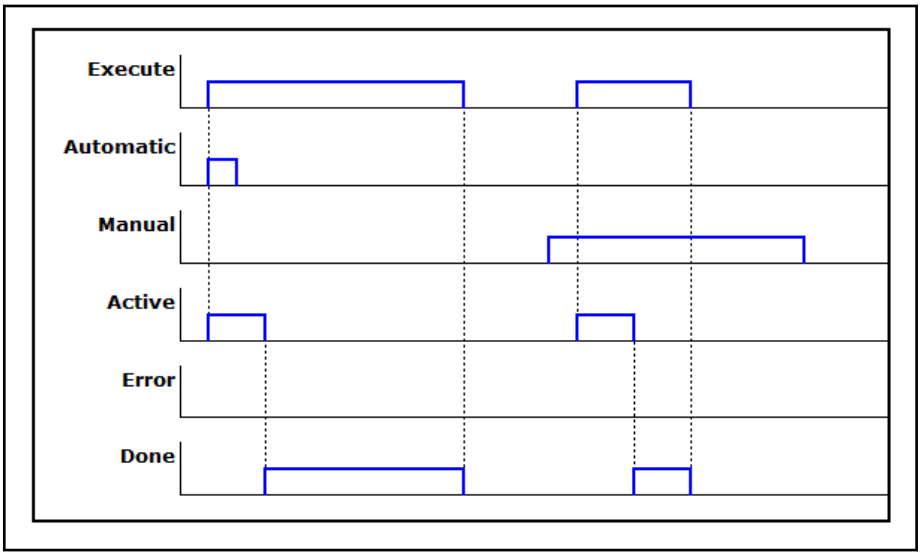


Fig. 14-54: Signal-time diagram MT_ChSetMainOPMode

Left: Setting operation mode "Automatic" completed successfully

Right: Setting operation mode "Manual" completed successfully

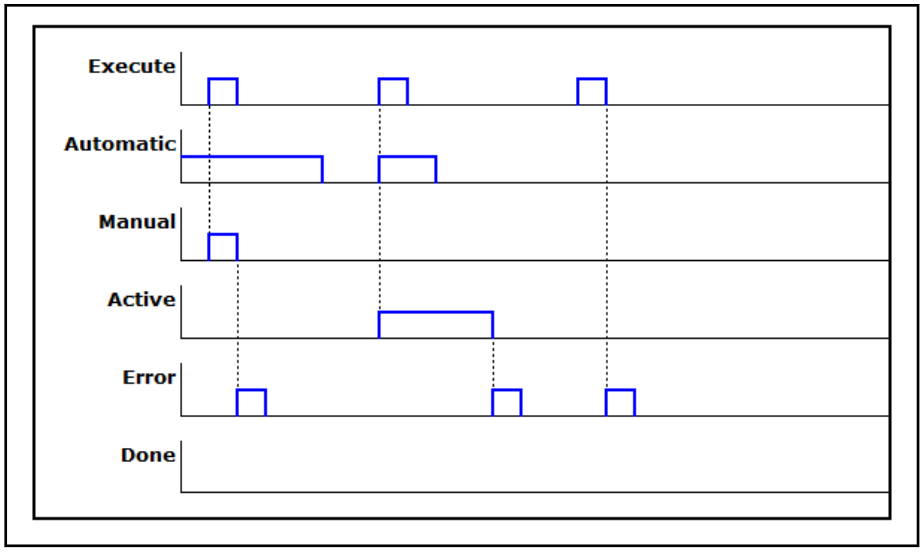


Fig. 14-55: Signal-time diagram MT_ChSetMainOPMode

Error left: More than one operation mode is preselected

Error in the center: Timeout (500ms), no acknowledgement that the operation mode was set

Error right: No operation mode was preselected

Functional description

The MTX uses three main operation modes and eleven secondary operation modes for each channel. All states a channel can assume with regard to the operation mode are bit-coded via the channel interface (MT_qCh_Type) by the signals "qCh_OpModeSel_00 .. 03" from the PLC to the NC. Only one main operation mode with one secondary mode (if available) is active at a time. If no operation mode is selected, this state is also valid. Only main operation modes can be switched using the function block "MT_SetMainOpMode". When switching the main operation mode (e.g. from automatic to manual mode), the currently active secondary operation mode is backed up (e.g. "continuous block"). When switching back to this main operation mode, the last active secondary operation mode is set again.



This function block works edge-controlled. That means that after the activation, the function block changes the operation mode exactly once.

A signal change from zero to one (0 -> 1) at "Execute" enables the function block and backs up the state of the inputs at this point in time. Only the inputs "Chan_S" and "Chan_C" are cyclically applied until the function is active ("Active"==TRUE). When changing the signal, the desired main operation mode has to be preselected (e.g. "Manual"). As soon as the "Done" output becomes TRUE, setting the desired operation mode is successfully completed. If the operation mode could not be changed, this is displayed by the "Error" output. As long as "Error" is TRUE, there is a valid error code under "ErrorIdent". Before switching the operation mode again or deleting the error, "Execute" has to be set to FALSE. Resetting "Execute" deletes "Error" and "Done". As long as "Active" is TRUE , signal changes at the input to preselect the operation mode (e.g. "Manual") have no effect. The outputs "Done", "Active" and "Error" exclude each other. That means that only one is TRUE at a time.

When switching the main operation mode, a channel reset is always executed.

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_ChSetMainOPMode"

Error code (hex)	Description
0x00000000	No error code available
0x00010001	Invalid input assignment. No operation mode or several operation modes were preselected
0x00040201	Changing the interface signals "Chan_C.qCh_OpModeSel_00..03" failed. The interfaces might have been overwritten outside of the function block
0x00020201	The operation mode change was not acknowledged via the "Chan_S.iCh_OpMode_00..03" interface signals within 500 ms. The interface for another channel might be connected to the function block.
0x 00050001	Internal error
0x 00050002	Internal error

Tab. 14-37: Error of "MT_SetMainOPMode" from "ErrorIdent.Additional1", valid if "Error"==TRUE

Implementation example

In this example, the operation mode is changed to the secondary operation mode "Automatic Continuous Block".

```
PROGRAM prgTestChSetMainOPMode
VAR
    fbSetMainOPMode         :MT_ChSetMainOPMode;
    bExecute                :BOOL:=FALSE;          (*execute setting operation mode*)

(*main opetation mode*)
    bAutomatic              :BOOL:=FALSE;          (*activate automatic operation *)
    bManual                 :BOOL:=FALSE;          (*activate manual operation *)
    bMdi                    :BOOL:=FALSE;          (*activate MDI operation (manual input nc block)*)

    bDone                   :BOOL:=FALSE;           (*process successful finished*)
    bActive                 :BOOL:=FALSE;          (*process active*)
    bError                  :BOOL:=FALSE;          (*process incorrect aborted, 'enErrorID' and 'stErrorIdent' valid*)
    enErrorID               :ERROR_CODE;           (*basic error information, only valid when 'bError'==TRUE*)
    stErrorIdent            :ERROR_STRUCT;         (*detailed error information, only valid when 'bError'==TRUE*)
END_VAR
```

Fig. 14-56: Declaration MT_ChSetMainOPMode

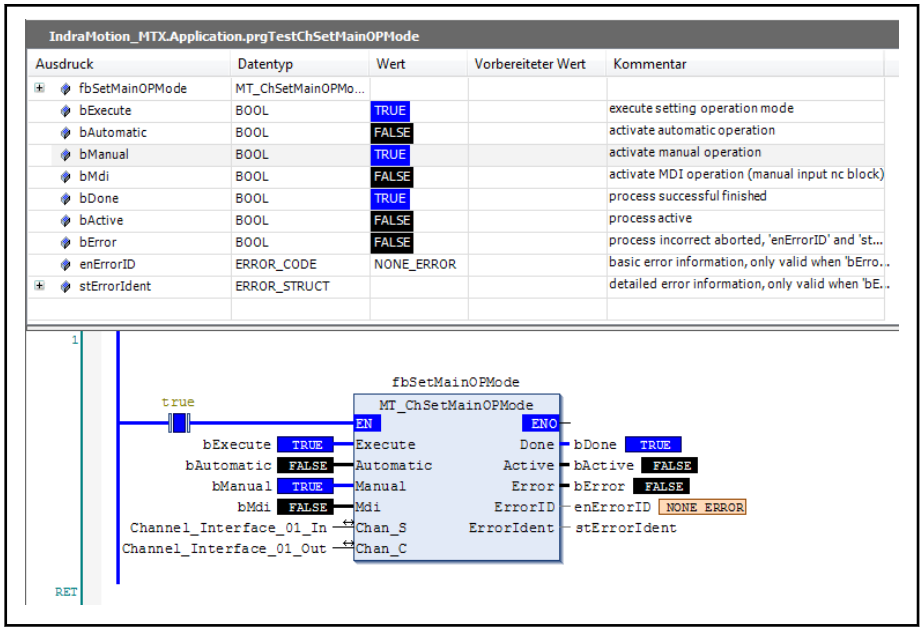


Fig. 14-57: Implementation MT_ChSetMainOPMode, changing operation mode completed successfully

14.3.11 MT_ChSetOPMode, Changing the NC operation modes

Brief description

Function block to change the NC main and secondary operation modes.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-38: Library names according to the target systems

Interface description

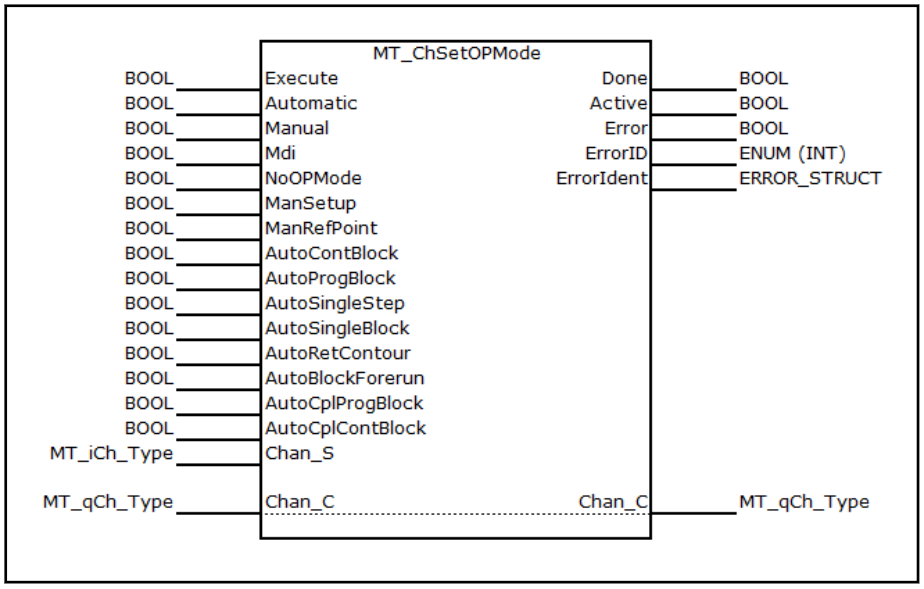


Fig. 14-58: Function block MT_ChSetOPMode

	Name	Type	Description
VAR_INPUT	Execute	BOOL	Setting the operation mode execution
	Automatic	BOOL	Primary mode: Automatic mode
	Manual	BOOL	Primary mode: Manual mode
	Mdi	BOOL	Primary mode: MDI - Manual data input; NC block
	NoOPMode	BOOL	No operation mode
	ManSetup	BOOL	Secondary mode: Manual setup of jog mode
	ManRefPoint	BOOL	Secondary mode: Setup approach reference point
	ManWcsSetup	BOOL	Secondary mode: Manual setup: workpiece coordinates
	AutoContBlock	BOOL	Secondary mode: Continuous block
	AutoProgBlock	BOOL	Secondary mode: Program block
	AutoSingleStep	BOOL	Secondary mode: Single step
	AutoSingleBlock	BOOL	Secondary mode: Single block
	AutoRetContour	BOOL	Secondary mode: Return to contour
	AutoCplProgBlock	BOOL	Secondary mode: CPL debugger (NC analyzer): program block
	AutoCplContBlock	BOOL	Secondary mode: CPL debugger: continuous block
	Chan_S	MT_iCh_Type	Channel interface: Channel-status signals
VAR_OUTPUT	Done	BOOL	Processing successfully completed
	Active	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "Error" = TRUE)
VAR_IN_OUT	Chan_C	MT_qCh_Type	Channel interface: Channel-control signals

Tab. 14-39: Inputs/outputs function block MT_ChSetOPMode

Signal-time diagram

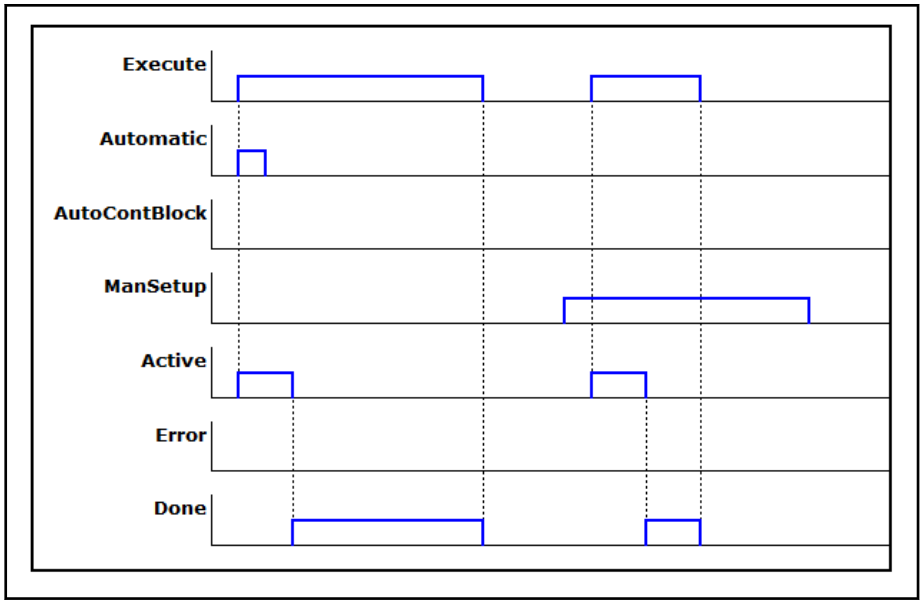


Fig. 14-59: Signal-time diagram MT_ChSetOPMode

Left: Setting the main operation mode "Automatic" completed successfully

Right: Setting secondary operation mode "Set manual jog mode" successful

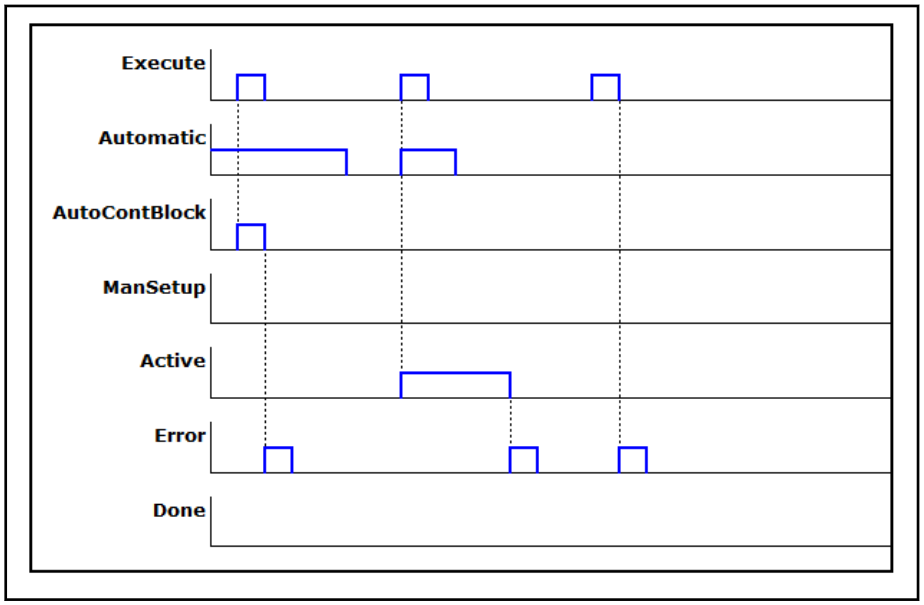


Fig. 14-60: Signal-time diagram MT_ChSetOPMode

Error left: More than one operation mode is preselected

Error in the center: Timeout (500ms), no acknowledgement that the operation mode was se

Error right: No operation mode was preselected

Functional description

The MTX uses three main operation modes and eleven secondary operation modes for each channel. All states a channel can assume with regard to the operation mode are bit-coded via the channel interface (MT_qCh_Type) by the signals "qCh_OpModeSel_00 .. 03" from the PLC to the NC. Only one main operation mode with one secondary mode (if available) is active at a time. If no operation mode is selected, this state is also valid. The following table shows all main operation modes with the assignment of the secondary operation modes and the respective function block input signals.

Main and secondary operation mode

Primary mode	Secondary mode	Function block input signal
Automatic mode	As before	Automatic
Manual mode	As before	Manual
MDI	None	Mdi
None	None	NoOPMode
Automatic mode	Continuous block	AutoContBlock
Automatic mode	Program block	AutoProgBlock
Automatic mode	Single step	AutoSingleStep
Automatic mode	Single block	AutoSingleBlock
Automatic mode	Return to contour	AutoContBlock
Automatic mode	Block pre-run	AutoBlockForerun
Automatic mode	CPL debugger: program block	AutoCplProgBlock
Automatic mode	CPL debugger: continuous block	AutoCplContBlock
Manual mode	Manual setup of jog mode	ManSetup
Manual mode	Setup approach reference point	ManRefPoint
Manual mode	Manual setup: workpiece coordinates	ManWcsSetup

Tab. 14-40: Overview on main and secondary operation modes with the respective interface signals

Main and secondary operation modes are switched with the "MT_ChSetOpMode" function block. When switching the main operation mode (e.g. from automatic to manual mode), the currently active secondary operation mode is backed up (e.g. "continuous block"). If switched back to this main operation mode later on, the latest secondary operation mode active is set again. If only the secondary operation mode is switched within a main operation mode (e.g. "continuous block" to "single block"), the main operation mode remains. If the secondary operation mode is switched outside the main operation mode (e.g. from "continuous block" to "manual setup: jogging"), the main operation mode is automatically switched as well.



This function block works edge-controlled. That means that the function block changes the operation mode changes exactly once after activation.

A signal change from zero to one (0 -> 1) at "Execute" enables the function block and backs up the state of the inputs at this point in time. Only the inputs "Chan_S" and "Chan_C" are cyclically applied until the function is active ("Active"==TRUE). When changing the signal, the desired main and secondary operation mode has to be preselected (e.g. "AutoContBlock"). When none or multiple operation modes are preselected when setting "Execute", the func-

tion block goes into error state ("Error" becomes TRUE). As soon as the "Done" output becomes TRUE, setting the desired operation mode is successfully completed. If the operation mode could not be changed, this is displayed by the "Error" output. As long as "Error" is TRUE, there is a valid error code under "ErrorIdent". Before switching the operation mode again or deleting the error, "Execute" has to be set to FALSE. Resetting "Execute" deletes "Error" and "Done". As long as one of the outputs "Done", "Active" or "Error" is TRUE, the signal changes at the inputs to preselect the operation mode (e.g. "AutoContBlock") has no effect. The outputs "Done", "Active" and "Error" exclude each other. That means that only one is TRUE at a time.

When switching the main operation mode, a channel reset is always executed.

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_ChSetOPMode"

Error code (hex)	Description
0x00000000	No error code available
0x00010001	Invalid input assignment. No operation mode or several operation modes were preselected
0x00040201	Changing the interface signals "Chan_C.qCh_OpModeSel_00..03" failed. The interfaces might have been overwritten outside of the function block
0x00020201	The operation mode change was not acknowledged via the "Chan_S.iCh_OpMode_00..03" interface signals within 500 ms. The interface for another channel might be connected to the function block.
0x 00050001	Internal error
0x 00050002	Internal error

Tab. 14-41: Error of "MT_SetOPMode" from "ErrorIdent.Additional1", valid if "bError" == TRUE

Implementation example

In this example, the operation mode is changed to the secondary operation mode "Automatic Continuous Block".

```
PROGRAM prgTestChSetOPMode
VAR
    fbSetPMode              :MT_ChSetOPMode;
    bExecute                :BOOL:=FALSE;          (*execute setting operation mode*)

(*main opetation mode*)
    bAutomatic              :BOOL:=FALSE;          (*activate automatic operation *)
    bManual                 :BOOL:=FALSE;          (*activate manual operation *)
    bMdi                    :BOOL:=FALSE;          (*activate MDI operation (manual input nc block)*)
(*sub operation mode*)
    bNoOPMode               :BOOL:=FALSE;          (*no operation mode active*)

    (*'manual operation': sub operation modes*)
    bManSetup               :BOOL:=FALSE;          (*manual setup, jogging operation active*)
    bManRefPoint            :BOOL:=FALSE;          (*manual basingpoint*)
    bManWcsSetup            :BOOL:=FALSE;          (*manual setup, jogging operation with tool coordinate system active*)

    (*'automatic operation': sub operation modes*)
    bAutoContBlock          :BOOL:=FALSE;          (*automatic, continuous block active*)
    bAutoProgBlock          :BOOL:=FALSE;          (*automatic, program block active*)
    bAutoSingleStep         :BOOL:=FALSE;          (*automatic, single step active*)
    bAutoSingleBlock        :BOOL:=FALSE;          (*automatic, single block active*)
    bAutoRetContour         :BOOL:=FALSE;          (*automatic, return to contur active*)
    bAutoBlockForerun       :BOOL:=FALSE;          (*automatic, block forerun*)

    bAutoCplProgBlock       :BOOL:=FALSE;          (*automatic, CPL debugger program block active*)
    bAutoCplContBlock       :BOOL:=FALSE;          (*automatic, CPL debugger continuous block active*)

    bDone                   :BOOL:=FALSE;           (*process successful finished*)
    bActive                 :BOOL:=FALSE;           (*process active*)
    bError                  :BOOL:=FALSE;           (*process incorrect aborted, 'enErrorID' and 'stErrorIdent' valid*)
    enErrorID               :ERROR_CODE;            (*basic error information, only valid when 'bError'==TRUE*)
    stErrorIdent            :ERROR_STRUCT;          (*detailed error information, only valid when 'bError'==TRUE*)
END_VAR
```

Fig. 14-61: Declaration MT_ChSetOPMode

IndraMotion_MTX.Application.prgTestChSetOPMode

Ausdruck	Datentyp	Wert	Vorbereiteter W...	Kommentar
fbSetPMode	MT_ChSetOPMode			
bExecute	BOOL	TRUE		execute setting operation mode
bAutomatic	BOOL	FALSE		activate automatic operation
bManual	BOOL	FALSE		activate manual operation
bMdi	BOOL	TRUE		activate MDI operation (manual input nc block)
bNoOPMode	BOOL	FALSE		no operation mode active
bManSetup	BOOL	FALSE		manual setup, jogging operation active

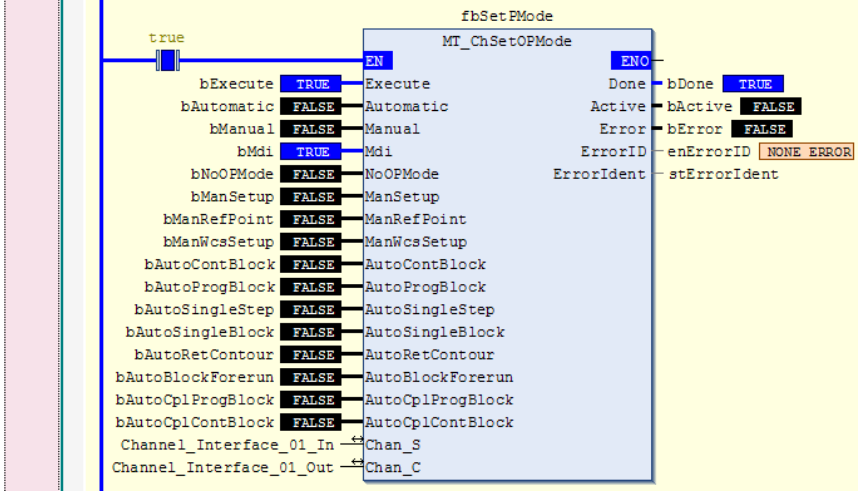


Fig. 14-62: Implementing MT_ChSetOPMode, changing operation mode completed successfully

14.3.12 MT_ChOPMode Changing the NC main operation modes

Brief description This function block is used to change the main and secondary operation mode of a channel. Furthermore, the active main and secondary operation mode of a channel is cyclically determined.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-42: Library names according to the target systems

Interface description

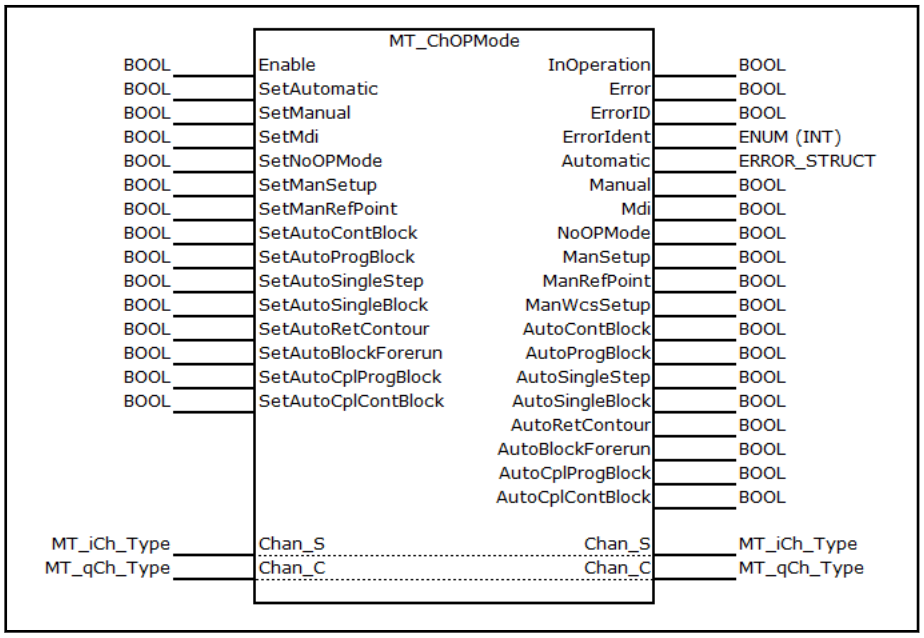


Fig. 14-63: MT_ChOPMode function block

	Name	Type	Description
VAR_INPUT	Enable	BOOL	Enables/disables the function block
	SetAutomatic	BOOL	Set main operation mode: Automatic mode
	SetManual	BOOL	Set main operation mode: Manual mode
	SetMdi	BOOL	Set main operation mode: MDI - Manual data input; NC block
	SetNoOPMode	BOOL	Do not set operation mode
	SetManSetup	BOOL	Set secondary mode: Manual setup of jog mode
	SetManRefPoint	BOOL	Set secondary mode: Setup approach reference point
	SetManWcsSetup	BOOL	Set secondary mode: Manual setup: workpiece coordinates
	SetAutoContBlock	BOOL	Set secondary mode: Continuous block
	SetAutoProgBlock	BOOL	Set secondary mode: Program block
	SetAutoSingleStep	BOOL	Set secondary mode: Single step
	SetAutoSingleBlock	BOOL	Set secondary mode: Single block
	SetAutoRetContour	BOOL	Set secondary mode: Return to contour
	SetAutoCplProgBlock	BOOL	Set secondary mode: CPL debugger (NC analyzer): program block
	SetAutoCplContBlock	BOOL	Set secondary mode: CPL debugger: continuous block

	Name	Type	Description
VAR_OUTPUT	InOperation	BOOL	Processing successfully completed
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "Error" = TRUE)
	Automatic	BOOL	Primary mode: Automatic mode active
	Manual	BOOL	Primary mode: Manual mode active
	Mdi	BOOL	Primary mode: MDI - Manual data input; NC block active
	NoOPMode	BOOL	No operation mode active
	ManSetup	BOOL	Secondary mode: Manual setup of jog mode active
	ManRefPoint	BOOL	Secondary mode: Setup approach reference point active
	ManWcsSetup	BOOL	Secondary mode: Manual setup: workpiece coordinates active
	AutoContBlock	BOOL	Secondary mode: Continuous block active
	AutoProgBlock	BOOL	Secondary mode: Program block active
	AutoSingleStep	BOOL	Secondary mode: Single step active
	AutoSingleBlock	BOOL	Secondary mode: Single block active
	AutoRetContour	BOOL	Secondary mode: Return to contour active
	AutoBlockForerun	BOOL	Secondary mode: Block pre-run active
	AutoCplProgBlock	BOOL	Secondary mode: CPL debugger: program block active
	AutoCplContBlock	BOOL	Secondary mode: CPL debugger: continuous block active
VAR_IN_OUT	Chan_S	MT_iCh_Type	Channel interface: Channel-status signals
	Chan_C	MT_qCh_Type	Channel interface: Channel-control signals

Tab. 14-43: Inputs/outputs function block MT_ChOPMode

Signal-time diagram

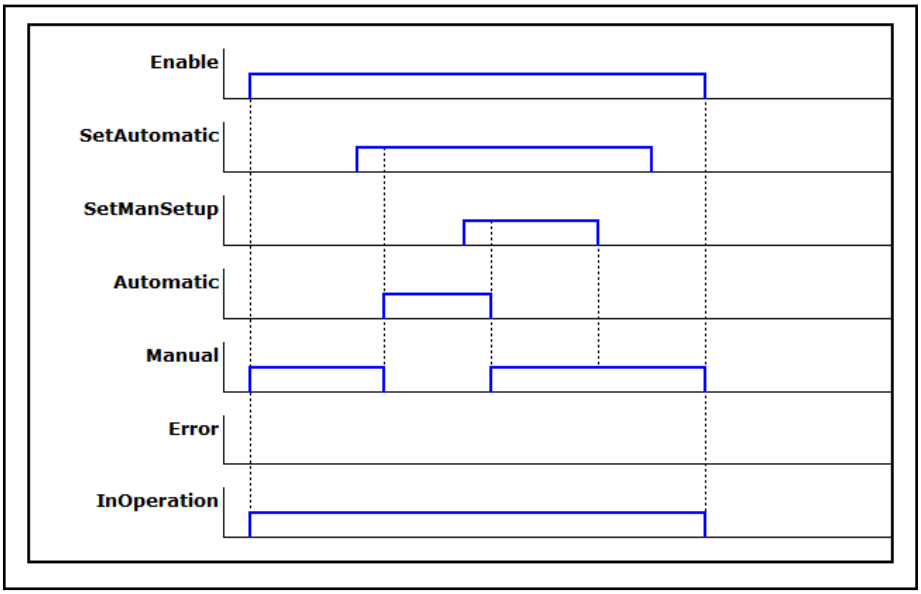


Fig. 14-64: Manual mode active, change to Automatic mode, change to manual mode

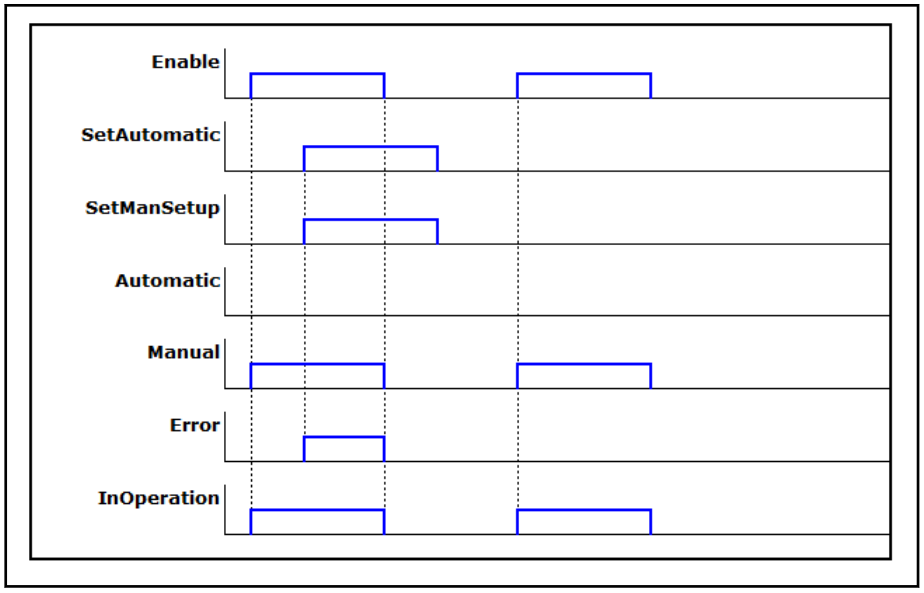


Fig. 14-65: Error: Simultaneous change to 2 operation modes, reset by Enable=FALSE

Functional description

Using this function block, the active main and secondary operation mode of a channel can be determined and changed.

The MTX uses three main operation modes and eleven secondary operation modes for each channel. The channel operating mode is changed bit-coded via the channel interface (MT_qCh_Type) by the signals "qCh_OpModeSel_00 ... 03" from the PLC to the NC. The active channel operation mode is read out bit-coded via the channel interface (MT_iCh_Type) by the signals "iCh_OpMode_00 .. 03" from the NC to the PLC. Only one main operating mode with one secondary mode (if available) is active at a time. If no operation mode is selected, this state is also valid. When switching the main operation mode (e.g. from automatic to manual mode), the currently active secondary operation mode is backed up (e.g. "continuous block"). When switching back to this main operation mode, the last active secondary operation mode is set again.



This function block works state-controlled. That means that the operation mode can always be changed after activation.

The function block is active as long as the "Enable" input is TRUE. The state is displayed via the "InOperation" output. Only if "InOperation" is TRUE, the operation mode outputs are valid and the operation mode can be changed. This is indicated by the "Error" output. As long as "Error" is TRUE, there is a valid error code under the "ErrorIdent" output. Set "Enable" to FALSE to delete the error. All inputs are applied upon each function block call. Thus, the current operation mode is determined from the interface signals "iCh_OpMode_00 .. 03" upon each function block call The operation mode is changed by a signal change (0-->1) at the corresponding operation mode input and becomes active after one PLC cycle. If several inputs are set, the input at which a signal change was detected last to change the operating mode. The outputs "InOperation" and "Error" exclude each other. That means that only one output is TRUE at a time.



When switching the main operation mode, a channel reset is always executed.

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

References to the diagnostic table

- MTX_TABLE

Error of "MT_ChOPMode"

Error code (hex)	Description
0x00000000	No error code available
0x00010001	Invalid input assignment. No operation mode or several operation modes were preselected
0x00040201	Changing the interface signals "Chan_C.qCh_OpMode-Sel_00..03" failed. The interfaces might have been overwritten outside of the function block
0x00020201	The acknowledgement on the change in operation mode via the interface signals "stChan_S.iCh_OpMode_00..03" was not made within a period of 500ms. The interface for another channel might be connected to the function block.
0x 00050001	Internal error
0x 00050002	Internal error

*Tab. 14-44: Error of **MT_ChOPMode**, of "ErrorIdent.Additional1", valid if "Error"==TRUE*

Implementation example

In this example, the operation mode is changed to the secondary operation mode "Automatic Continuous Block".

```
PROGRAM prgTestChOPMode
VAR
    bEnable                 :BOOL:=TRUE;
    fbOPMode                :MT_ChOPMode;
//Eingänge
    bSetAutomatic           :BOOL:=FALSE;       (*activate automatic operation *)
    bSetManual              :BOOL:=FALSE;       (*activate manual operation *)
    bSetMdi                 :BOOL:=FALSE;       (*activate MDI operation (manual input nc block)*)
(*sub operation mode*)
    bSetNoOPMode            :BOOL:=FALSE;       (*no operation mode active*)
    (*'manual operation': sub operation modes*)
    bSetManSetup            :BOOL:=FALSE;       (*manual setup, jogging operation active*)
    bSetManRefPoint         :BOOL:=FALSE;       (*manual basingpoint*)
    bSetManWcsSetup         :BOOL:=FALSE;       (*manual setup, jogging operation with tool coordinate
    (*'automatic operation': sub operation modes*)
    bSetAutoContBlock       :BOOL:=FALSE;       (*automatic, continuous block active*)
    bSetAutoProgBlock       :BOOL:=FALSE;       (*automatic, program block active*)
    bSetAutoSingleStep      :BOOL:=FALSE;       (*automatic, single step active*)
    bSetAutoSingleBlock     :BOOL:=FALSE;       (*automatic, single block active*)
    bSetAutoRetContour      :BOOL:=FALSE;       (*automatic, return to contur active*)
    bSetAutoBlockForerun    :BOOL:=FALSE;       (*automatic, block forerun*)
    bSetAutoCplProgBlock    :BOOL:=FALSE;       (*automatic, CPL debugger program block active*)
    bSetAutoCplContBlock    :BOOL:=FALSE;       (*automatic, CPL debugger continuous block active*)

//Ausgänge
    (*main opetation mode*)
    bAutomatic              :BOOL:=FALSE;     (*automatic operation active*)
    bManual                 :BOOL:=FALSE;     (*manual operation active*)
    bMdi                    :BOOL:=FALSE;     (*MDI operation active (manual input nc block)*)
    (*sub operation mode*)
    bNoOPMode               :BOOL:=FALSE;     (*no operation mode active*)
    (*'manual operation': sub operation modes*)
```

Fig. 14-66: Declaration MT_ChOPMode

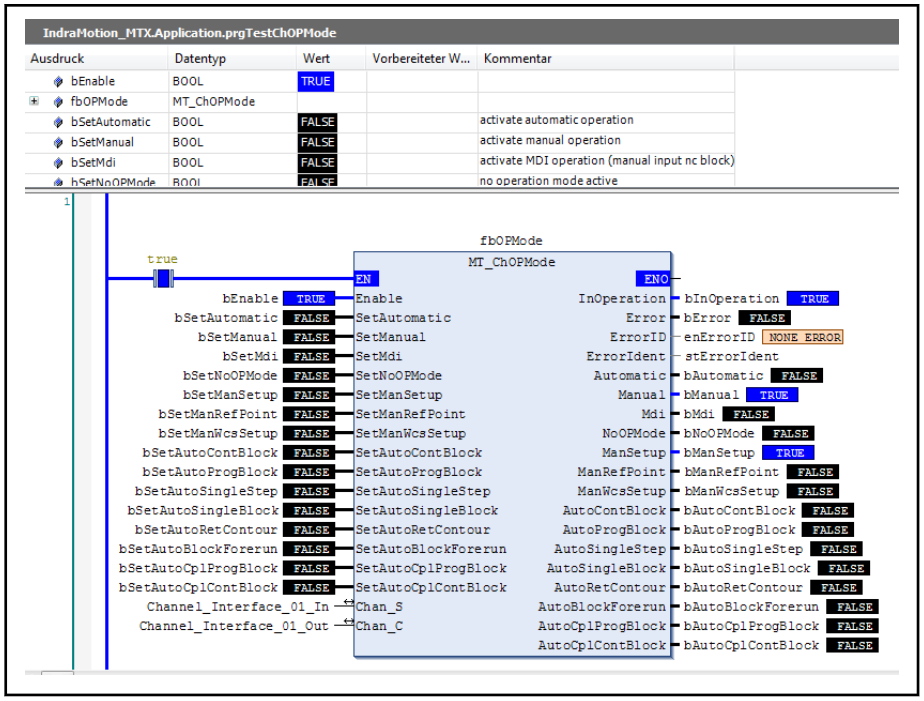


Fig. 14-67: Implementation MT_ChOPMode

14.4 Axes

14.4.1 Overview

Overview on axis functions		
Category	**Function/function block**	**Brief description**
General information	MT_DrvGetSafeState, page 274	Determines the active safety engineering operating state of a drive (also for spindle)
	MT_AxGetChIndex, page 278	Check channel assignment of an axis
	MT_AxHandwheelACS, page 278	Operates handwheel for an axis in the axis coordinate system
	MT_AxProbeMonitor, page 280	Monitoring of a probe axis
Asynchronous axes	MT_AxAsyncSetOverride, page 301	Change feed for asynchronous axes
	MT_AxAsyncOPMode, page298	Determines and changes the operation mode of an asynchronous axis (combines MT_AxAsyncGetOpMode and MT_AxAsyncSetOpMode)
	MT_AxAsyncGetOPMode, page 294	Determines the active operation mode of an asynchronous axis
	MT_AxAsyncSetOPMode, page 295	Changes the operation mode of an asynchronous axis

Jog distance	MT_AxJogMode, page 289	Determines and changes the active jog distance of an axis (combines MT_AxGetJogMode and MT_AxSetJogMode)
	MT_AxGetJogMode, page 283	Determines the active jog distance for an axis
	MT_AxSetJogMode, page 285	Changes the active jog distance for an axis
Power & diagnostics	MT_AxPower, page 308	Determines the drive status and sets/resets the controller enable (combines MT_AxSetPowerOn, MT_AxSetPowerOff and MT_AxGetPowerStateExt)
	MT_AxSetPowerOn, page 305	Sets the controller enable for an axis
	MT_AxSetPowerOff, page 306	Resets the controller enable for an axis
	MT_AxGetPowerState, page 302	Determines the drive status of an axis
	MT_AxGetPowerStateExt, page 303	Determines the drive status of an axis including the detailed information about errors and warnings

Tab. 14-45: Overview on axis functions

14.4.2 MT_DrvGetSafeState, Determining the active Safety Engineering operating state

Brief description

Using a drive, the function block determines the active Safety engineering operating state.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-46: Library names according to the target systems

Interface description

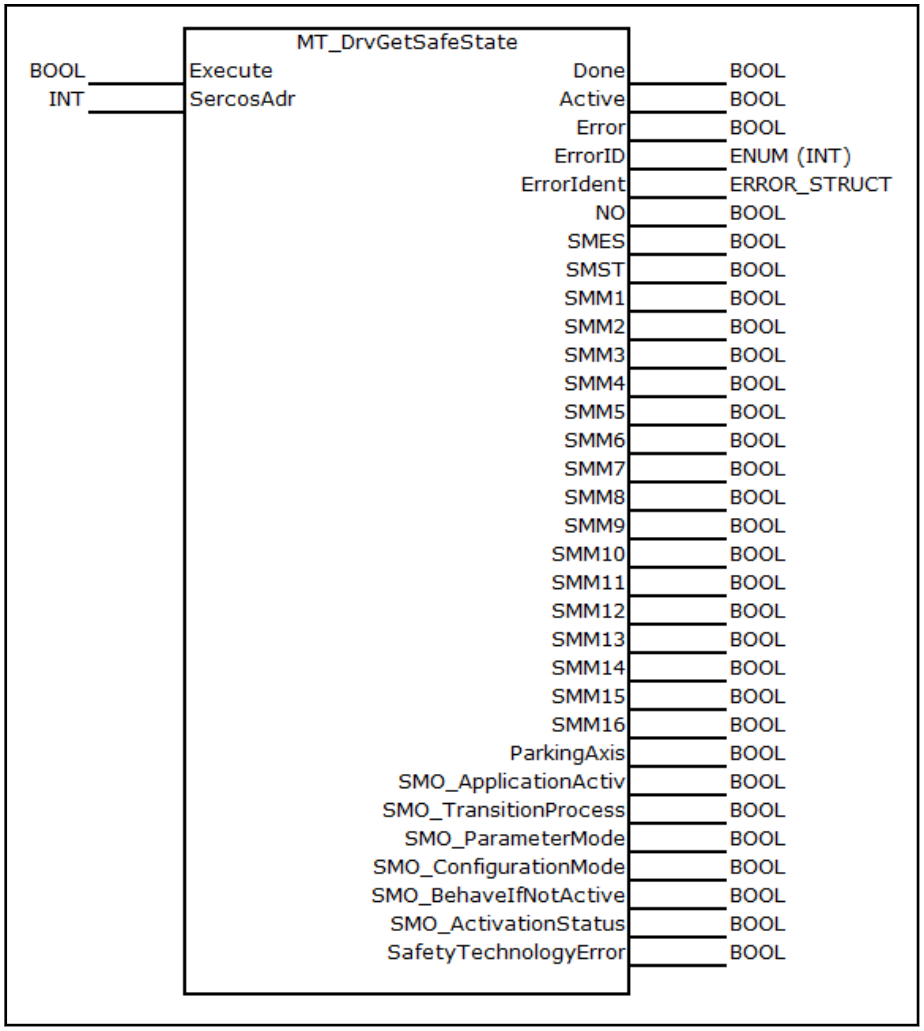


Fig. 14-68: Function block MT_DrvGetSafeState

	Name	Type	Description
VAR_INPUT	Execute	BOOL	Execute determining the Safety Engineering operating state
	SercosAdr	INT	Sercos address of the drive

	Name	Type	Description
VAR_OUTPUT	Done	BOOL	Processing successfully completed
	Active	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "bError" = TRUE)
	NO	BOOL	Normal mode, 1: active, 0: not active
	SMES	BOOL	EMERGENCY STOP, 1: active, 0: not active
	SMST	BOOL	Safe standstill, 1: active, 0: not active
	SMM1	BOOL	Safe motion 1, 1: active, 0: not active
	SMM2	BOOL	Safe motion 2, 1: active, 0: not active
	SMM3	BOOL	Safe motion 3, 1: active, 0: not active
	SMM4	BOOL	Safe motion 4, 1: active, 0: not active
	SMM5	BOOL	Safe motion 5, 1: active, 0: not active
	SMM6	BOOL	Safe motion 6, 1: active, 0: not active
	SMM7	BOOL	Safe motion 7, 1: active, 0: not active
	SMM8	BOOL	Safe motion 8, 1: active, 0: not active
	SMM9	BOOL	Safe motion 9, 1: active, 0: not active
	SMM10	BOOL	Safe motion 10, 1: active, 0: not active
	SMM11	BOOL	Safe motion 11, 1: active, 0: not active
	SMM12	BOOL	Safe motion 12, 1: active, 0: not active
	SMM13	BOOL	Safe motion 13, 1: active, 0: not active
	SMM14	BOOL	Safe motion 14, 1: active, 0: not active
	SMM15	BOOL	Safe motion 15, 1: active, 0: not active
	SMM16	BOOL	Safe motion 16, 1: active, 0: not active
	ParkingAxis	BOOL	Parking axis, 1: active, 0: not active
	SMO_ApplicationActiv	BOOL	Application active, 1: active, 0: not active
	SMO_TransitionProcess	BOOL	SMO transfer, 1: active, 0: not active
	SMO_ParameterMode	BOOL	SMO parameterization mode, 1: active, 0: not active
	SMO_Configuration-Mode	BOOL	SMO configuration mode, 1: active, 0: not active
	SMO_BehaveIfNotActive	BOOL	Behavior in case of active "Safe Motion" (only relevant if bit 31 =1) 1: functional commissioning, output element and brake(s) are controlled by the standard firmware 0: Delivery status
	SMO_ActivationStatus	BOOL	SMO activation status 1: "Safe Motion" Functionality is **not** active 0: "Safe Motion" functionality is active
	SafetyTechnologyError	BOOL	Safety engineering error 1: **no** error message 0: Error message

 Tab. 14-47: Inputs/outputs function block MT_DrvGetSafeState

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

The Sercos parameter P-0-3231.0.0 provides binary status signals for online monitoring of the Safety engineering operation mode of Safe Motion. This function block reads out the parameter and provides the result at the outputs.

The parameter is read out via the Sercos service channel. Repeated function block calls can increase the required capacity of the service channel. The recommended interval between two function block calls is at least 500 ms.



This function block works edge-controlled. That means that after the activation, the function block executes the Safety engineering operation mode exactly once.

A signal change from zero to one (0 -> 1) at "Execute" enables the function block and backs up the state of the inputs at this point in time. As soon as the "Done" output becomes TRUE, setting the desired Safety engineering operation mode is completed successfully. Otherwise, the output "Error" is TRUE. As long as "Error" is TRUE, there is a valid error code under "ErrorIdent". Before determining the Safety engineering operation mode, "Execute" has to be set to FALSE. Resetting "Execute" deletes "Error" and "Done". As long as one of the outputs "Done", "Active" or "Error" is TRUE, the signal changes at the "SercosAdr" input are not applied. The outputs "Done", "Active" and "Error" exclude each other. That means that only one is TRUE at a time.

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

The function block exclusively displays errors of the internally used function block "IL_SIIISvcRead" of the "RIL_SercosIII" library.

Implementation example

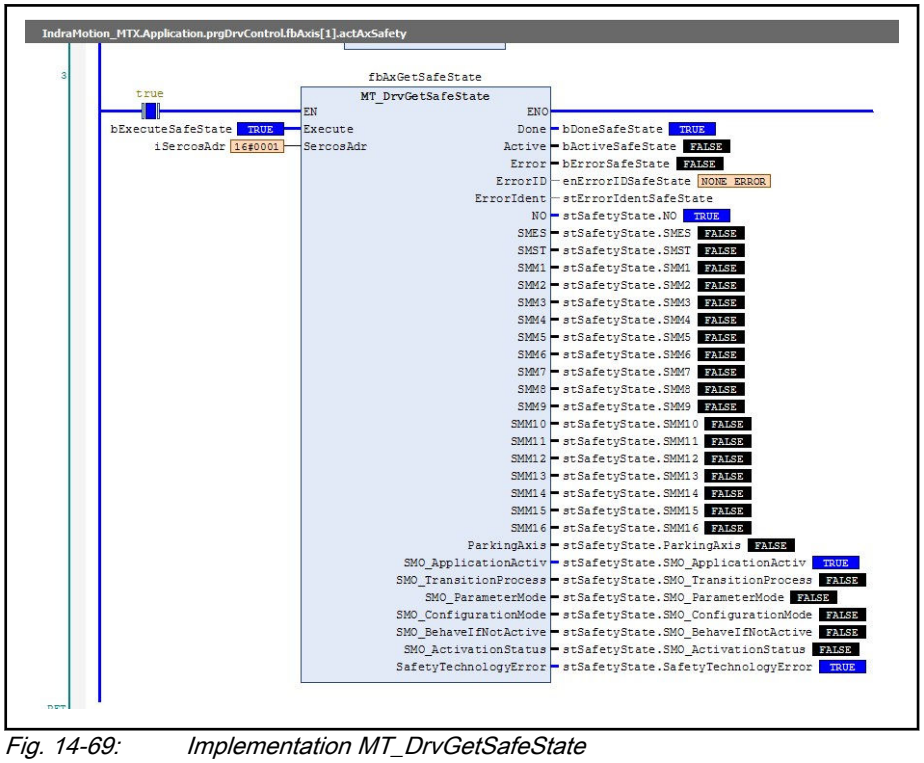


Fig. 14-69: Implementation MT_DrvGetSafeState

14.4.3 MT_AxGetChIndex, Determining the channel assignment of the axis

Brief description

Function to determine the channel assignment of an axis.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-48: Library names according to the target systems

Interface description

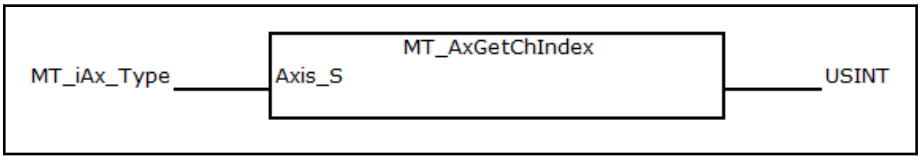


Fig. 14-70: Function block MT_AxGetChIndex

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

The current channel assignment of an axis is controlled via the channel interface (MT_iAx_Type) by the signals "iAx_ChIndex_00 .. 03" from the NC to the PLC. This function decodes this information and displays the channel number at the output of the function.

Troubleshooting

No error handling required for this function.

Implementation example

The operation mode of the first channel is determined in the example program.

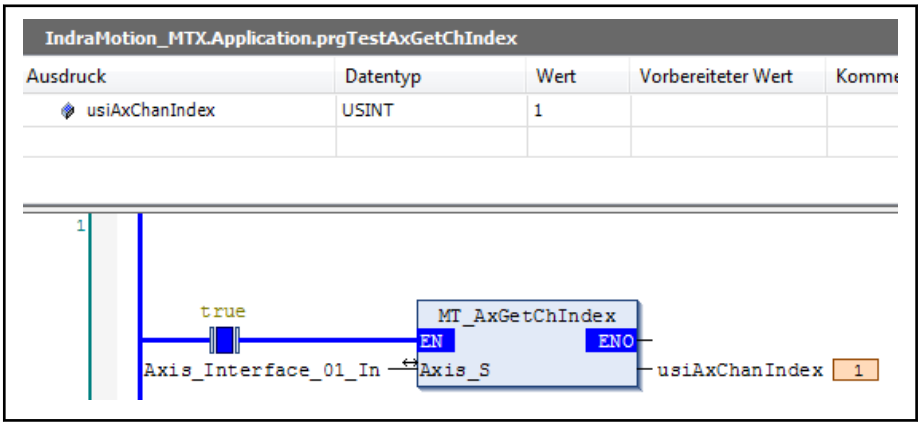


Fig. 14-71: Implementation MT_AxGetChIndex

14.4.4 MT_AxHandwheelAcs, Changing the handwheel

Brief description

The function block simplifies the handling when using the handwheel in the axis coordinate system.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-49: Library names according to the target systems

Interface description

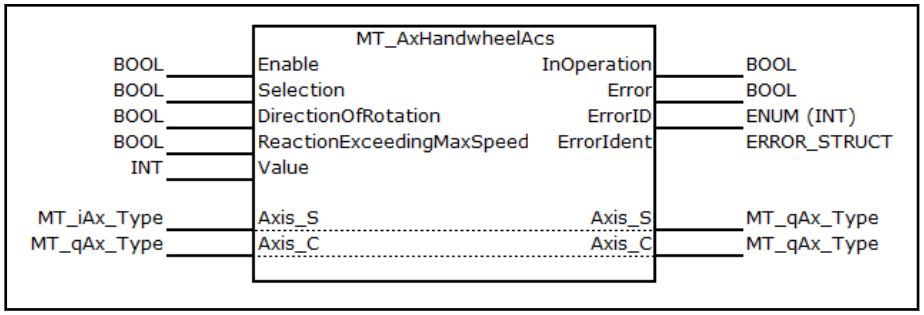


Fig. 14-72: Function block MT_AxHandwheelAcs

	Name	Type	Description
VAR_INPUT	Enable	BOOL	Enables the function block
	Selection	BOOL	Handwheel selection, FALSE==Handwheel1, TRUE==Handwheel2
	DirectionOfRotation	BOOL	FALSE==direction of rotation of handwheel and axis clockwise FALSE==direction of rotation of handwheel and axis counterclockwise
	ReactionExceeding-MaxSpeed	BOOL	FALSE==When exceeding the maximum velocity, the distance to be traversed is reduced accordingly. TRUE==When exceeding the maximum velocity, the distance to be traversed specified by the handwheel motions is also processed. Attention: Axis motions can also occur after completing the handwheel motion
	Value	INT	Handwheel pulse
VAR_OUTPUT	InOperation	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "Error" = TRUE)
VAR_IN_OUT	Axis_S	MT_iAx_Type	Axis interface: Axis status signals
	Axis_C	MT_qAx_Type	Axis interface: Axis control signals

Tab. 14-50: Inputs/outputs function block MT_AxHandwheelAcs

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

This function block simplifies the handling during commissioning of the handwheel. The function block connects the required interface signals "qAx_HandwPosMode", "qAx_HandwSel_00..01" and processes the "MT_Handw" function block so that no other actions are required.

The status of the inputs is applied once upon activation of the "Selection", "DirectionOfRotation" and "ReactionExceedingMaxSpeed" function blocks by settings "Enable".

The inputs "Value" as well as the AxisInterfaces "Axis_S" and "Axis_C" are read in upon each function block call and transferred to the NC.

☞ Refer to the settings in the machine parameters "OPF/Handw/Wh[1..2]/IncrPerRev".

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_AxHandweelAcs"

Error code (hex)	Description
0x00000000	No error code available
0x00060101	Asynchronous axis is not in manual operation mode
0x00060402	The selected handwheel is invalid
0x00030401	NC-internal communication overloaded. There are no resources available to transmit the task
0x00030403	General parameter error
0x00040101	Changing of at least one interface signal "qAx_HandwPosMode", "qAx_HandwSel_00..01" failed. Maybe the signals have been overwritten outside of the function block

Tab. 14-51: Error of "MT_AxHandwheelAcs" from "ErrorIdent.Additional1", valid if "Error"==TRUE

Implementation example

Example program

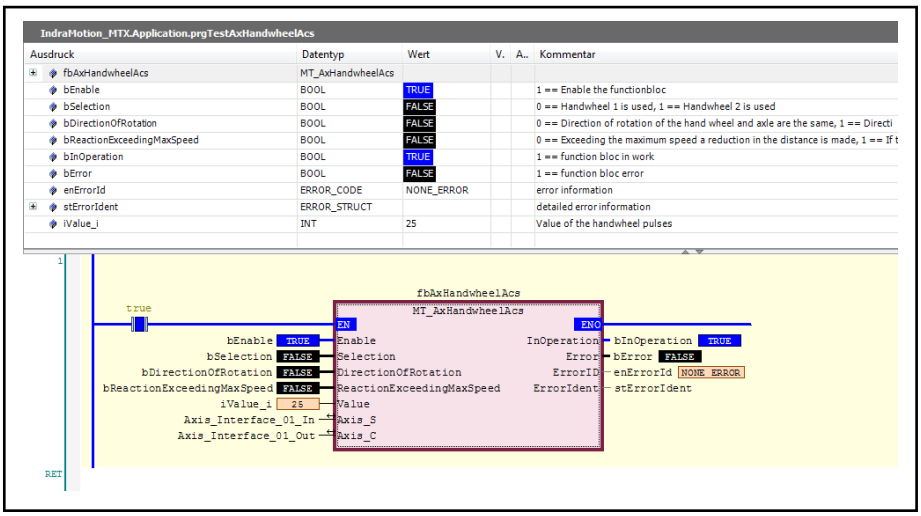


Fig. 14-73: Implementation MT_AxHandwheelAcs

14.4.5 MT_AxProbeMonitor, Monitoring the probe axis

Brief description

Function block for probe axis monitoring.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-52: Library names according to the target systems

Interface description

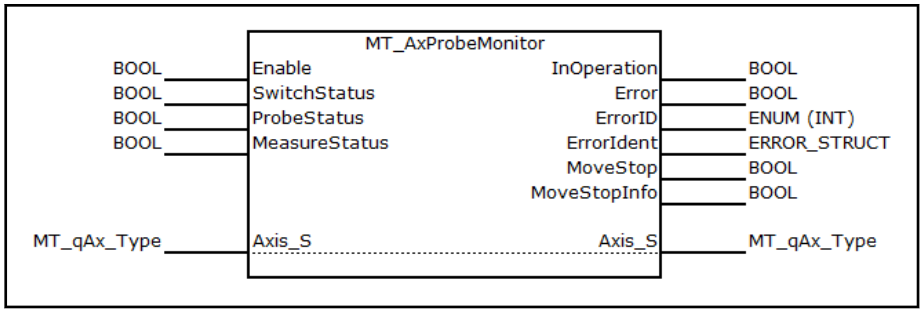


Fig. 14-74: Function block MT_AxProbeMonitor

	Name	Type	Description
VAR_INPUT	Enable	BOOL	Enables the function block
	SwitchStatus	BOOL	Activation state of the probe (1==on, 0==off)
	ProbeStatus	BOOL	Deflection state of the probe (1==deflected, 0==not deflected)
	MeasureStatus	BOOL	Measuring status (1==measurement active (G75), 0==measurement not active)
VAR_OUTPUT	InOperation	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "Error" = TRUE)
	MoveStop	BOOL	Motion stop should be set for this axis
	MoveStopInfo	BOOL	Reason for motion stop recommendation
VAR_IN_OUT	Axis_S	MT_iAx_Type	Axis interface: Axis status signals

Tab. 14-53: Inputs/outputs function block MT_AxProbeMonitor

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

The function block can monitor an axis included in the measuring process. Axis motions resulting in damaging the probe can minimized.



The deflection signal of the probe is directly connected to the drive and in parallel to the PLC. When measuring with G75, the deflection signal triggers an immediate stop of the axis at the drive. The drive does not react if the probe is deflected without G75. The signal is running from the probe to the PLC. From there, DriveLock is sent to the NC and the signal is forwarded to the drive via Sercos. This long signal runtime leads to a respectively long reaction time. For this reason, stopping the axis by the PLC cannot replace the probe function in the drive!

In general, the function block output "MoveStop" provides only a recommendation to prevent the axis motion via the interface signal "qAx_DrvLock" of the respective axis. The link has to be implemented by the PLC programmer. If the function block is activated by setting "Enable", the states of all inputs are monitored. If a direction-related motion lock becomes active, the output

"InOperation" is set to TRUE. The reset of "Enable" deletes all states stored in the function block as well as all outputs.



If the probe is deactivated, the function block cannot determine its actual deflection state. The deflection state "ProbeStatus" is only monitored if the probe is activated "SwitchStatus".

In the moment of the deflection, the function block stores the active motion direction of the axis and recommends a motion lock in the following situations:

Direction-related motion lock:

If the probe is activated and deflected, a motion lock is recommended ("MoveStop") if a motion is signalized in deflection direction. For a deflection direction opposite to the motion signal, the motion lock ("MoveStop") is deleted.

"MoveStopInfo"=2001 or 2002

Motion lock on assumed probe abort:

If an activated probe aborts during driving without G75, this corresponds to a brief deflection. If this happens in automatic mode, a motion lock is recommended ("MoveStop") even if the probe is no longer deflected. This can only be canceled by deleting the travel command.

"MoveStopInfo"=1000

Motion lock on measuring with deactivated probe:

When traveling with G75. the NC programmer expects that the drive stops the axis if the probe is triggered. However, this is only the case if the probe is activated. For this reason, the motion lock is recommended if the probe is not activated and the travel command for measuring (G75) applies. The motion lock can only be canceled by canceling the travel command or activation of the probe.

"iMoveStopInfo"=1003

Motion lock if probe deflected and deactivated:

If a previously deflected probe is deactivated, the previously recommended motion lock is retained. This can either be canceled by reactivation of the probe or by canceling the travel command.

"iMoveStopInfo"=1001

MoveStopInfo	Cause	Recovery
1000	The probe is currently not deflected, however, was previously deflected during travel without G75, a probe breakage is assumed	Cancel travel command for the axis
1001	Probe deactivated in deflected state	Reactivate probe or cancel travel command for the axis
1003	Travel command with measurement (G75) started but probe not activated	Activate probe or cancel travel command for the axis

2001	Motion announcement of the axis in positive direction although the probe is deflected in positive direction	Move axis in negative direction
2002	Motion announcement of the axis in negative direction although the probe is deflected in negative direction	Move axis in positive direction

Tab. 14-54: Overview of reasons for motion stop recommendation

Troubleshooting

Currently, no error codes available for this function block.

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_AxProbeMonitor"

Error code (hex)	Description
0x00000000	No error code available

Tab. 14-55: Error of "MT_AxProbeMonitor", of "ErrorIdent.Additional1", valid if "Error"==TRUE

Implementation example

Example program

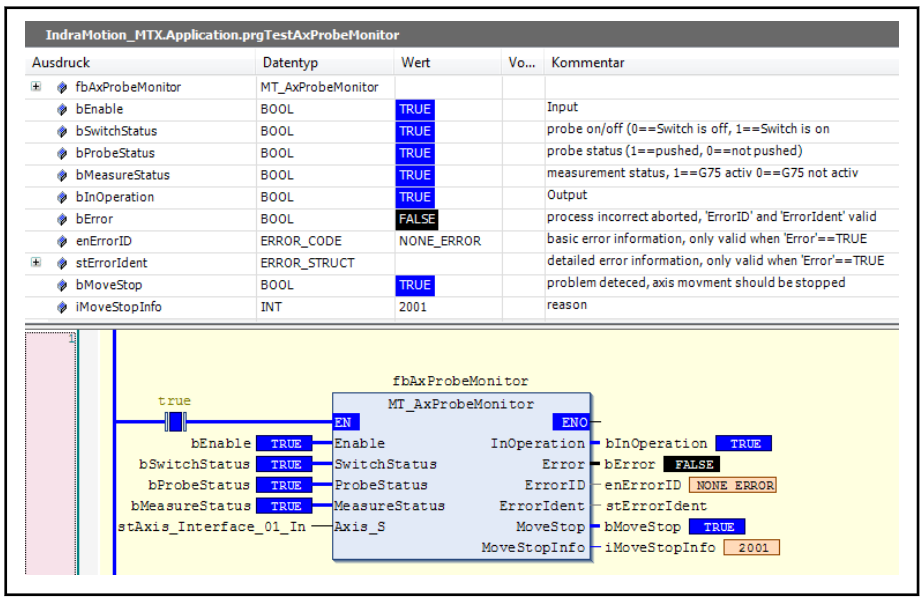


Fig. 14-75: Implementation MT_AxProbeMonitor

14.4.6 MT_AxGetJogMode, Determining Active JogMode

Brief description

Function to determine the active JogMode of an axis.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-56: Library names according to the target systems

Interface description

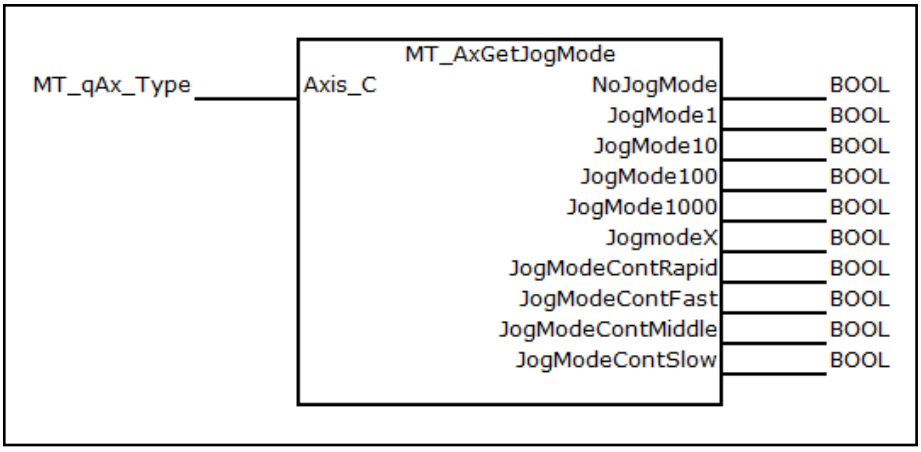


Fig. 14-76: Function block MT_AxGetJogMode

	Name	Type	Description
VAR_INPUT	Axis_C	MT_qAX_TYPE	Axis interface: Axis control signals
VAR_OUTPUT	NoJogMode	BOOL	No JogMode active
	JogMode1	BOOL	1 increment jogging active
	JogMode10	BOOL	10 increments jogging active
	JogMode100	BOOL	100 increments jogging active
	JogMode1000	BOOL	1000 increments jogging active
	JogModeX	BOOL	x increments jogging active
	JogModeContRapid	BOOL	Continuous jogging in rapid traverse
	JogModeContFast	BOOL	Continuous jogging fast velocity
	JogModeContMiddle	BOOL	Continuous jogging average velocity
	JogModeContSlow	BOOL	Continuous jogging slow velocity

Tab. 14-57: Inputs/outputs function block MT_AxGetJogMode

Signal-time diagram No special behavior that requires a signal-time diagram.

Functional description The JogMode is specified exclusively by the PLC to the NC. Thus, this function determines the JogMode last specified by the PLC of the channel interface (MT_qAx_Type) of the "qAx_ManFeed_00..03" signals.

Troubleshooting No error handling required for this function.

Implementation example

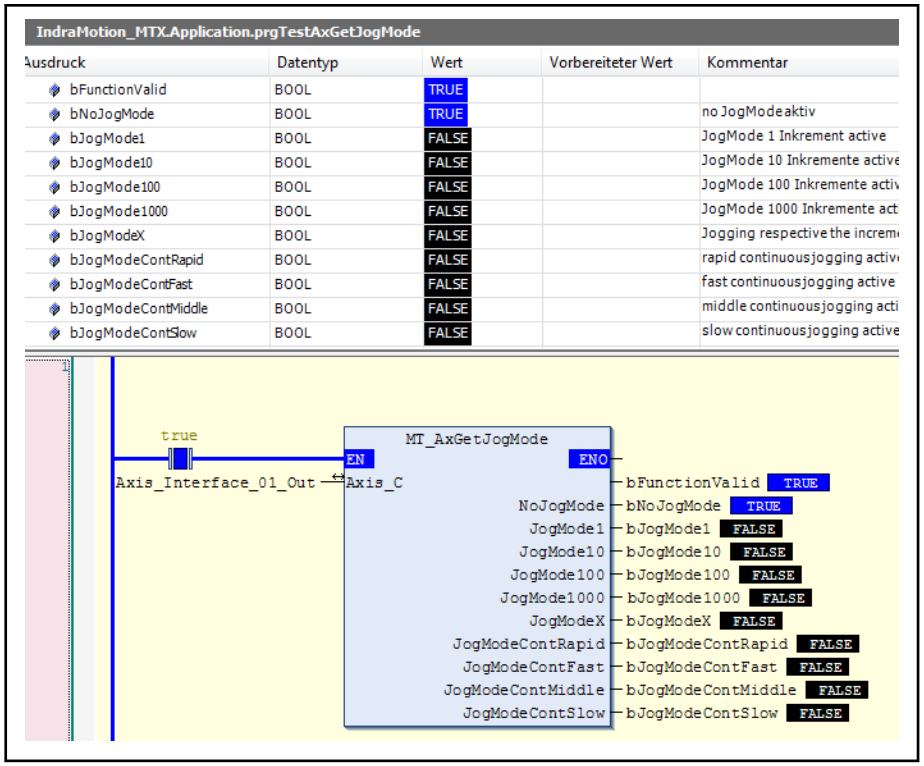


Fig. 14-77: Implementation MT_AxGetJogMode

14.4.7 MT_AxSetJogMode, Changing the JogMode

Brief description

Function block to change the JogMode (jog distance).

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-58: Library names according to the target systems

Interface description

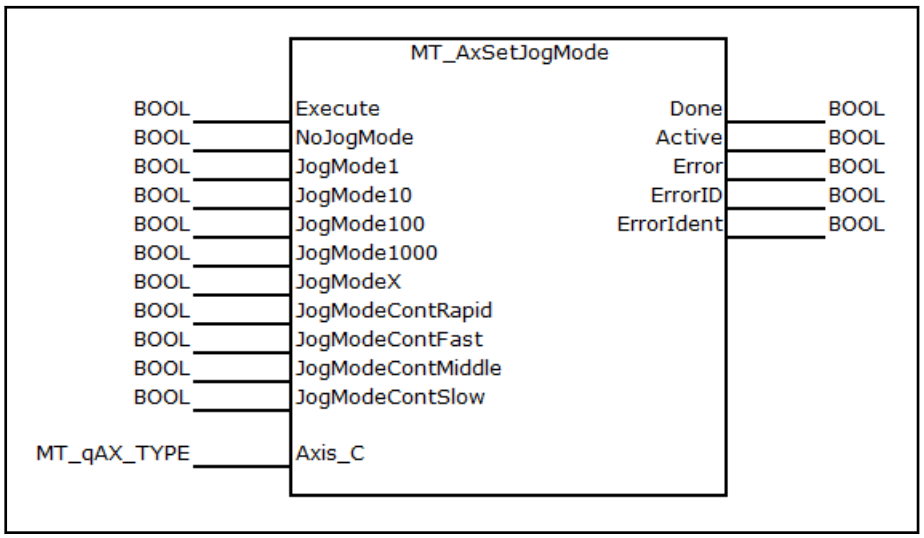


Fig. 14-78: Function block MT_AxSetJogMode

	Name	Type	Description
VAR_INPUT	Execute	BOOL	Setting the preselected jog mode
	NoJogMode	BOOL	No jog mode
	JogMode1	BOOL	JogMode, 1 increment
	JogMode10	BOOL	JogMode, 10 increments
	JogMode100	BOOL	JogMode 100 increments
	JogMode1000	BOOL	JogMode 1000 increments
	JogModeX	BOOL	JogMode for jogging according to the increment specifications in the NC
	JogModeContRapid	BOOL	Continuous jogging in rapid traverse
	JogModeContFast	BOOL	Continuous jogging, fast
	JogModeContMiddle	BOOL	Continuous jogging, average
	JogModeContSlow	BOOL	Continuous jogging, slow
VAR_OUTPUT	Done	BOOL	Processing successfully completed
	Active	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "bError" = TRUE)
VAR_IN_OUT	Axis_C	MT_qAx_Type	Axis interface: Axis control signals

Tab. 14-59: Inputs/outputs function block MT_AxSetJogMode

Signal-time diagram

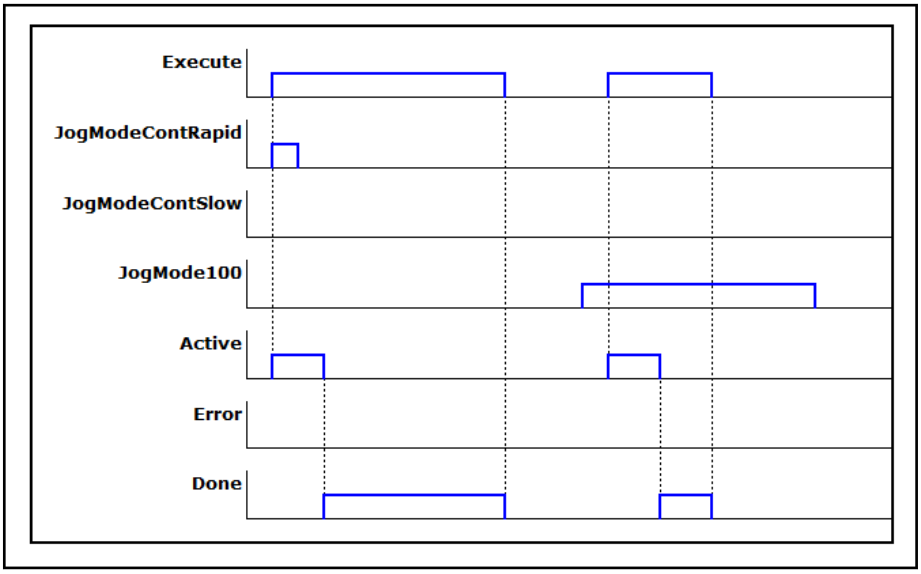


Fig. 14-79: MT_AxSetJogMode signal-time diagram

Left: Setting "Continuous jogging in rapid traverse" completed successfully

Right: Setting "JogMode 100 increments" completed successfully

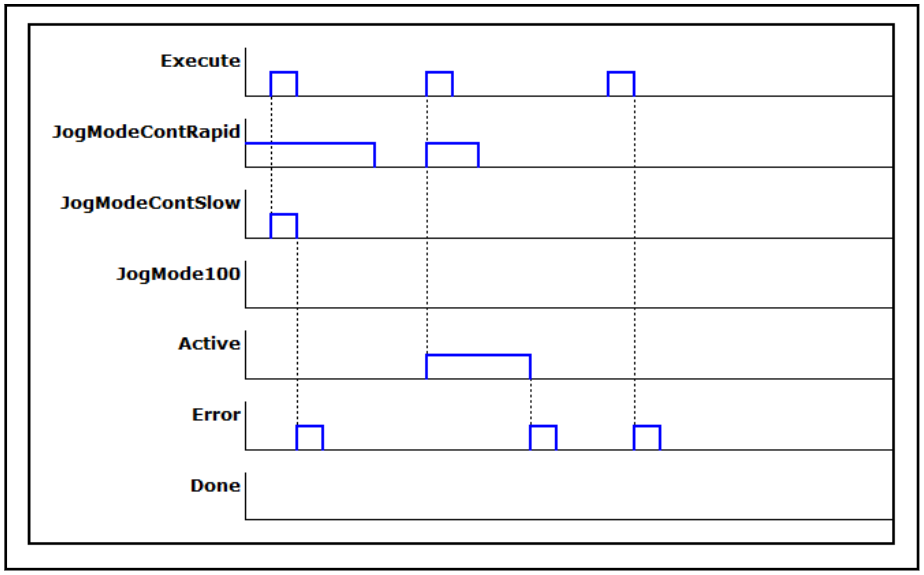


Fig. 14-80: MT_AxSetJogMode signal-time diagram

Error left: More than one JogMode is preselected

Error in the center: Timeout (500ms), no acknowledgement that the JogMode was set

Error right: No operation jog mode was preselected

Functional description

The MTX can be used to continuously jog the axes in four velocities or in increments with five different step widths (machine parameter "AXSP\Dr[x]\AxFun\Jog\Dist" - "MaCoDa no.: 101500002"). The nine JogModes are exclusively preselected already coded by the PLC via the PLC AxisInterface (MT_qAx_Type) by means of the signals "qAx_ManFeed_00..03". These JogModes can be enabled for one axis using the function block "MT_SetJogMode".



This function block works edge-controlled. That means that the function block is executed exactly once after activation of the JogMode change.

A signal change from zero to one (0 → 1) at "bExecute" enables the function block and backs up the state of the inputs at this point in time. Only the input "Axis_C" is cyclically applied as long as the function is active ("Active"==TRUE). When changing the signal, the desired JogMode has to be preselected (e.g. "JogModeContRapid"). If no JogMode or multiple ones are preselected when setting "Execute", the function block goes into error state ("Error" becomes TRUE). As soon as the "Done" output becomes TRUE, setting the desired JogMode is successfully completed. If the JogMode could not be changed, this is displayed by the "Error" output. As long as "Error" is TRUE, there is a valid error code under "ErrorIdent". Before switching the JogMode again or deleting the error, "Execute" has to be set to FALSE. Resetting "Execute" deletes "Error" and "Done". As long as the outputs "Done", "Active" or "Error" are TRUE, signal changes at the inputs to preselect the JogMode (e.g. "JogModeContRapid") have no effect. The outputs "Done", "Active" and "Error" exclude each other. That means that only one is TRUE at a time.

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_SetJogMode"

Error code (hex)	Description
0x00000000	No error code available
0x00010001	Invalid input assignment. No JogMode or several JogModes were preselected
0x00040201	Changing the interface signals "qSp_ManSpeed_00..03" failed. The interfaces might have been overwritten outside of the function block
0x 00050001	Internal error

Tab. 14-60: Error of "MT_AxSetJogMode" from "ErrorIdent.Additional1" if "Error"==TRUE

Implementation example

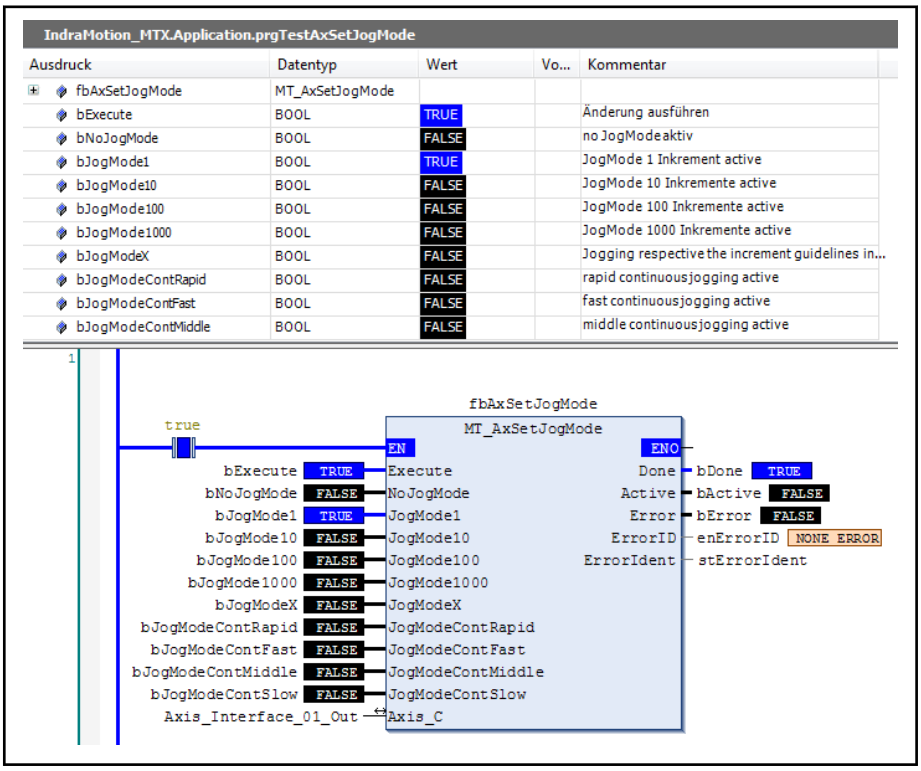


Fig. 14-81: Implementation MT_AxSetJogMode, changing JogMode completed successfully

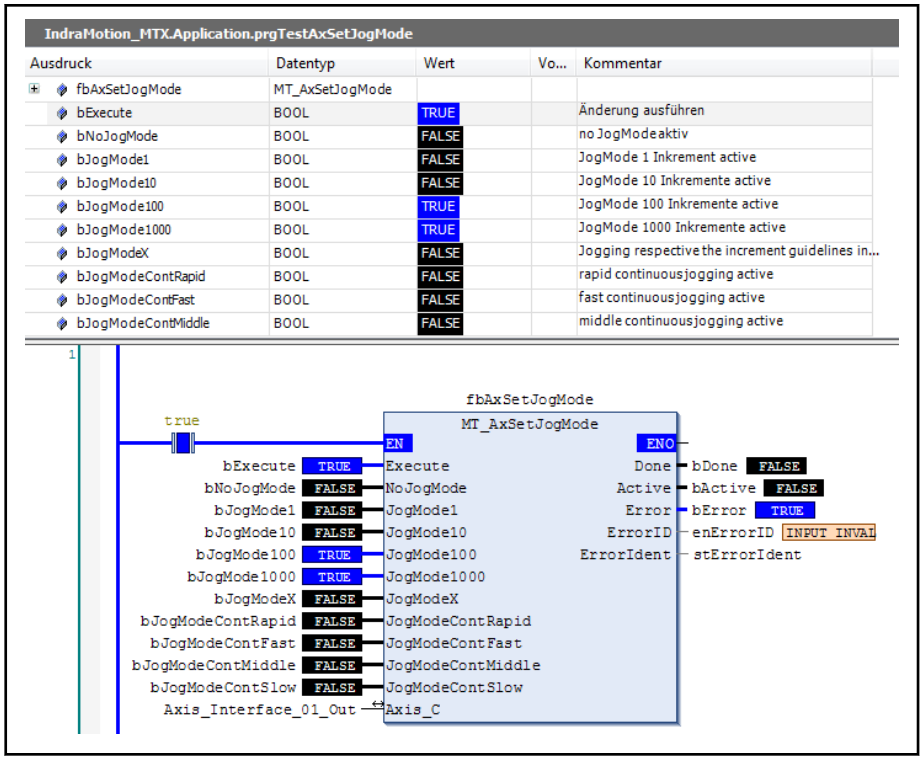


Fig. 14-82: Implementation MT_AxSetJogMode, changing JogMode failed (no JogMode preselected)

14.4.8 MT_AxJogMode, Changing and determining the active jog distance

Brief description

Function block to determine the currently active JogMode (jog distance) as well as for setting a new JogMode.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-61: Library names according to the target systems

Interface description

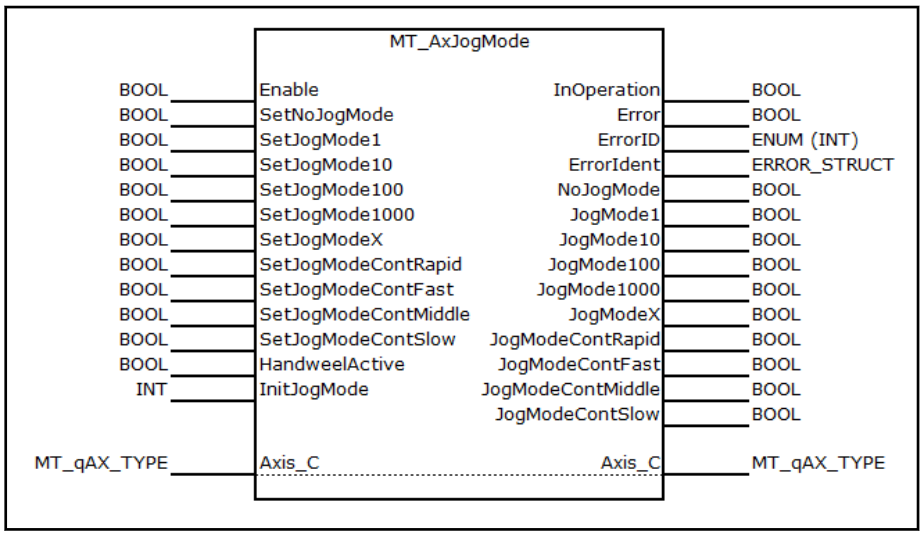


Fig. 14-83: Function block MT_AxJogMode

	Name	Type	Description
VAR_INPUT	Enable	BOOL	Enable function block
	SetNoJogMode	BOOL	No jog mode
	SetJogMode1	BOOL	Jog mode, 1 increment
	SetJogMode10	BOOL	Jog mode, 10 increments
	SetJogMode100	BOOL	Jog mode 100 increments
	SetJogMode1000	BOOL	JogMode 1000 increments
	SetJogModeX	BOOL	JogMode for jogging according to the increment specifications in the NC
	SetJogModeContRapid	BOOL	Continuous jogging in rapid traverse
	SetJogModeContFast	BOOL	Continuous jogging, fast
	SetJogModeContMiddle	BOOL	Continuous jogging, average
	SetJogModeContSlow	BOOL	Continuous jogging, slow
	HandweelActive	BOOL	Handwheel active if TRUE only possible in incremental JogModes
	InitJogMode	BOOL	Preselection of the JogMode when activating the function block
VAR_OUTPUT	InOperation	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "Error" = TRUE)
	NoJogMode	BOOL	No JogMode active
	JogMode1	BOOL	1 increment jogging active
	JogMode10	BOOL	10 increments jogging active
	JogMode100	BOOL	100 increments jogging active
	JogMode1000	BOOL	1000 increments jogging active
	JogModeX	BOOL	x increments jogging active
	JogModeContRapid	BOOL	Continuous jogging in rapid traverse
	JogModeContFast	BOOL	Continuous jogging fast velocity
	JogModeContMiddle	BOOL	Continuous jogging average velocity
	JogModeContSlow	BOOL	Continuous jogging slow velocity
VAR_IN_OUT	Axis_C	MT_qAx_Type	Axis interface: Axis control signals

Tab. 14-62: Inputs/outputs function block MT_AxJogMode

Signal-time diagram

The function shows the signal behavior of a permanently state-controlled function block, see chapter14.2 Standard interfaces at function blocks, page 225.

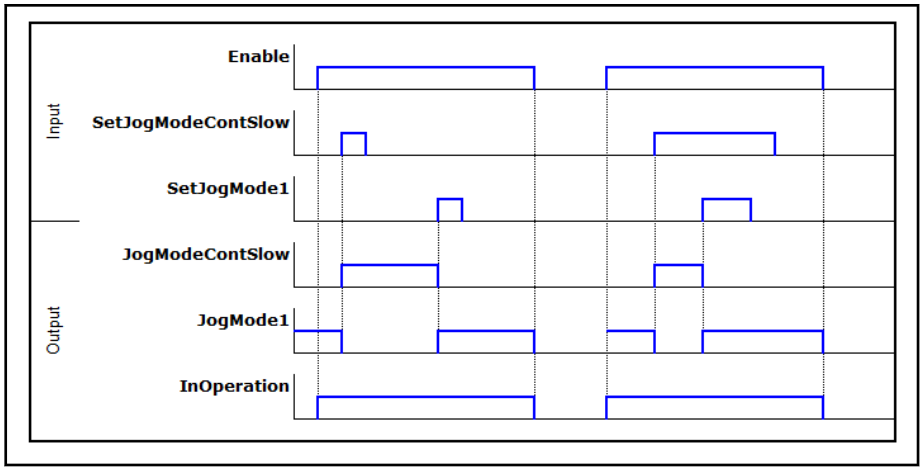


Fig. 14-84: Normal behavior when changing the jog distance

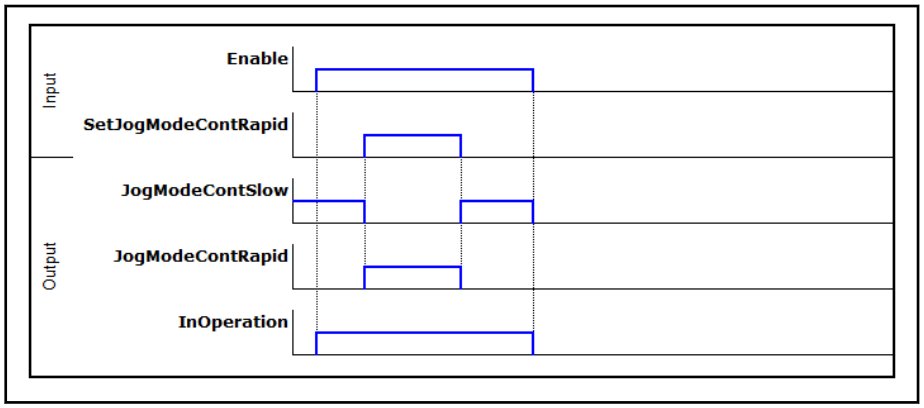


Fig. 14-85: Function block behavior when setting "SetJogModeContRapid"

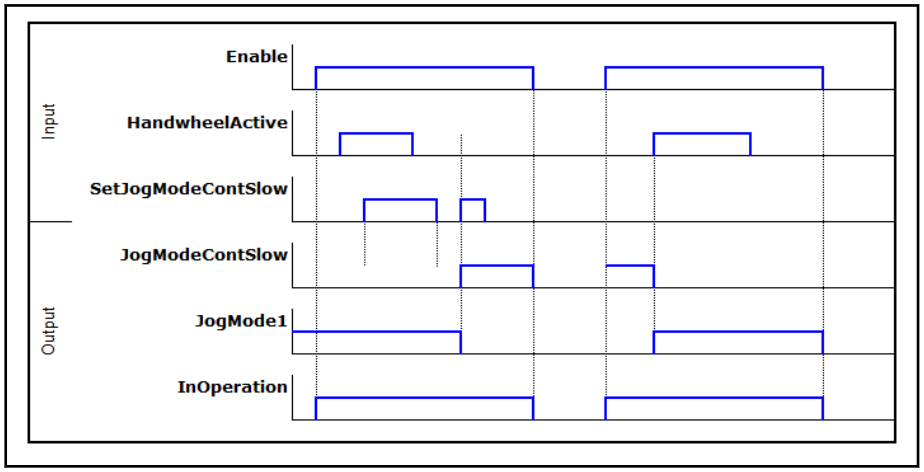


Fig. 14-86: Function block behavior when setting "HandweelActive"

Functional description

This function block combines and extends the MT_AxGetJogMode and MT_AxSetJogMode functionalities.

The MTX can be used to continuously jog the axes in four velocities or in increments with five different step widths (machine parameter "AXSP\Dr[x]\AxFun\Jog\Dist" - "MaCoDa no.: 101500002"). The nine JogModes are exclu-

sively preselected already coded by the PLC via the PLC AxisInterface (MT_qAx_Type) by means of the signals "qAx_ManFeed_00..03". These JogModes can be enabled for one axis using the function block MT_SetJog-Mode. These JogModes can be activated for an axis and the current status can be read out using this function block.

As soon as "Enable" is TRUE, the current JogMode is determined upon each function block call from the connected AxisInterface. Upon a signal change (0-->1) at one of the "SetJog... inputs", the JogMode is changed. If several "SetJog... inputs" are TRUE, the last set input is active. If the JogMode could not be changed, this is displayed by the "Error" output. As long as "Error" is TRUE, there is a valid error code under "ErrorIdent". "Enable" has to be FALSE to delete errors.

Special features:

The "SetJogModeContRapid" input is handled separately. If this JogMode is selected it is only active as long as the input is TRUE. As soon as the input is FALSE again, the previously set JogMode is automatically set.

The "InitJogMode" input facilitates setting a JogMode once upon activating the FB without one of the inputs of the "SetJog... inputs" being TRUE.

The following specifications are valid:

InitJogMode	JogMode
0	No initialization
1	NoJogMode
2	JogMode1
3	JogMode10
4	JogMode100
5	JogMode1000
6	JogModeX
7	JogModeContRapid
8	JogModeContFast
9	JogModeContMiddle
10	JogModeContSlow

Tab. 14-63: Initialization JogMode

As axis motions with the handwheel are only possible with incremental JogModes as long as the "HandwheelActive" input is TRUE, only incremental JogModes can be set. If one of the continuous JogModes is active at the time when "HandweelActive" becomes TRUE (0-->1), it is automatically changed to "JogMode1".

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_SetJogMode"

Error code (hex)	Description
0x00000000	No error code available
0x00010001	Invalid input assignment. No JogMode or several JogModes were preselected
0x00040201	Changing the interface signals "qSp_ManSpeed_00..03" failed. The interfaces might have been overwritten outside of the function block
0x 00050001	Internal error

Tab. 14-64: Error of "MT_AxSetJogMode" from "ErrorIdent.Additional1" if "Error"==TRUE

Implementation example

```
VAR
    fbAxJogMode             :MT_AxJogMode;
    //Input
    bEnable                 :BOOL:=FALSE;   (*FB enable*)
    SetNoJogMode            :BOOL:=FALSE;   (*target Jogmode: no JogMode*)
    (*incremental JogModi*)
    SetJogMode1             :BOOL:=FALSE;   (*target Jogmode: 1 increment*)
    SetJogMode10            :BOOL:=FALSE;   (*target Jogmode: 10 increment*)
    SetJogMode100           :BOOL:=FALSE;   (*target Jogmode: 100 increment*)
    SetJogMode1000          :BOOL:=FALSE;   (*target Jogmode: 1000 increment*)
    SetJogModeX             :BOOL:=FALSE;   (*target JogMode: Jogging respective the increment guidelines in the machine parameter*)
    (*continuous JogModi*)
    SetJogModeContRapid     :BOOL:=FALSE;   (*target JogMode: rapid continuous jogging*)
    SetJogModeContFast      :BOOL:=FALSE;   (*target JogMode: fast continuous jogging*)
    SetJogModeContMiddle    :BOOL:=FALSE;   (*target JogMode: middle continuous jogging*)
    SetJogModeContSlow      :BOOL:=FALSE;   (*target JogMode: slow continuous jogging*)

    HandweelActive          :BOOL:=FALSE;   (*Hand wheel is active*)
    InitJogMode             :INT:=8;        (*Set JogMode in the first PLC cycle*)
    //Output
    InOperation             :BOOL:=FALSE;   (*process *)
    bError                  :BOOL:=FALSE;   (*process incorrect aborted, 'ErrorID' and 'ErrorIdent' valid*)
    enErrorID               :ERROR_CODE;    (*basic error information, only valid when 'Error'==TRUE*)
    stErrorIdent            :ERROR_STRUCT;  (*detailed error information, only valid when 'Error'==TRUE*)
    bNoJogMode              :BOOL:=FALSE;   (*no JogMode aktiv*)
    (*Inkrementelle JogModi*)
    bJogMode1               :BOOL:=FALSE;   (*JogMode 1 Inkrement active*)
    bJogMode10              :BOOL:=FALSE;   (*JogMode 10 Inkremente active*)
    bJogMode100             :BOOL:=FALSE;   (*JogMode 100 Inkremente active*)
    bJogMode1000            :BOOL:=FALSE;   (*JogMode 1000 Inkremente active*)
    bJogModeX               :BOOL:=FALSE;   (*Jogging respective the increment guidelines in the machine parameter active*)
    (*Kontinuierliche JogModi*)
    bJogModeContRapid       :BOOL:=FALSE;   (*rapid continuous jogging active*)
    bJogModeContFast        :BOOL:=FALSE;   (*fast continuous jogging active*)
    bJogModeContMiddle      :BOOL:=FALSE;   (*middle continuous jogging active*)
    bJogModeContSlow        :BOOL:=FALSE;   (*slow continuous jogging active*)
END_VAR
```

Fig. 14-87: Declaration MT_AxJogMode

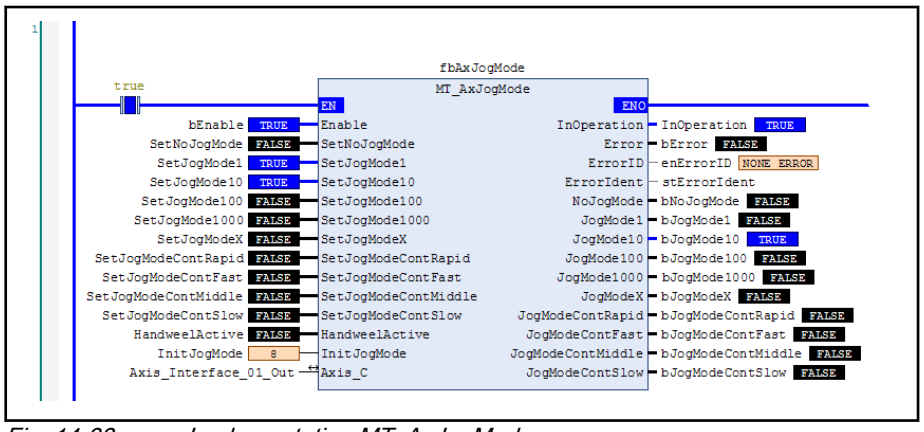


Fig. 14-88: Implementation MT_AxJogMode

14.4.9 MT_AxAsyncGetOPMode, Determining the operation mode of an asynchronous axis

Brief description

Function to determine the active operation mode of an asynchronous axis.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-65: Library names according to the target systems

Interface description

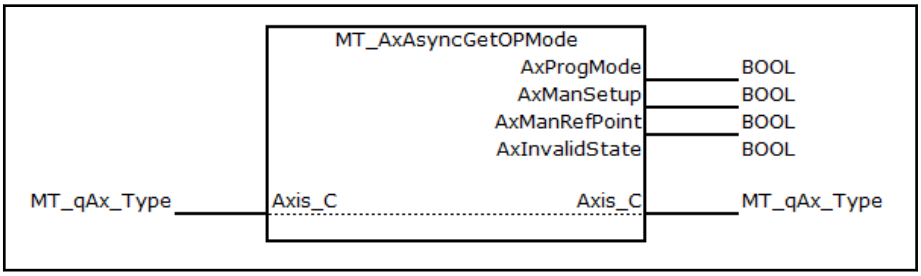


Fig. 14-89: MT_AxAsyncGetOPMode function block

	Name	Type	Description
VAR_INPUT	Axis_C	MT_qAX_TYPE	Axis interface: Axis control signals
VAR_OUTPUT	AxProgMode	BOOL	Programming mode
	AxManSetup	BOOL	Manual setup (jogging)
	AxManRefPoint	BOOL	Setup, approach reference point
	AxInvalidState	BOOL	Unknown operation mode

Tab. 14-66: Inputs/outputs function block MT_AxAsyncGetOPMode

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

The MTX allows three operation modes for each asynchronous axis. These states are represented as bit-coded via the AxisInterface (MT_qAx_Type) by the "qAx_OpModeSel_00 .. 01" signals. Only one main operation mode is active at a time. The function decodes the interface signals and displays the active operation mode via the corresponding outputs.



The operation modes displayed by the function are only relevant for the axis if this axis is asynchronous and thus cannot be assigned to a channel. If the function "MT_AxGetChIndex" returns the value ZERO, the axis is not assigned to any channel and is thus asynchronous.

Troubleshooting

No error handling required for this function.

Implementation example

The operation mode of the an axis is determined in the example program.

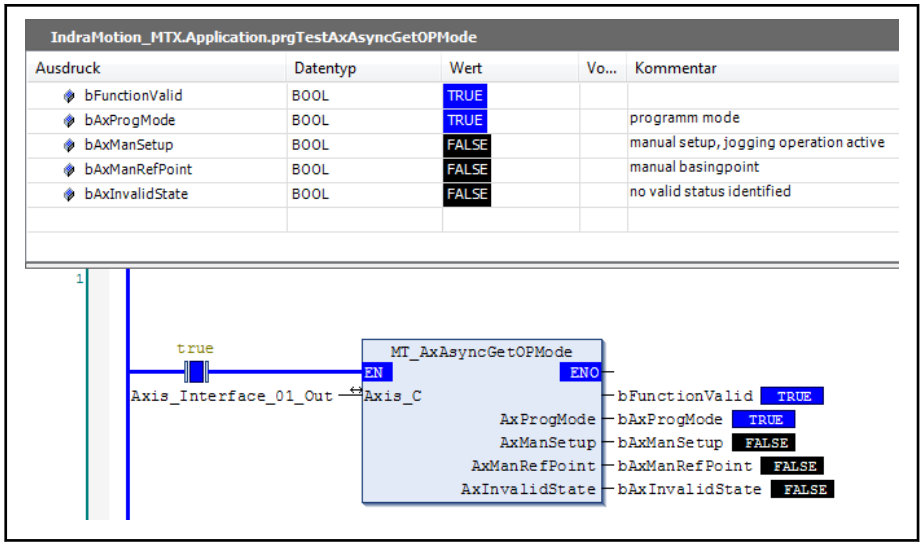


Fig. 14-90: Implementation MT_AxAsyncGetOPMode

14.4.10 MT_AxAsyncOPMode, Changing the operation mode of an asynchronous axis

Brief description

Function to determine the active operation mode of an asynchronous axis.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-67: Library names according to the target systems

Interface description

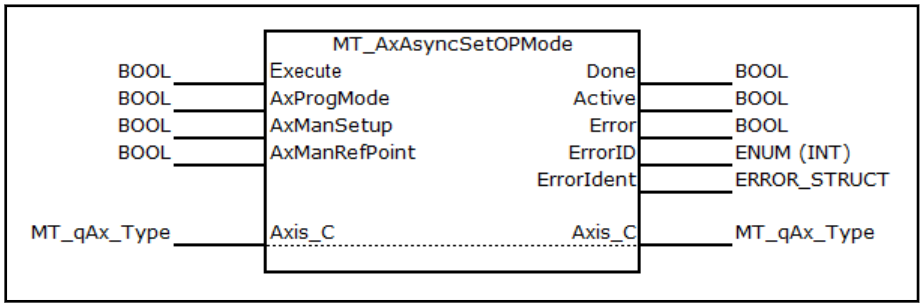


Fig. 14-91: MT_AxAsyncSetOPMode function block

	Name	Type	Description
VAR_INPUT	Execute	BOOL	Execute set operation mode
	AxProgMode	BOOL	Programming mode
	AxManSetup	BOOL	Manual setup (jogging)
	AxManRefPoint	BOOL	Setup, approach reference point

	Name	Type	Description
VAR_OUTPUT	Done	BOOL	Processing successfully completed
	Active	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "bError" = TRUE)
VAR_IN_OUT	Axis_C	MT_qAX_TYPE	Axis interface: Axis control signals

Tab. 14-68: Inputs/outputs function block MT_AxAsyncSetOPMode

Signal-time diagram

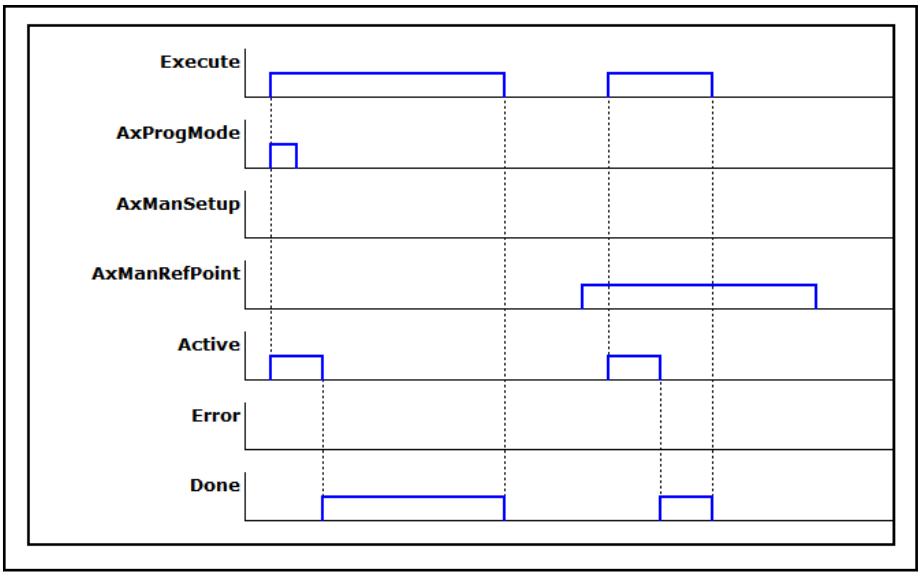


Fig. 14-92: MT_AxAsyncSetOPMode signal-time diagram

Left: Setting the "Programming mode" completed successfully

Right: Setting "Setup, approach reference point" completed successfully

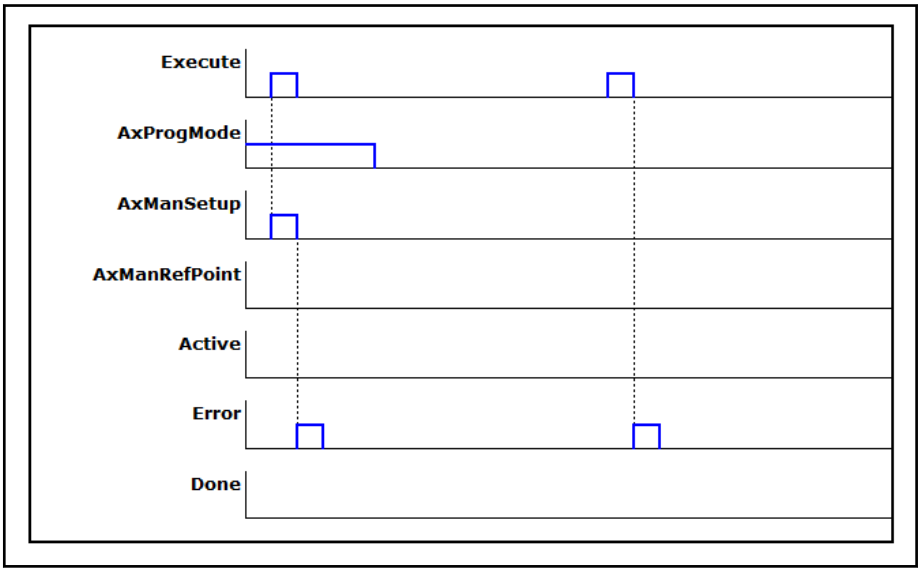


Fig. 14-93: MT_AxAsyncSetOPMode signal-time diagram

Error left: More than one operation mode is preselected

Error right: No operation mode was preselected

Functional description

The MTX allows three operation modes for each asynchronous axis. These states are represented as bit-coded via the AxisInterface (MT_qAx_Type) by the "qAx_OpModeSel_00 .. 01" signals. Upon the rising edge at the Execute input, the inputs are applied once and the corresponding AxisInterface signal is set. As soon as the "Done" output becomes TRUE, setting the desired operation mode is successfully completed. In case an error occurs, "Error" is TRUE. Before changing the operation mode again or deleting the error, "Execute" has to be FALSE. Resetting "Execute" deletes "Error" and "Done". The outputs "Done", "Active" and "Error" exclude each other. That means that only one output is TRUE at a time.

This function block works edge-controlled. That means that after the activation, setting the operation mode "qAx_OpModeSel_00 .. 01" executed exactly once.

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_AxAsyncSetOPMode"

Error code (hex)	Description
0x00000000	No error code available
0x00010402	Invalid input assignment. No operation mode or several operation modes were preselected
0x00040101	Setting the AxisInterface signal (qAx_OpModeSel) failed as the signal was reset outside the FB

Tab. 14-69: Error of "MT_AxAsyncSetOPMode", from"ErrorIdent.Additional1", if "Error"==TRUE

Implementation example

In the example program, the operation mode of an asynchronous axis was changed to "Setup manually".

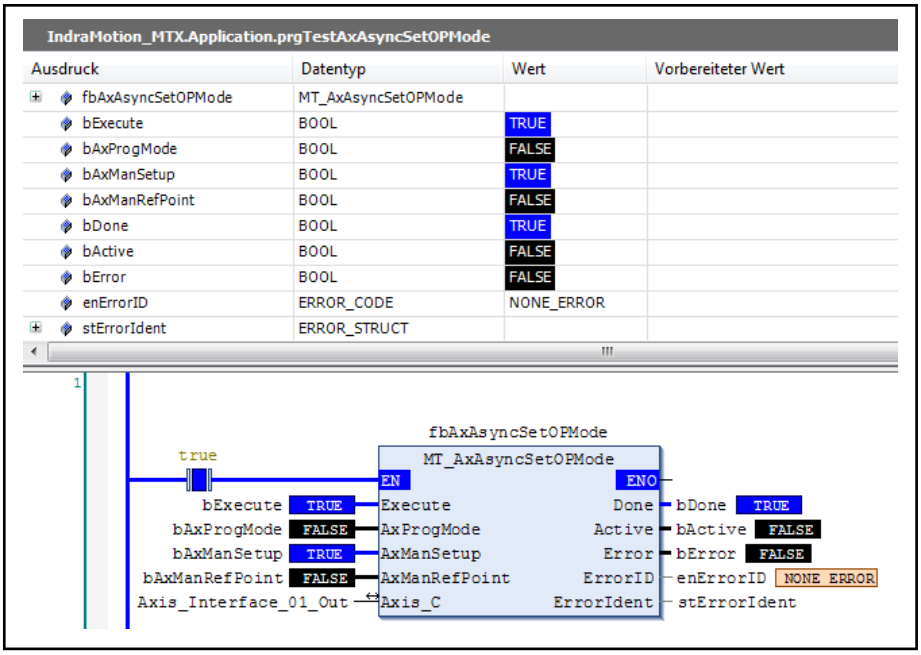


Fig. 14-94: Implementation MT_AxAsyncSetOPMode

14.4.11 MT_AxAsyncOPMode, operation mode of an asynchronous axis

Brief description

Function to determine the active operation mode of an asynchronous axis.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-70: Library names according to the target systems

Interface description

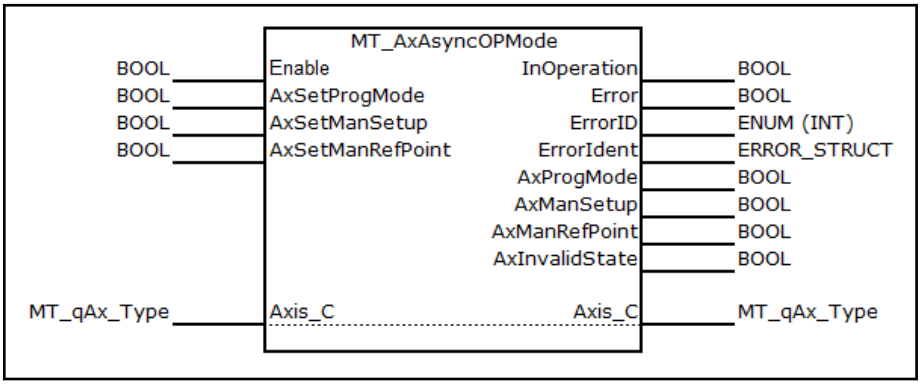


Fig. 14-95: Function block MT_AxAsyncOPMode

	Name	Type	Description
VAR_INPUT	Enable	BOOL	Enable function block
	AxSetProgMode	BOOL	Programming mode
	AxSetManSetup	BOOL	Manual setup (jogging)
	AxSetManRefPoint	BOOL	Setup, approach reference point

	Name	Type	Description
VAR_OUTPUT	AxProgMode	BOOL	Programming mode
	AxManSetup	BOOL	Manual setup (jogging)
	AxManRefPoint	BOOL	Setup, approach reference point
	AxInvalidState	BOOL	Unknown operation mode
VAR_IN_OUT	Axis_C	MT_qAX_TYPE	Axis interface: Axis control signals

Tab. 14-71: Inputs/outputs function block MT_AxAsyncOPMode

Signal-time diagram

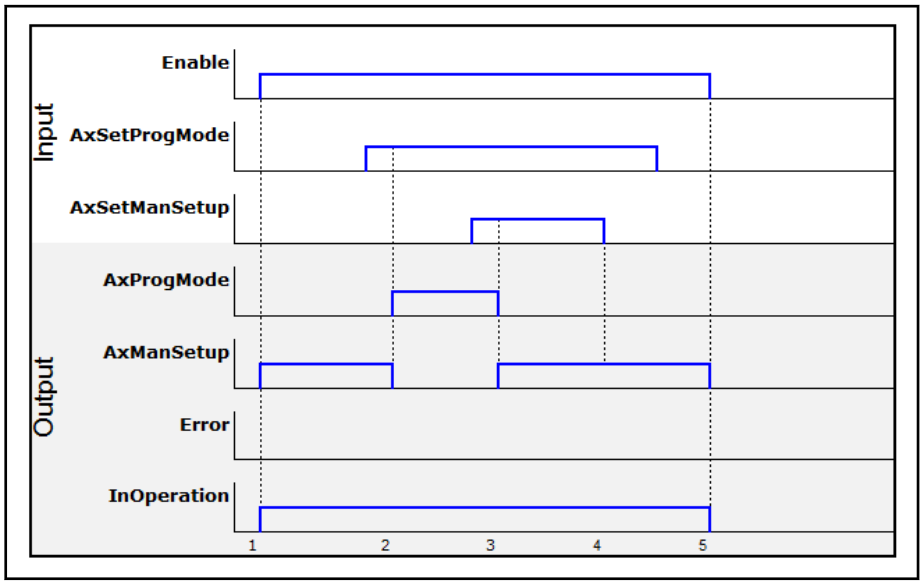


Fig. 14-96: MT_AxAsyncOPMode signal-time diagram

1. AxManSetup is active
2. One cycle after switching, AxProgMode is active
3. Last signal change at AxSetManSetup, thus, AxManSetup active
4. AxManSetup remains active, even if the input is FALSE
5. All outputs FALSE, if Enable FALSE

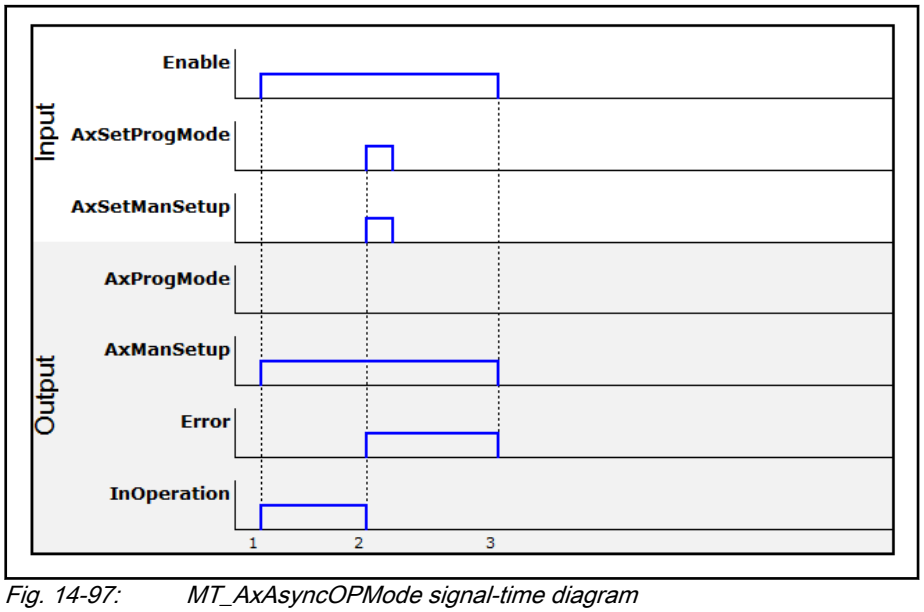


Fig. 14-97: MT_AxAsyncOPMode signal-time diagram

1. AxManSetup is active
2. Error - Simultaneous signal change (0-->1) at two AxSet... inputs
3. Deleting errors by resetting Enable

Functional description

The MTX allows three operation modes for each asynchronous axis. These states are represented as bit-coded via the AxisInterface (MT_qAx_Type) by the "qAx_OpModeSel_00 .. 01" signals. Only one main operation mode is active at a time. The function block decodes the interface signals, shows the active operation mode and facilitates the operation mode change.

☞ The active operation mode displayed by the function is only relevant for the axis if this axis is asynchronous and thus is not assigned to a channel. If the function "MT_AxGetChIndex" returns the value ZERO, the axis is not assigned to any channel and is thus asynchronous.

The function block is active as long as the "Enable" input is TRUE. The state is displayed via the "InOperation" output. This is indicated by the "Error" output. As long as "Error" is TRUE, there is a valid error code under the "ErrorIdent" output. Set "Enable" to FALSE to delete the error. All inputs are applied upon each function block call. Thus, the current operation mode is determined from the interface signal "qAx_OpModeSel_00 .. 01" upon each function block call The operation mode is changed by a signal change (0-->1) at the corresponding operation mode input and becomes active after one PLC cycle. If several inputs are TRUE, the input at which a signal change (0-->1) was detected last is used to change the operating mode.

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_SetPowerOn“

Error code (hex)	Description
0x00000000	No error code available
0x00010402	Invalid input assignment. No operation mode or several operation modes were preselected
0x00040101	Setting the AxisInterface signal (qAx_OpModeSel) failed as the signal was reset outside the FB

Tab. 14-72: Error of "MT_SetPowerOn" of "ErrorIdent.Additional1", if Error == TRUE

Implementation example

In the example program, the operation mode was changed to AxManRefPoint.

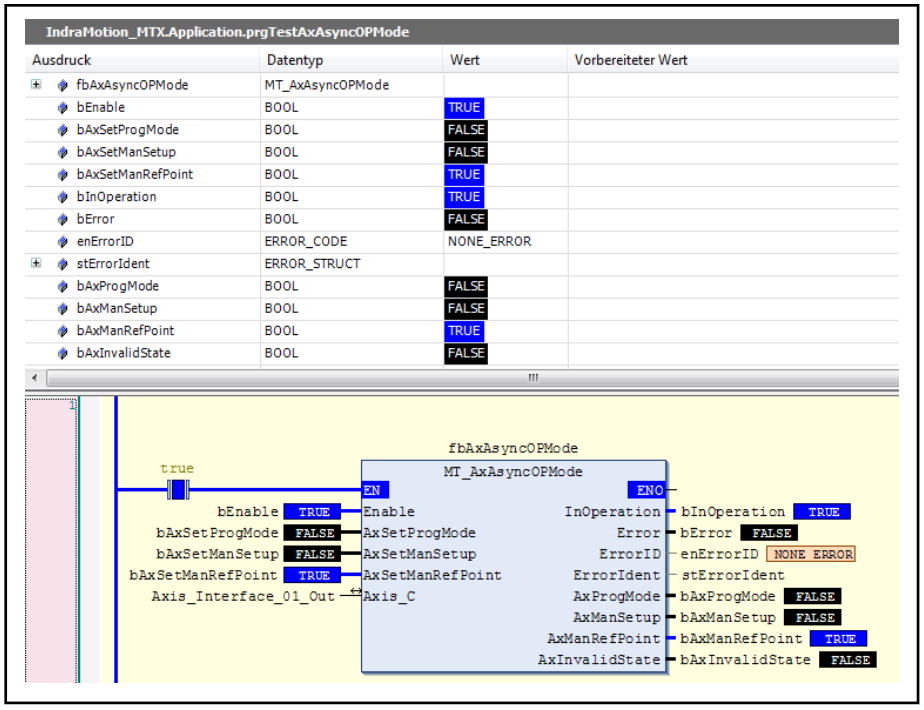


Fig. 14-98: Implementation MT_AxAsyncOPMode

14.4.12 MT_AxAsyncSetOverride, Changing the feed for an asynchronous axis

Brief description

Function to set the feed value for an asynchronous axis.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-73: Library names according to the target systems

Interface description

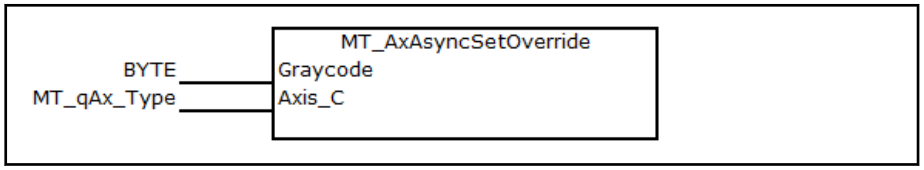


Fig. 14-99: MT_AxAsyncSetOverride function block

	Name	Type	Description
VAR_INPUT	Graycode	BYTE	Override value as gray code
VAR_IN_OUT	Axis_C	MT_qAx_Type	Axis interface: Axis control signals

Tab. 14-74: Inputs/outputs function block MT_AxAsyncSetOverride

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

The axis feed value has to transmitted for each asynchronous axis from the PLC to the NC via its axis interface (MT_qAx_Type) using the "qAx_Override" signal. Using this function, the value at the "Graycode" input is changed and transmitted to the connected AxisInterface.

Troubleshooting

No error handling required for this function.

Implementation example

Example program for changing the override for the first axis.

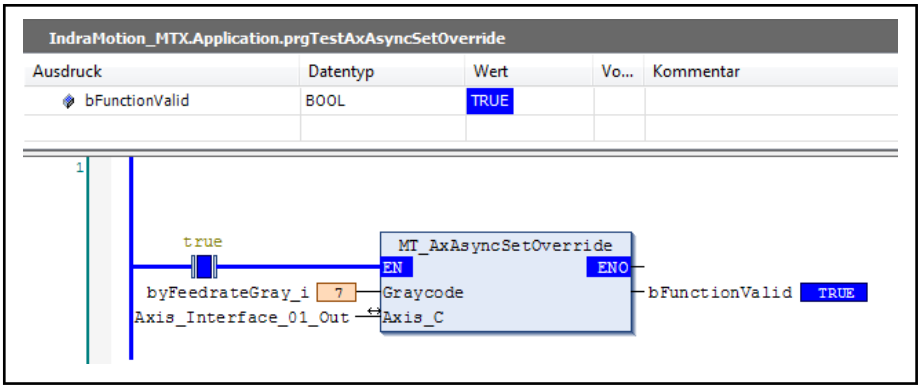


Fig. 14-100: Implementation MT_AxAsyncSetOverride

14.4.13 MT_AxGetPowerState, Determining the drive status

Brief description

Function to determine the active drive status of an axis.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-75: Library names according to the target systems

Interface description

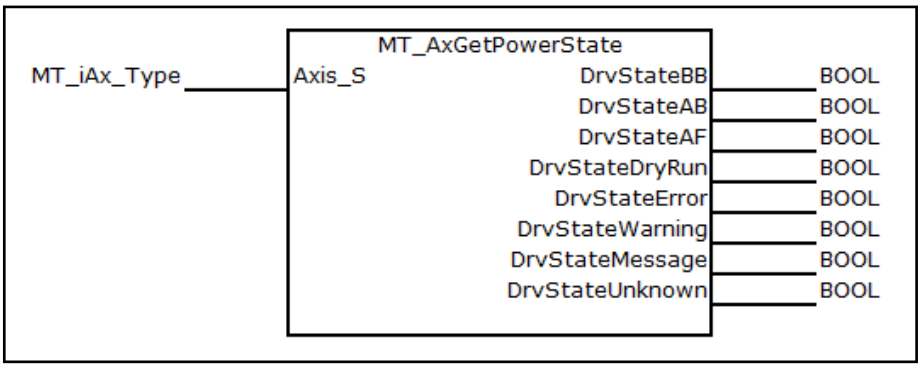


Fig. 14-101: Function block MT_AxGetPowerState

	Name	Type	Description
VAR_INPUT	Axis_S	MT_iAX_TYPE	Axis interface: Axis status signals
VAR_OUTPUT	DrvStateBB	BOOL	Drive ready
	DrvStateAB	BOOL	Drive ready for controller enable
	DrvStateAF	BOOL	Drive in control
	DrvStateDryRun	BOOL	Drive in virtual mode
	DrvStateError	BOOL	Drive reports an error
	DrvStateWarning	BOOL	Drive reports a warning
	DrvStateMessage	BOOL	Drive reports a note

Tab. 14-76: Inputs/outputs function block MT_AxGetPowerState

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

This function determines the performance from the AxisInterface (MT_iAx_Type) and displays potential errors, warnings or information at the drive.

The following Interface signals are interpreted: , , ,

- iAx_DrvPower
- iAx_DrvReady
- iAx_DrvAct
- iAx_DryRun
- iAx_DrvErrClass1
- iAx_DrvChangeClass2
- iAx_DrvChangeClass3

Troubleshooting No error handling required for this function.

Implementation example The axis status is determined in the example program.

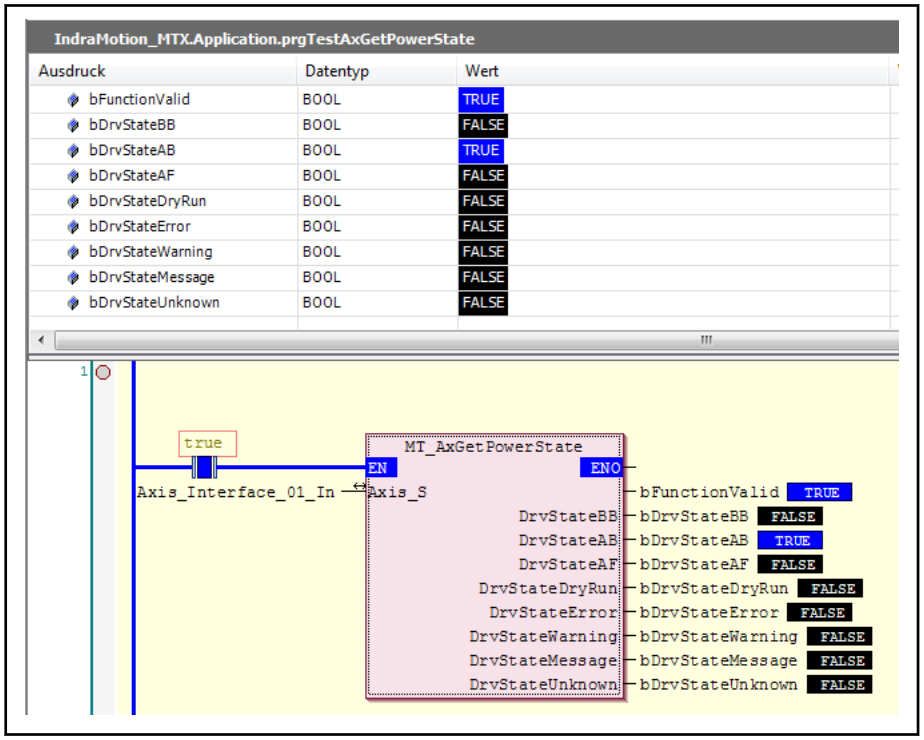


Fig. 14-102: Implementation MT_AxGetPowerState

14.4.14 MT_AxGetPowerStateExt, drive status incl. Determining errors and warnings

Brief description Function to determine the active drive status of an axis with additional information about errors and warnings.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-77: Library names according to the target systems

Interface description

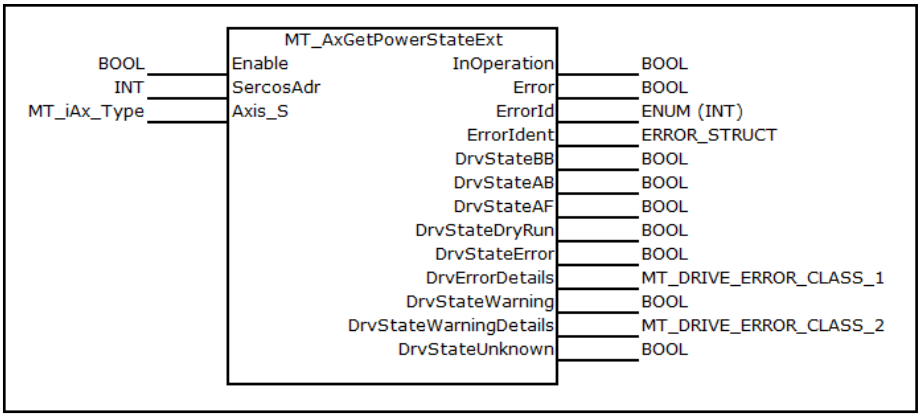


Fig. 14-103: MT_AxGetPowerStateExt function block

	Name	Type	Description
VAR_INPUT	Axis_S	MT_iAX_TYPE	Axis interface: Axis status signals
VAR_OUTPUT	DrvStateBB	BOOL	Drive ready
	DrvStateAB	BOOL	Drive ready for controller enable
	DrvStateAF	BOOL	Drive in control
	DrvStateDryRun	BOOL	Drive in virtual mode
	DrvStateError	BOOL	Drive reports an error
	DrvStateErrorDetails	MT_DRIVE_ERROR_CLASS_1	Detail information about the error (S-0-0011)
	DrvStateWarning	BOOL	Drive reports a warning
	DrvStateWarningDetails	MT_DRIVE_ERROR_CLASS_2	Detailed information on the warning (S-0-0012)

Tab. 14-78: Inputs/outputs function block MT_AxGetPowerStateExt

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

This function block determines the performance from the AxisInterface (MT_iAx_Type) and displays potential errors or warnings at the drive. Additionally, detailed information is output from the drive parameters S-0-0011 or S-0-0012 in case of pending errors or warnings.

The following Interface signals are interpreted:

- iAx_DrvPower
- iAx_DrvReady
- iAx_DrvAct
- iAx_DryRun
- iAx_DrvErrClass1
- iAx_DrvChangeClass2
- iAx_DrvChangeClass3

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table.

The function block displays errors of the internally used function block "IL_SIIISvcRead" of the "RIL_SercosIII" library.

Implementation example

The axis status is determined in the example program in case of an error.

14.4.15 MT_AxSetPowerOn, Setting the controller enable for an axis

Brief description

Function block to activate the controller enable for an axis.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-79: Library names according to the target systems

Interface description

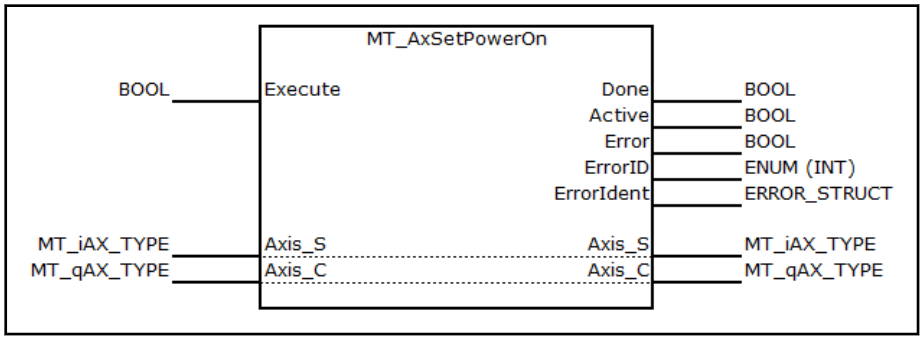


Fig. 14-104: Function block MT_AxSetPowerOn

	Name	Type	Description
VAR_INPUT	Execute	BOOL	Sets the controller enable for an axis
VAR_OUTPUT	Done	BOOL	Processing successfully completed
	Active	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "bError" = TRUE)
VAR_IN_OUT	Axis_S	MT_iAx_Type	Axis interface: Axis status signals
	Axis_C	MT_qAx_Type	Axis interface: Axis control signals

Tab. 14-80: Inputs/outputs function block MT_AxSetPowerOn

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

The controller enable (AF) for an axis is set in the AxisInterface (MT_qAx_Type) via the axis control signal "qAx_DrvOn" in MTX. The function block checks if the axis is ready for controller enable, sets the signal and monitors if feedback is send within 5 seconds that the axis is in control via the axis status signals (MT_iAx_Type).



This function block works edge-controlled. That means that setting the controller enable "qAx_DrvOn" is executed exactly once after activation.

A signal change from zero to one (0 -> 1) at "Execute" enables the function block. Only the inputs "Axis_S" and "Axis_C" are cyclically applied until the function is active ("Active"==TRUE). As soon as the "Done" output becomes TRUE, setting the controller enable is successfully completed. If the controller enable could not be set for the drive, this is displayed by the "Error" output. In case of an error, the interface signal "qAx_DrvOn" is reset. As long as "Error" is TRUE, there is a valid error code under "ErrorIdent". Before setting the controller enable or deleting the error, "Execute" has to be set to FALSE. Resetting "Execute" deletes "Error" and "Done". The outputs "Done", "Active" and "Error" exclude each other. That means that only one is TRUE at a time.

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_SetPowerOn"

Error code (hex)	Description
0x00000000	No error code available
0x00060102	Drive not ready for controller enable, signal status (iAx_DrvReady), the drive has to be in status AB
0x00040101	Setting the AxisInterface signal (qAx_DrvOn) failed as the signal was reset outside the FB
0x00020101	Controller enable was required but no feedback was sent to the drive (axis interface signal: iAx_DrvAct). Maybe, the wrong AxisInterface was connected to the function block

Tab. 14-81: Error of "MT_SetPowerOn" of "ErrorIdent.Additional1", valid if "Error"==TRUE

Implementation example

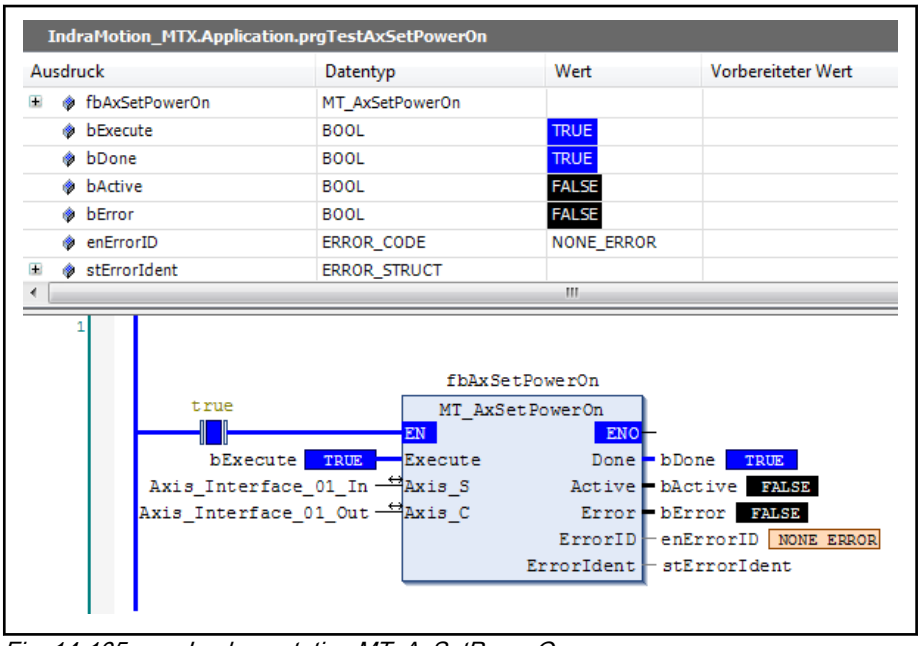


Fig. 14-105: Implementation MT_AxSetPowerOn

14.4.16 MT_AxSetPowerOff, Setting the controller enable for an axis

Brief description

Function block to deactivate the controller enable for an axis.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-82: Library names according to the target systems

Interface description

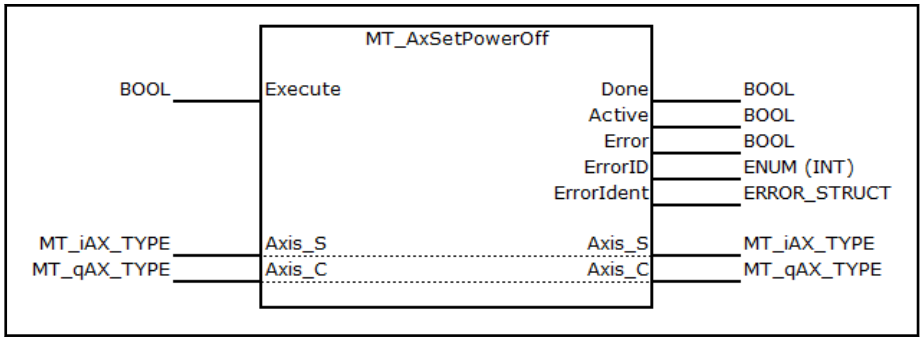


Fig. 14-106: Function block MT_AxSetPowerOff

	Name	Type	Description
VAR_INPUT	Execute	BOOL	Resets controller enable for axis
VAR_OUTPUT	Done	BOOL	Processing successfully completed
	Active	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "bError" = TRUE)
VAR_IN_OUT	Axis_S	MT_iAx_Type	Axis interface: Axis status signals
	Axis_C	MT_qAx_Type	Axis interface: Axis control signals

Tab. 14-83: Inputs/outputs function block MT_AxSetPowerOff

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

The controller enable (AF) for an axis is reset in the NC PLC AxisInterface (MT_qAx_Type) via the axis control signal "qAx_DrvOn". The function block sets the signal to FALSE and monitors if a feedback is send within 5 seconds via the axis status signals (MT_iAx_Type) that the axis is not in control anymore.



This function block works edge-controlled. That means that the function block is executed exactly once after activation of resetting the controller enable "qAx_DrvOn".

A signal change from zero to one (0 -> 1) at "Execute" enables the function block. Only the inputs "Axis_S" and "Axis_C" are cyclically applied until the function is active ("Active"==TRUE). As soon as the "Done" output becomes TRUE, resetting of the controller enable is successfully completed. If the controller enable of the drive could not be reset, this is displayed by the "Error" output. In case of an error, the interface signal "qAx_DrvOn" remains FALSE. As long as "Error" is TRUE, there is a valid error code under "ErrorIdent". Before resetting the controller enable again or deleting the error, "Execute" has to be set to FALSE. Resetting "Execute" deletes "Error" and "Done". The outputs "Done", "Active" and "Error" exclude each other. That means that only one is TRUE at a time.

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_SetPowerOn"

Error code (hex)	Description
0x00000000	No error code available
0x00040101	Resetting the AxisInterface signal (qAx_DrvOn) failed as the signal was reset outside the FB
0x00020101	Resetting the controller enable was requested but no feedback was sent to the drive (axis interface signal: iAx_DrvAct). Maybe, the wrong AxisInterface was probably connected to the function block

Tab. 14-84: Error of "MT_SetPowerOff" of "ErrorIdent.Additional1", valid if "Error"==TRUE

Implementation example

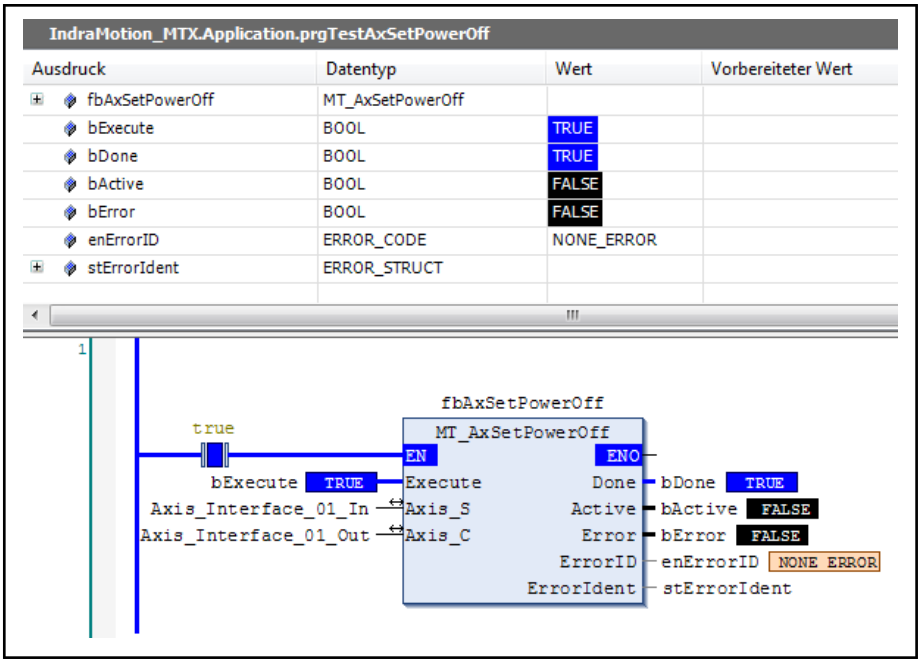


Fig. 14-107: Implementation MT_AxSetPowerOff

14.4.17 MT_AxPower, Controller enable and axis status

Brief description

Function block to activate/deactivate the controller enable as well as for determining the current axis status.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-85: Library names according to the target systems

Interface description

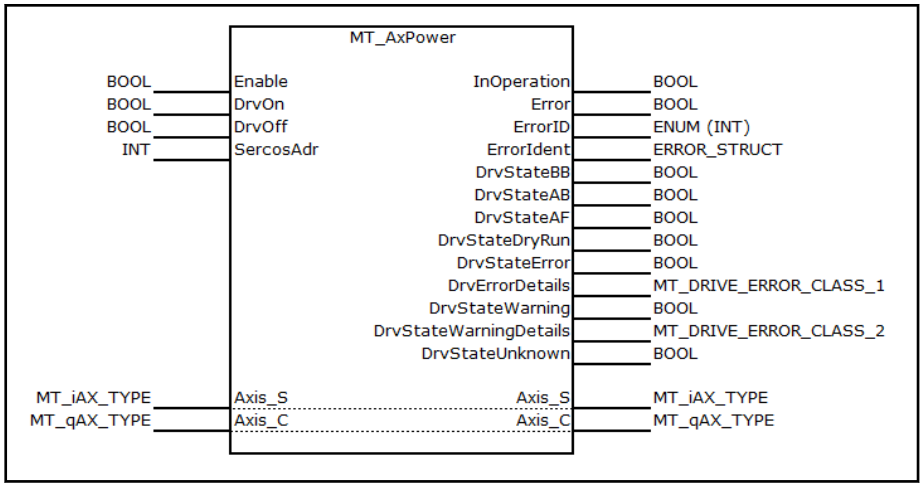


Fig. 14-108: Function block MT_AxPower

	Name	Type	Description
VAR_INPUT	Enable	BOOL	Enable function block
	DrvOn	BOOL	Sets controller enable (DrvOff also has to be TRUE)
	DrvOff	BOOL	Falling edge resets controller enable
	SercosAdr	INT	Sercos address of the drive
VAR_OUTPUT	InOperation	BOOL	Processing successfully completed
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "bError" = TRUE)
	DrvStateBB	BOOL	Drive ready
	DrvStateAB	BOOL	Drive ready for controller enable
	DrvStateAF	BOOL	Drive in control
	DrvStateDryRun	BOOL	Drive in virtual mode
	DrvStateError	BOOL	Drive reports an error
	DrvStateErrorDetails	MT_DRIVE_ERROR_CLASS_1	Detail information about the error (S-0-0011)
	DrvStateWarning	BOOL	Drive reports a warning
	DrvStateWarningDetails	MT_DRIVE_ERROR_CLASS_2	Detailed information on the warning (S-0-0012)
VAR_IN_OUT	Axis_S	MT_iAx_Type	Axis interface: Axis status signals
	Axis_C	MT_qAx_Type	Axis interface: Axis control signals

Tab. 14-86: Inputs/outputs function block MT_AxPower

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

This function block combines the functionality of "MT_AxSetPowerOn", "MT_AxSetPowerOff" and "MT_AxGetPowerStateExt". The function block can control the controller enable of an axis, determine its status cyclically and provided detailed information on warnings and errors.

By setting "Enable", all inputs are read cyclically, the axis status is continuously determined from the AxisInterface and is displayed via the "DrvState..." outputs. As long as "Enable" and "DrvOff" are TRUE, setting of the controller enable for the axis can be requested with a rising edge at "DrvOn". The controller enable is reset via a falling edge at DrvOff. As long as the axis is in state "DrvStateDryRun", "DrvOn" and "DrvOff" are inactive.

The controller enable (AF) is set and reset in the NC PLC AxisInterface (MT_qAx_Type) via the axis control signal "qAx_DrvOn". Prior to setting the controller enable, it is checked if the axis is ready for controller enable (AB). Only then, the signal is set. When setting or resetting the controller enable, it is monitored if a feedback is send within 5 seconds via the axis status signals (MT_iAx_Type).

The function block is executed successfully as long as the "InOperation" output is TRUE. In case of an error, the "Error" output indicates the error and "InOperation" is set to FALSE. In case of an error, the interface signal "qAx_DrvOn" is reset. As long as "Error" is TRUE, there is a valid error code under "ErrorIdent".

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

In addition to the errors mentioned below, the function block displays errors of the internally used function block "IL_SIIISvcRead" of the "RIL_SercosIII" library (F_RELATED_ TABLE) as well as errors of the Sercos service channel (error table SERCOS_TABLE).

Error of "MT_AxPower", MTX_TABLE

Error code (hex)	Description
0x00000000	No error code available
0x00060102	Drive not ready for controller enable, signal status (iAx_DrvReady), the drive has to be in status AB
0x00040101	Setting or resetting the AxisInterface signal (qAx_DrvOn) failed as the signal was changed outside the FB
0x00020101	Setting/resetting the controller enable was requested but no feedback was sent to the drive (axis interface signal: iAx_DrvAct). Maybe, the wrong AxisInterface was probably connected to the function block

Tab. 14-87: Error of "MT_AxPower" from "ErrorIdent.Additional1" if "Error"==TRUE

Implementation example

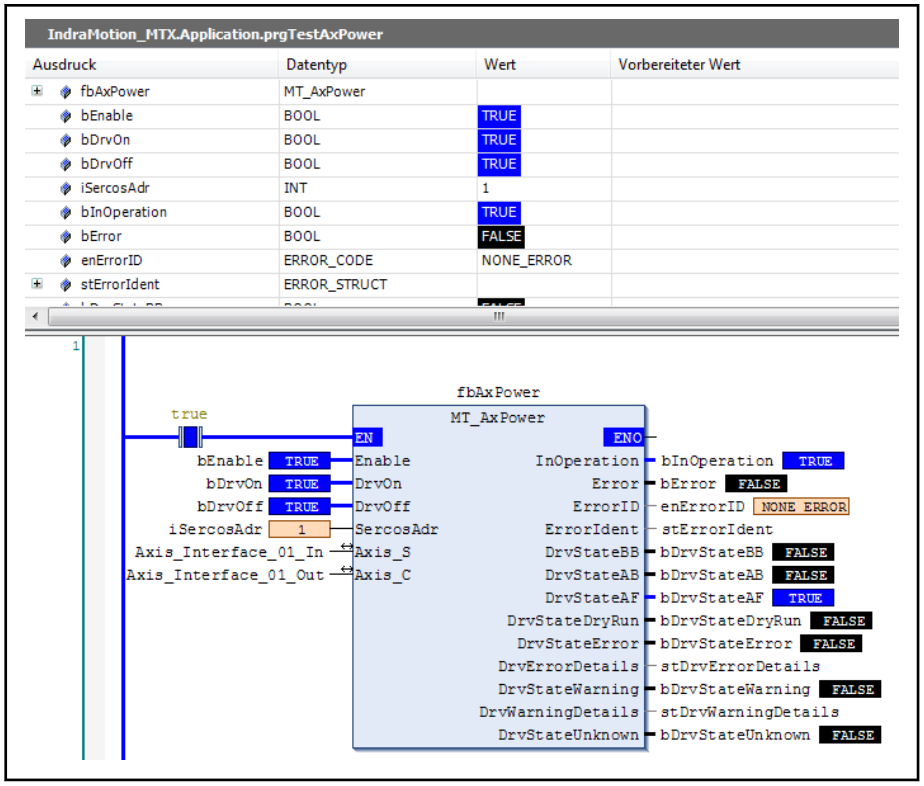


Fig. 14-109: Implementation MT_AxPower

14.5 Spindles

14.5.1 Overview

Overview on spindle functions		
Category	**Function/function block**	**Brief description**
General information	MT_SpSetOverride, page 312	Change feed for a spindle
Jog velocity	MT_SpJogSpeed, page 317	Determines and changes the active jog velocity of a spindle (combines (MT_SpGetJogSpeed and MT_SpSetJogSpeed)
	MT_SpGetJogSpeed, page 313	Determines the current jog velocity of a spindle
	MT_SpSetJogSpeed, page 314	Changes the active jog velocity of a spindle

Power & diagnostics	MT_SpPower, page 327	Determines the drive status and sets/ resets the controller enable (combines MT_SpSetPowerOn, MT_SpSetPowerOff and MT_SpGetPowerStateExt)
	MT_SpSetPowerOn, page 323	Sets controller enable for a spindle
	MT_SpSetPowerOff, page 325	Resets the controller enable of a spindle
	MT_SpGetPowerState, page 320	Determines the drive status of a spindle
	MT_SpGetPowerStateExt, page 322	Determines the drive status of a spindle including the detailed information about errors and warning

Tab. 14-88: Overview on spindle functions

14.5.2 MT_SpSetOverride, Changing the feed for a spindle

Brief description

Function to set the feed value for a spindle.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-89: Library names according to the target systems

Interface description

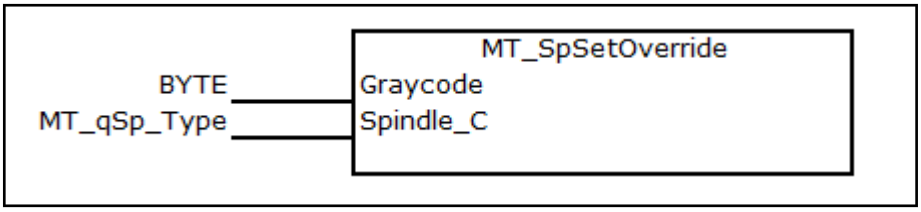


Fig. 14-110: Function block MT_SpSetOverride

	Name	Type	Description
VAR_INPUT	Graycode	BYTE	Override value as gray code
VAR_IN_OUT	Spindle_C	MT_qSp_Type	Axis interface: Spindle control signals

Tab. 14-90: Inputs/outputs function block MT_SpSetOverride

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

The axis feed value has to transmitted for each spindle from the PLC to the NC via its spindle interface (MT_qSp_Type) using the "qCh_Override" signal. Using this function, the value at the "Graycode" input is changed and transmitted to the connected spindle interface.

Troubleshooting

No error handling required for this function.

Implementation example

Example program for changing the override for the first axis.

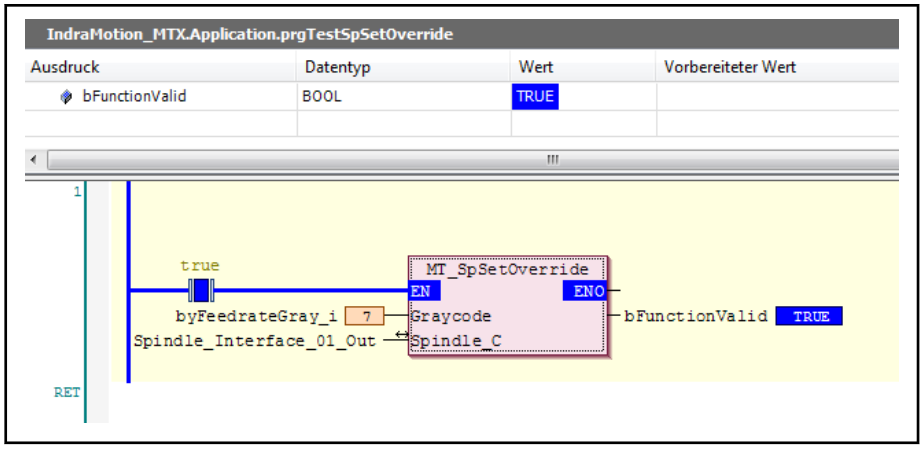


Fig. 14-111: Implementation MT_SpSetOverride

14.5.3 MT_SpGetJogSpeed, Determining the active jog velocity

Brief description

Function block to determine the active jog velocity of a spindle.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX_CMLxx	MT_BaseAdditional.lib

Tab. 14-91: Library names according to the target systems

Interface description

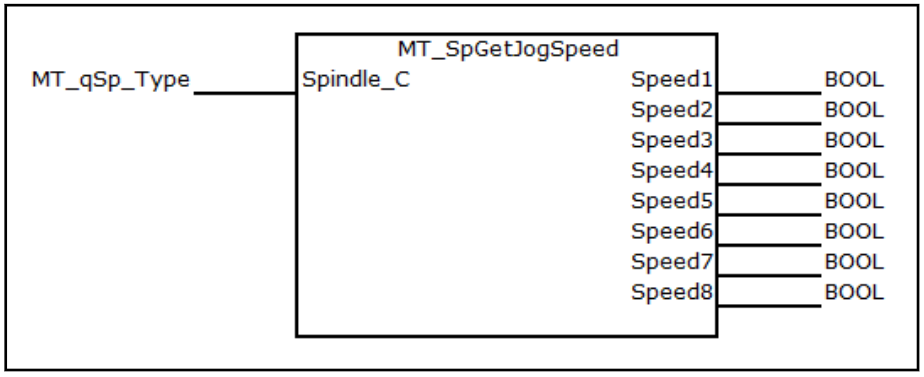


Fig. 14-112: Function block MT_SpGetJogSpeed

	Name	Type	Description
VAR_INPUT	Spindle_C	MT_qSP_TYPE	Spindle interface: Spindle control signals
VAR_OUTPUT	Speed1	BOOL	Spindle jog velocity 1
	Speed2	BOOL	Spindle jog velocity 2
	Speed3	BOOL	Spindle jog velocity 3
	Speed4	BOOL	Spindle jog velocity 4
	Speed5	BOOL	Spindle jog velocity 5
	Speed6	BOOL	Spindle jog velocity 6
	Speed7	BOOL	Spindle jog velocity 7
	Speed8	BOOL	Spindle jog velocity 8

Tab. 14-92: Inputs/outputs function block MT_SpGetJogSpeed

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

Using the MTX, the spindle can be jogged in 8 different velocities (machine parameter: "SP\SPFUNC\Sp[x]\Jog\Speed" - "MaCoDa no.: 104000015"). The jog velocity for the spindle is specified exclusively by the PLC to the NC. Thus, this function determines the specified jog velocity of the spindle interface last specified by the PLC (MT_qSp_Type) of the "qSp_ManSpeed_00..03" signals.

Troubleshooting

No error handling required for this function.

Implementation example

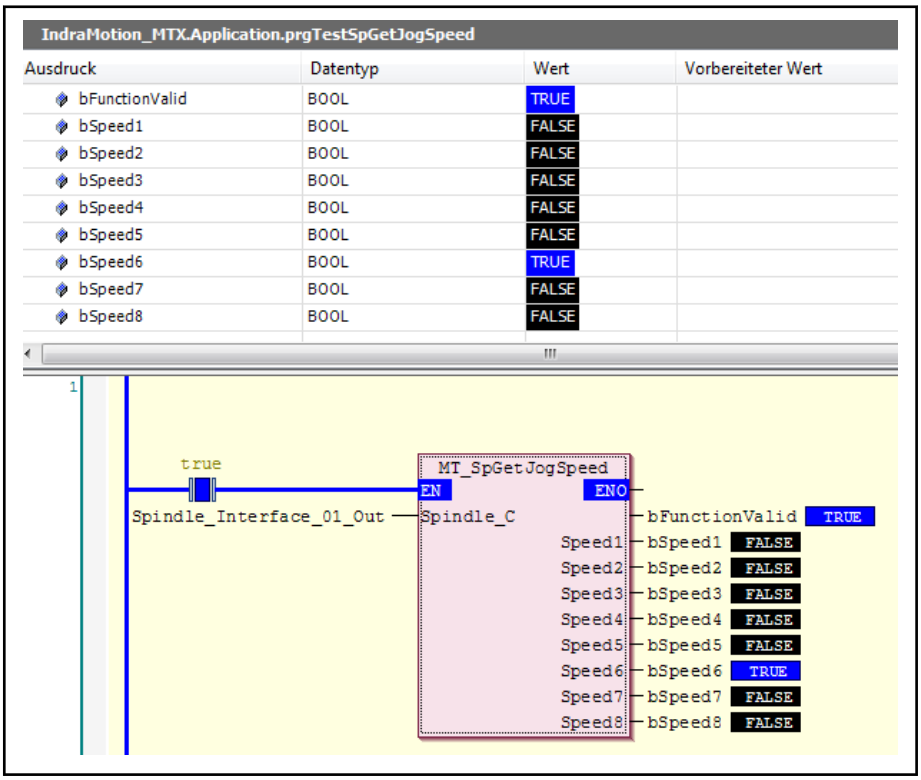


Fig. 14-113: Implementation MT_SpGetJogSpeed

14.5.4 MT_SpSetJogSpeed, Changing the jog velocity of a spindle

Brief description

Function block to change the jog velocity of a spindle.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-93: Library names according to the target systems

Interface description

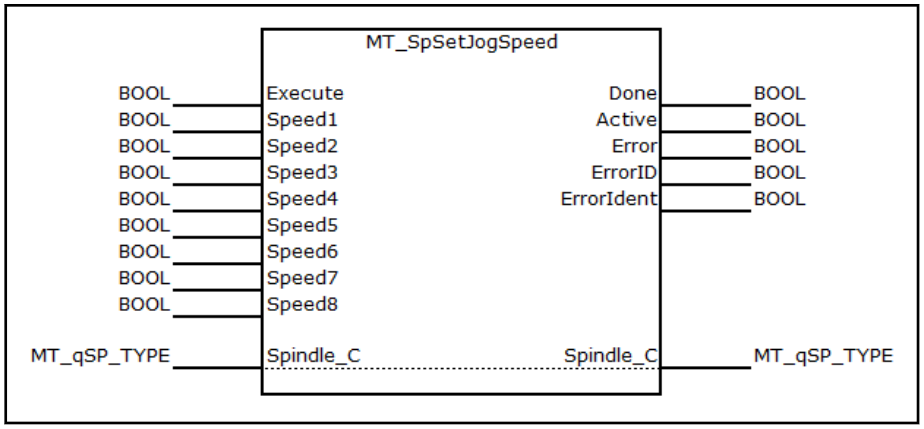


Fig. 14-114: Function block MT_SpSetJogSpeed

	Name	Type	Description
VAR_INPUT	Execute	BOOL	Execute setting the preselected jog velocity
	Speed1	BOOL	Spindle jog velocity 1
	Speed2	BOOL	Spindle jog velocity 2
	Speed3	BOOL	Spindle jog velocity 3
	Speed4	BOOL	Spindle jog velocity 4
	Speed5	BOOL	Spindle jog velocity 5
	Speed6	BOOL	Spindle jog velocity 6
	Speed7	BOOL	Spindle jog velocity 7
	Speed8	BOOL	Spindle jog velocity 8
VAR_OUTPUT	Done	BOOL	Processing successfully completed
	Active	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "bError" = TRUE)
VAR_IN_OUT	Spindle_C	MT_qSp_Type	Spindle interface: Spindle control signals

Tab. 14-94: Inputs/outputs function block MT_SpSetJogSpeed

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

Using the MTX, the spindle can be jogged in 8 different velocities (machine parameter: "SP\SPFUNC\Sp[x]\Jog\Speed" - "MaCoDa no.: 104000015"). The 8 jog velocities are exclusively preselected as coded by the PLC via the PLC AxisInterface (MT_qAx_Type) by means of the signals " qSp_ManSpeed_00..03". These velocities can be enabled for one spindle using the function block "MT_SpSetJogSpeed".



This function block works edge-controlled. That means that the function block executed a jog velocity change exactly once after activation.

A signal change from zero to one (0 -> 1) at "Execute" enables the function block and backs up the state of the inputs at this point in time. Only the input "Spindle_C" is cyclically applied as long as the function is active ("Ac-

tive"==TRUE). When changing the signal, the desired jog velocity "Speed1...8" has to be preselected. If no jog velocity or several jog velocities are preselected when setting "Execute", the function block goes into error state ("Error" becomes TRUE). As soon as the "Done" output becomes TRUE, setting the desired "Speed" is successfully completed. If the "Speed" could not be changed, this is displayed by the "Error" output. As long as "Error" is TRUE, there is a valid error code under "ErrorIdent". Before switching the "Speed" again or deleting the error, "Execute" has to be set to FALSE. Resetting "Execute" deletes "Error" and "Done". As long as the outputs "Done", "Active" or "Error" are TRUE, signal changes at the "Speed" inputs have no effect. The outputs "Done", "Active" and "Error" exclude each other. That means that only one is TRUE at a time.

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_SpSetJogSpeed"

Error code (hex)	Description
0x00000000	No error code available
0x00010404	Invalid input assignment. No "Speed" or more than one have to be preselected
0x00040301	Changing the interface signals "qSp_ManSpeed_00..03" failed. Possibly overwritten outside the function block.

Tab. 14-95: Error of "MT_SpSetJogSpeed" of "ErrorIdent.Additional1", valid if "Error"==TRUE

Implementation example

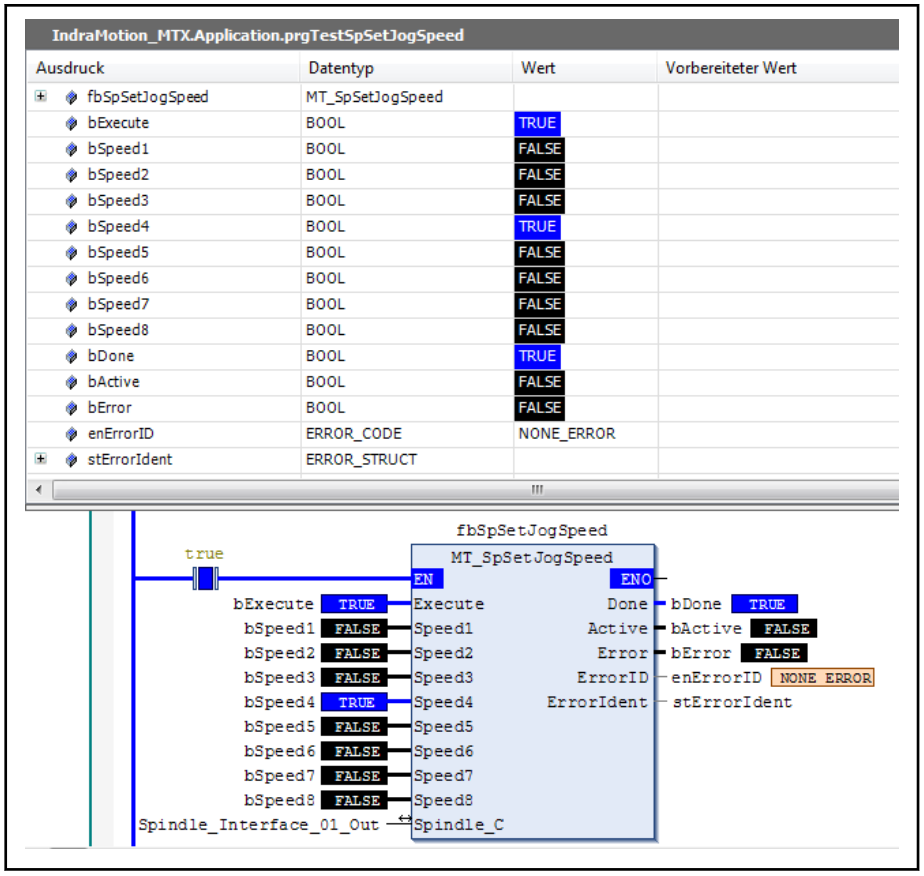


Fig. 14-115: Implementation MT_SpSetJogSpeed

14.5.5 MT_SpJogSpeed, Changing and determining the active jog velocity

Brief description

Function block used to determine the active jog velocity, as well as to change it.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-96: Library names according to the target systems

Interface description

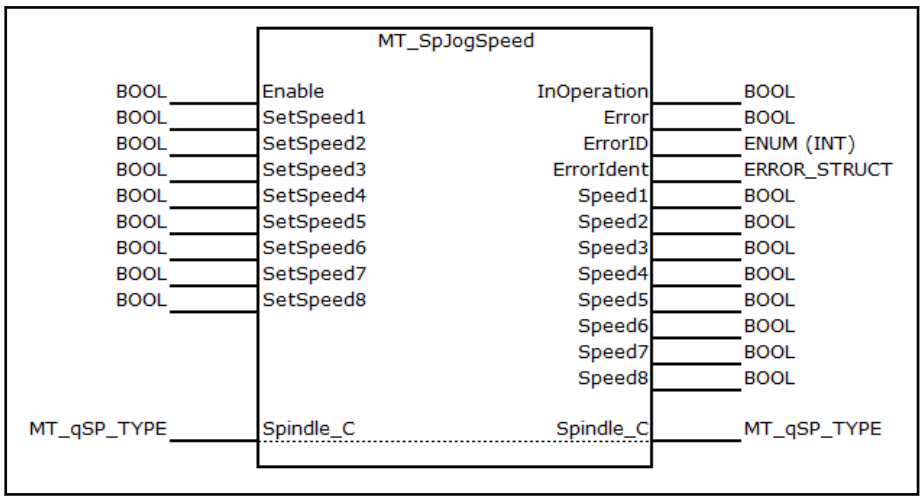


Fig. 14-116: Function block MT_SpJogSpeed

	Name	Type	Description
VAR_INPUT	Enable	BOOL	Enable function block
	SetSpeed1	BOOL	Request spindle jog velocity 1
	SetSpeed2	BOOL	Request spindle jog velocity 2
	SetSpeed3	BOOL	Request spindle jog velocity 3
	SetSpeed4	BOOL	Request spindle jog velocity 4
	SetSpeed5	BOOL	Request spindle jog velocity 5
	SetSpeed6	BOOL	Request spindle jog velocity 6
	SetSpeed7	BOOL	Request spindle jog velocity 7
	SetSpeed8	BOOL	Request spindle jog velocity 8
	InitJogSpeed	BOOL	Preselects JogSpeed when enabling the function block
VAR_OUTPUT	InOperation	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "Error" = TRUE)
	Speed1	BOOL	Spindle jog velocity 1 is active
	Speed2	BOOL	Spindle jog velocity 2 is active
	Speed3	BOOL	Spindle jog velocity 3 is active
	Speed4	BOOL	Spindle jog velocity 4 is active
	Speed5	BOOL	Spindle jog velocity 5 is active
	Speed6	BOOL	Spindle jog velocity 6 is active
	Speed7	BOOL	Spindle jog velocity 7 is active
	Speed8	BOOL	Spindle jog velocity 8 is active
VAR_IN_OUT	Spindle_C	MT_qSp_Type	Spindle interface: Spindle control signals

Tab. 14-97: Inputs/outputs function block MT_SpJogSpeed

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

Using the MTX, the spindle can be jogged in 8 different velocities (machine parameter: "SP\SPFUNC\Sp[x]\Jog\Speed" - "MaCoDa no.: 104000015"). The 8 jog velocities are exclusively preselected as coded by the PLC via the PLC AxisInterface (MT_qAx_Type) by means of the signals " qSp_Man-Speed_00..03". These velocities can be enabled for one spindle using the function block MT_SpSetJogSpeed. These jog velocities can be activated for a spindle and the current status can be read out using this function block.

As soon as "Enable" is TRUE, the current jog velocity is determined upon each function block call from the connected spindle interface. Upon a signal change (0-->1) at one of the "SetSpeed... inputs", the jog velocity is changed. If the jog velocity could not be changed, this is displayed at the "Error"output. As long as "Error" is TRUE, there is a valid error code under "ErrorIdent". "Enable" has to be FALSE to delete errors.

The "InitJogSpeed" input facilitates setting a JogSpeed once upon activating the FB without one of the inputs of the "SetJog... inputs" being TRUE.

The following specifications are valid:

InitJogSpeed	JogSpeed
0	No initialization
1	JogSpeed1
2	JogSpeed2
3	JogSpeed3
4	JogSpeed4
5	JogSpeed5
6	JogSpeed6
7	JogSpeed7
8	JogSpeed8

Tab. 14-98: Initialization JogSpeed

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_SpJogSpeed"

Error code (hex)	Description
0x00000000	No error code available
0x00010404	Invalid input assignment. No "Speed" or more than one have to be preselected
0x00040301	Changing the interface signals "qSp_ManSpeed_00..03" failed. The interfaces might have been overwritten outside of the function block

Tab. 14-99: Error of "MT_SpSetJogSpeed" of "ErrorIdent.Additional1", valid if "Error"==TRUE

Implementation example

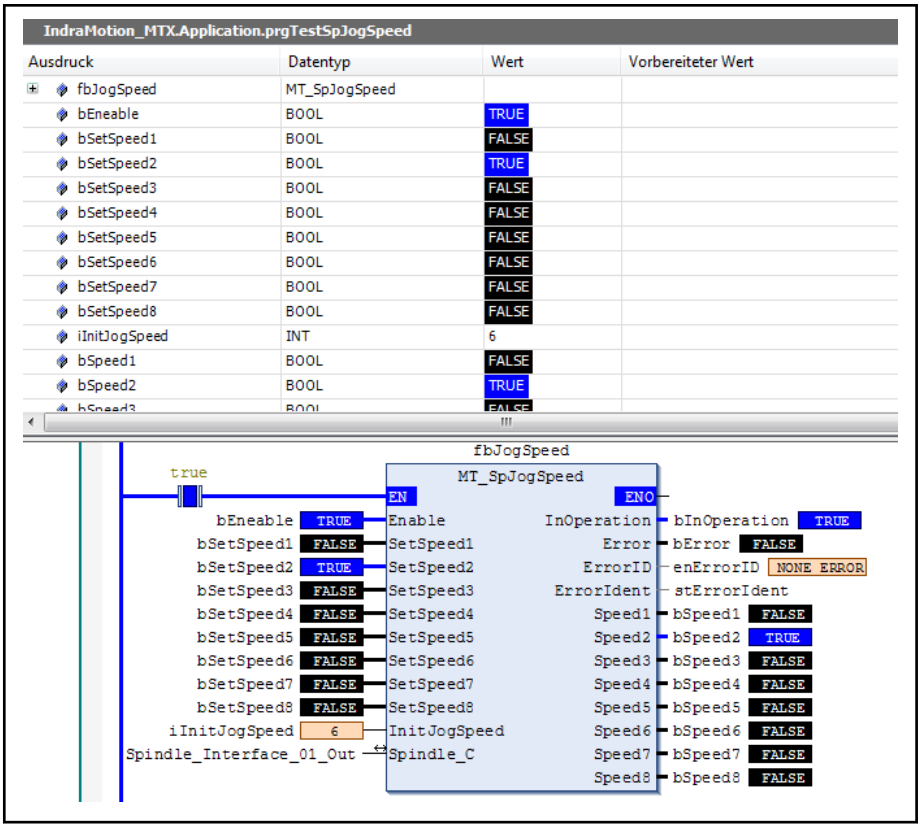


Fig. 14-117: Implementation MT_SpJogSpeed

14.5.6 MT_SpGetPowerState, Determining the drive status

Brief description

Function to determine the active drive status of a spindle.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-100: Library names according to the target systems

Interface description

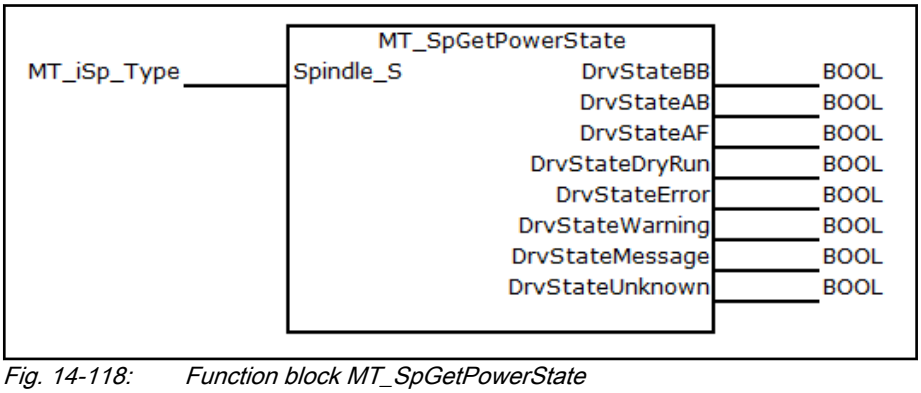


Fig. 14-118: Function block MT_SpGetPowerState

	Name	Type	Description
VAR_INPUT	Spindle_S	MT_iSP_TYPE	Axis interface: Spindle status signals
VAR_OUTPUT	DrvStateBB	BOOL	Drive ready
	DrvStateAB	BOOL	Drive ready for controller enable
	DrvStateAF	BOOL	Drive in control
	DrvStateDryRun	BOOL	Drive in virtual mode
	DrvStateError	BOOL	Drive reports an error
	DrvStateWarning	BOOL	Drive reports a warning
	DrvStateMessage	BOOL	Drive reports a note

Tab. 14-101: Inputs/outputs function block MT_SpGetPowerState

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

This function determines the performance from the AxisInterface (MT_iSp_Type) and displays potential errors, warnings or information at the drive.

The following Interface signals are interpreted:

- iSp_DrvPower
- iSp_DrvReady
- iSp_DrvAct
- iSp_DryRun
- iSp_DrvErrClass1
- iSp_DrvChangeClass2
- iSp_DrvChangeClass3

Troubleshooting

No error handling required for this function.

Implementation example

The status of a spindle is determined in the example program.

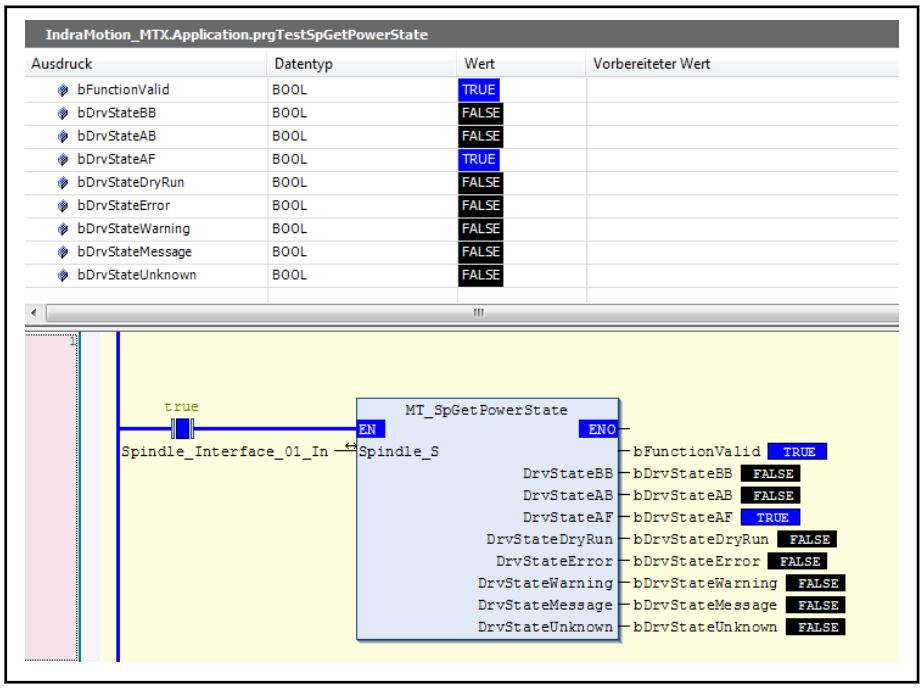


Fig. 14-119: Implementation MT_SpGetPowerState

14.5.7 MT_SpGetPowerStateExt, drive status incl. Determining errors and warnings

Brief description

Function to determine the active drive status of a spindle with additional information about errors and warnings.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-102: Library names according to the target systems

Interface description

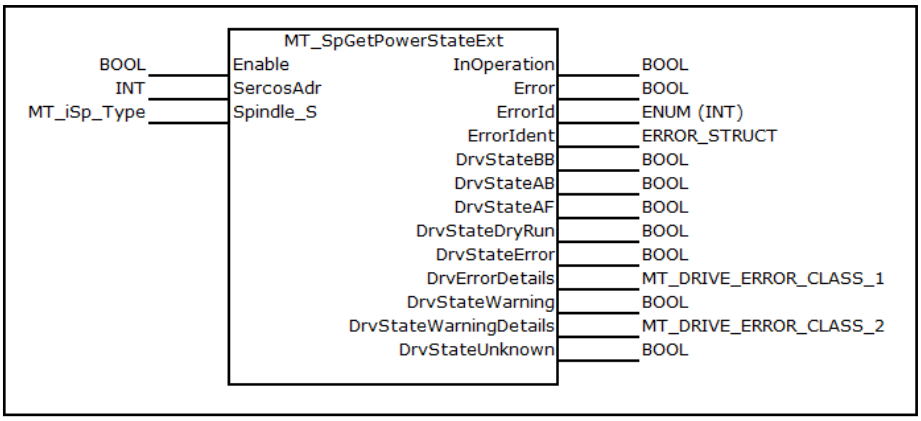


Fig. 14-120: MT_SpGetPowerStateExt function block

	Name	Type	Description
VAR_INPUT	Spindle_S	MT_iSP_TYPE	Axis interface: Spindle control signals
VAR_OUTPUT	DrvStateBB	BOOL	Drive ready
	DrvStateAB	BOOL	Drive ready for controller enable
	DrvStateAF	BOOL	Drive in control
	DrvStateDryRun	BOOL	Drive in virtual mode
	DrvStateError	BOOL	Drive reports an error
	DrvStateErrorDetails	MT_DRIVE_ERROR_CLASS_1	Detail information about the error (S-0-0011)
	DrvStateWarning	BOOL	Drive reports a warning
	DrvStateWarningDetails	MT_DRIVE_ERROR_CLASS_2	Detailed information on the warning (S-0-0012)

Tab. 14-103: Inputs/outputs function block MT_SpGetPowerStateExt

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

This function block determines the performance from the AxisInterface (MT_iSp_Type) and displays potential errors or warnings at the drive. Additionally, detailed information is output from the drive parameters S-0-0011 or S-0-0012 in case of pending errors or warnings.

The following Interface signals are interpreted:

- iSp_DrvPower
- iSp_DrvReady
- iSp_DrvAct
- iSp_DryRun

- iSp_DrvErrClass1
- iSp_DrvChangeClass2
- iSp_DrvChangeClass3

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table.

The function block displays errors of the internally used function block "IL_SIIISvcRead" of the "RIL_SercosIII" library.

Implementation example

The status of a spindle is determined in the example program.

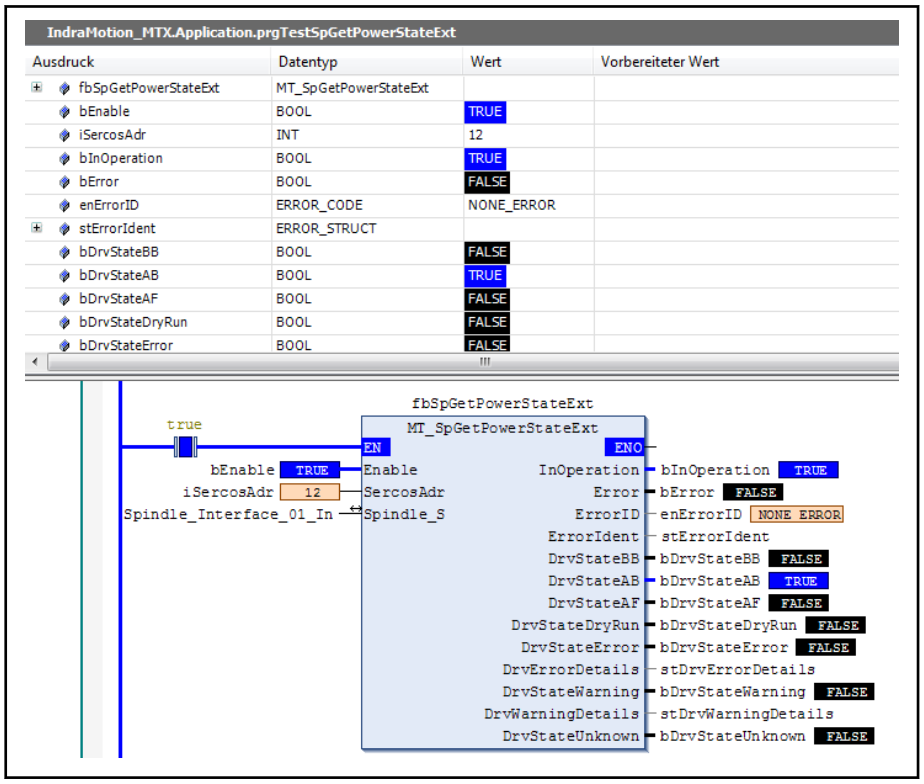


Fig. 14-121: Implementation MT_SpGetPowerStateExt

14.5.8 MT_SpSetPowerOff, Setting the controller enable for a spindle

Brief description

Function block to activate the controller enable for a spindle.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-104: Library names according to the target systems

Interface description

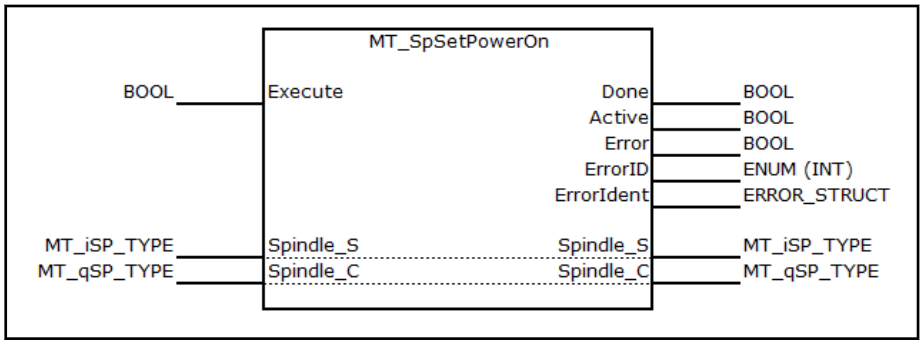


Fig. 14-122: Function block MT_SpSetPowerOn

	Name	Type	Description
VAR_INPUT	Execute	BOOL	Sets controller enable for a spindle
VAR_OUTPUT	Done	BOOL	Processing successfully completed
	Active	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "bError" = TRUE)
VAR_IN_OUT	Spindle_S	MT_iSp_Type	Axis interface: Spindle control signals
	Spindle_C	MT_qSp_Type	Axis interface: Spindle control signals

Tab. 14-105: Inputs/outputs function block MT_SpSetPowerOn

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

The controller enable (AF) for a spindle is set in the AxisInterface (MT_qSp_Type) via the axis control signal "qSp_DrvOn" in MTX. The function block checks if the spindle is ready for controller enable, sets the signal and monitors if feedback is send within 5 seconds that the spindle is in control via the axis status signals (MT_iSp_Type).



This function block works edge-controlled. That means that the function block executes the controller enable "qSp_DrvOn" exactly once after activation.

A signal change from zero to one (0 -> 1) at "Execute" enables the function block. Only the inputs "Spindle_S" and "Spindle_C" are cyclically applied as long as the function is active ("Active" = TRUE). As soon as the "Done" output becomes TRUE, setting the controller enable is successfully completed. If the controller enable could not be set for the drive, this is displayed by the "Error" output. In case of an error, the interface signal "qSp_DrvOn" is reset. As long as "Error" is TRUE, there is a valid error code under "ErrorIdent". Before setting the controller enable or deleting the error, "Execute" has to be set to FALSE. Resetting "Execute" deletes "Error" and "Done". The outputs "Done", "Active" and "Error" exclude each other. That means that only one is TRUE at a time.

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_SpSetPowerOn"

Error code (hex)	Description
0x00000000	No error code available
0x00060302	Drive not ready for controller enable, signal status (iASp_DrvReady), the drive has to be in status AB
0x00040301	Setting the AxisInterface signals (qSp_DrvOn) failed, the signal was reset outside the function block

Error code (hex)	Description
0x00020301	The controller enable was required but the drive does not give any feedback (AxisInterface signal: iSp_DrvAct). Possibly, the wrong AxisInterface was connected to the AxisInterface

Tab. 14-106: Error of "MT_SpSetPowerOn" of "ErrorIdent.Additional1", valid if "Error"==TRUE

Implementation example

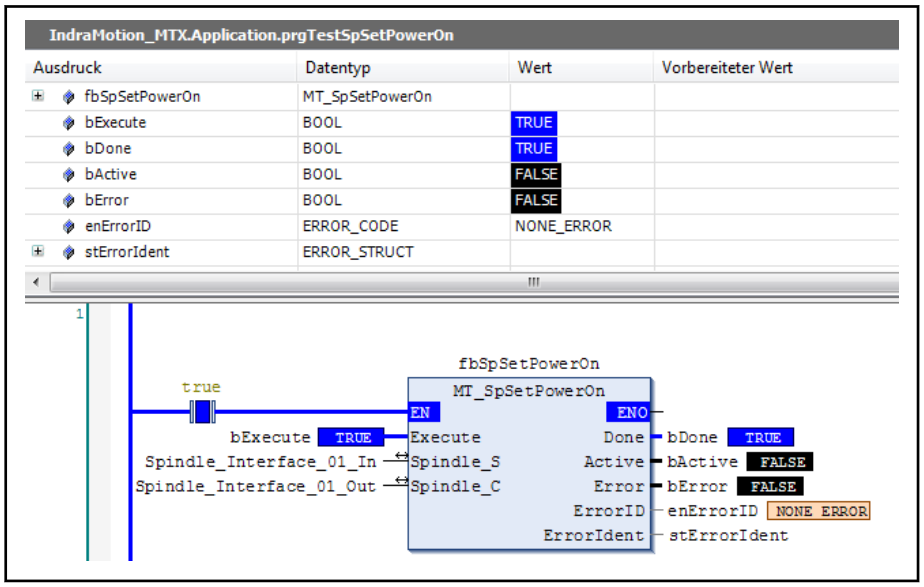


Fig. 14-123: Implementation MT_SpSetPowerOn

14.5.9 MT_SpSetPowerOff, Sets controller enable for a spindle

Brief description

Function block to deactivate the controller enable for a spindle.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-107: Library names according to the target systems

Interface description

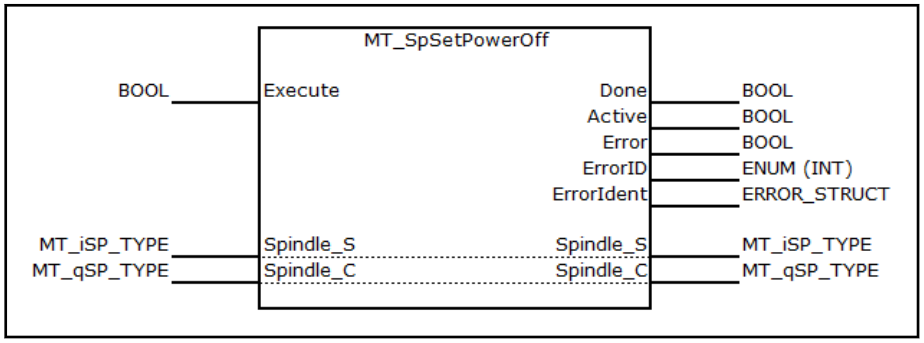


Fig. 14-124: Function block MT_SpSetPowerOff

	Name	Type	Description
VAR_INPUT	Execute	BOOL	Resets controller enable for a spindle
VAR_OUTPUT	Done	BOOL	Processing successfully completed
	Active	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "bError" = TRUE)
VAR_IN_OUT	Spindle_S	MT_iSp_Type	Axis interface: Spindle control signals
	Spindle_C	MT_qSp_Type	Axis interface: Spindle control signals

Tab. 14-108: Inputs/outputs function block MT_SpSetPowerOff

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

The controller enable (AF) for a spindle is reset in the AxisInterface (MT_qSp_Type) via the axis control signal "qSp_DrvOn" in MTX. The function block sets the signal to FALSE and monitors if a feedback is send within 5 seconds via the axis status signals (MT_iSp_Type) that the spindle is not in control anymore.



This function block works edge-controlled. That means that the function block executed resetting of the controller enable "qSp_DrvOn" exactly once after activation.

A signal change from zero to one (0 -> 1) at "Execute" enables the function block. Only the inputs "Spindle_S" and "Spindle_C" are cyclically applied as long as the function is active ("Active" = TRUE). As soon as the "Done" output becomes TRUE, resetting of the controller enable is successfully completed. If the controller enable of the drive could not be reset, this is displayed by the "Error" output. In case of an error, the interface signal "qSp_DrvOn" remains FALSE. As long as "Error" is TRUE, there is a valid error code under "ErrorIdent". Before resetting the controller enable again or deleting the error, "Execute" has to be set to FALSE. Resetting "Execute" deletes "Error" and "Done". The outputs "Done", "Active" and "Error" exclude each other. That means that only one output is TRUE at a time.

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_SetPowerOn"

Error code (hex)	Description
0x00000000	No error code available
0x00040301	Resetting the AxisInterface signal (qSp_DrvOn) failed as the signal was reset outside the FB

Error code (hex)	Description
0x00020301	Resetting the controller enable was requested but the drive does not give any feedback (AxisInterface signal: iSp_DrvAct). Possibly, the wrong AxisInterface was connected to the AxisInterface

Tab. 14-109: Error of "MT_SpSetPowerOff" of "ErrorIdent.Additional1", valid if "Error"==TRUE

Implementation example

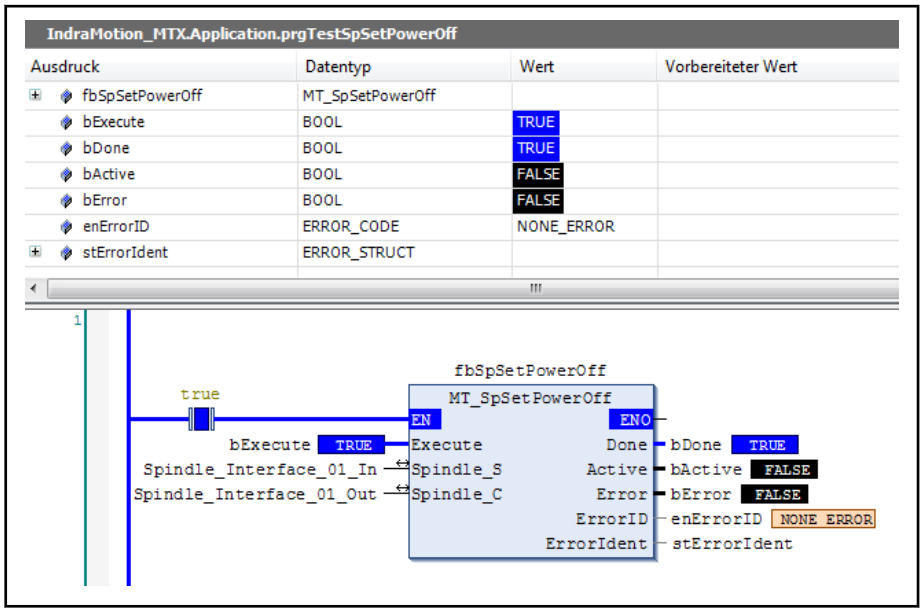


Fig. 14-125: Implementation MT_SpSetPowerOff

14.5.10 MT_SpPower, Controller enable and spindle status

Brief description

Function block to activate/deactivate the controller enable as well as for determining the current spindle status.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-110: Library names according to the target systems

Interface description

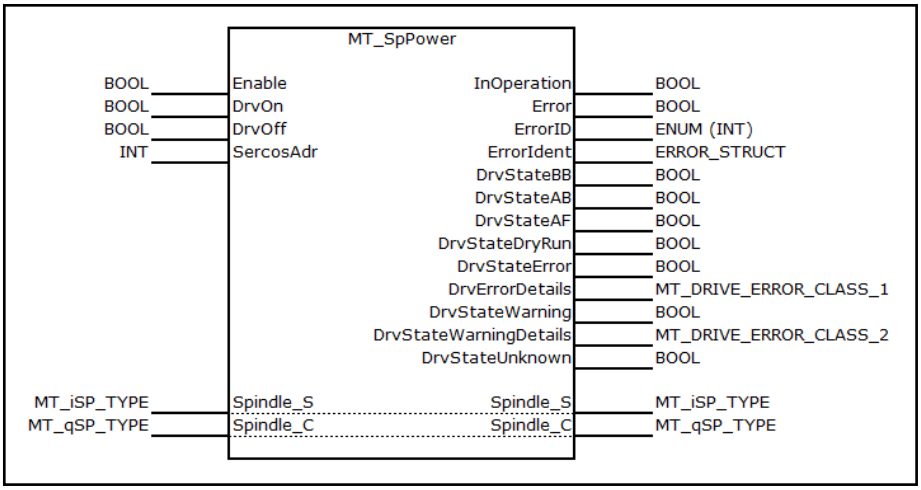


Fig. 14-126: Function block MT_SpPower

	Name	Type	Description
VAR_INPUT	Enable	BOOL	Enable function block
	DrvOn	BOOL	Sets controller enable (DrvOff also has to be TRUE)
	DrvOff	BOOL	Falling edge resets controller enable
	SercosAdr	INT	Sercos address of the drive
VAR_OUTPUT	InOperation	BOOL	Processing successfully completed
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "bError" = TRUE)
	DrvStateBB	BOOL	Drive ready
	DrvStateAB	BOOL	Drive ready for controller enable
	DrvStateAF	BOOL	Drive in control
	DrvStateDryRun	BOOL	Drive in virtual mode
	DrvStateError	BOOL	Drive reports an error
	DrvStateErrorDetails	MT_DRIVE_ERROR_CLASS_1	Detail information about the error (S-0-0011)
	DrvStateWarning	BOOL	Drive reports a warning
	DrvStateWarningDetails	MT_DRIVE_ERROR_CLASS_2	Detailed information on the warning (S-0-0012)
VAR_IN_OUT	Spindle_S	MT_iSp_Type	Axis interface: Spindle control signals
	Spindle_C	MT_qSp_Type	Axis interface: Spindle control signals

Tab. 14-111: Inputs/outputs function block MT_SpPower

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

This function block combines the functionalities of "MT_SpSetPowerOn", "MT_SpSetPowerOff" and "MT_SpGetPowerStateExt". The function block can control the controller enable of a spindle, determine its status cyclically and provide detailed information on warnings and errors.

By setting "Enable", all inputs are read cyclically, the drive status is continuously determined from the spindle interface and is displayed via the "DrvState..." outputs. As long as "Enable" and "DrvOff" are TRUE, setting of the controller enable for the spindle can be requested with a rising edge at "DrvOn". The controller enable is reset via a falling edge at DrvOff. As long as the spindle is in state "DrvStateDryRun", "DrvOn" and "DrvOff" are inactive.

The controller enable (AF) is set and reset in the NC PLC AxisInterface (MT_qSp_Type) via the axis control signal "qSp_DrvOn". Prior to setting the controller enable, it is checked if the axis is ready for controller enable (AB). Only then, the signal is set. When setting or resetting the controller enable, it is monitored if a feedback is send within 5 seconds via the axis status signals (MT_iSp_Type).

The function block is executed successfully as long as the "InOperation" output is TRUE. In case of an error, the "Error" output indicates the error and "InOperation" is set to FALSE. In case of an error, the interface signal "qSp_DrvOn" is reset. As long as "Error" is TRUE, there is a valid error code under "ErrorIdent".

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

In addition to the errors mentioned below, the function block displays errors of the internally used function block "IL_SIIISvcRead" of the "RIL_SercosIII" library (F_RELATED_ TABLE) as well as errors of the Sercos service channel (error table SERCOS_TABLE).

Error of "MT_SpPower", MTX_TABLE

Error code (hex)	Description
0x00000000	No error code available
0x00060302	Drive not ready for controller enable, signal status (iASp_DrvReady), the drive has to be in status AB
0x00040301	Setting or resetting the AxisInterface signal (qSp_DrvOn) failed as the signal was changed outside the FB
0x00020301	Setting/resetting the controller enable was requested but the drive does not give any feedback (AxisInterface signal: iSp_DrvAct). Possibly, the wrong AxisInterface was connected to the AxisInterface

Tab. 14-112: Error of "MT_SpPower" of "ErrorIdent.Additional1", valid if "Error"==TRUE

Implementation example

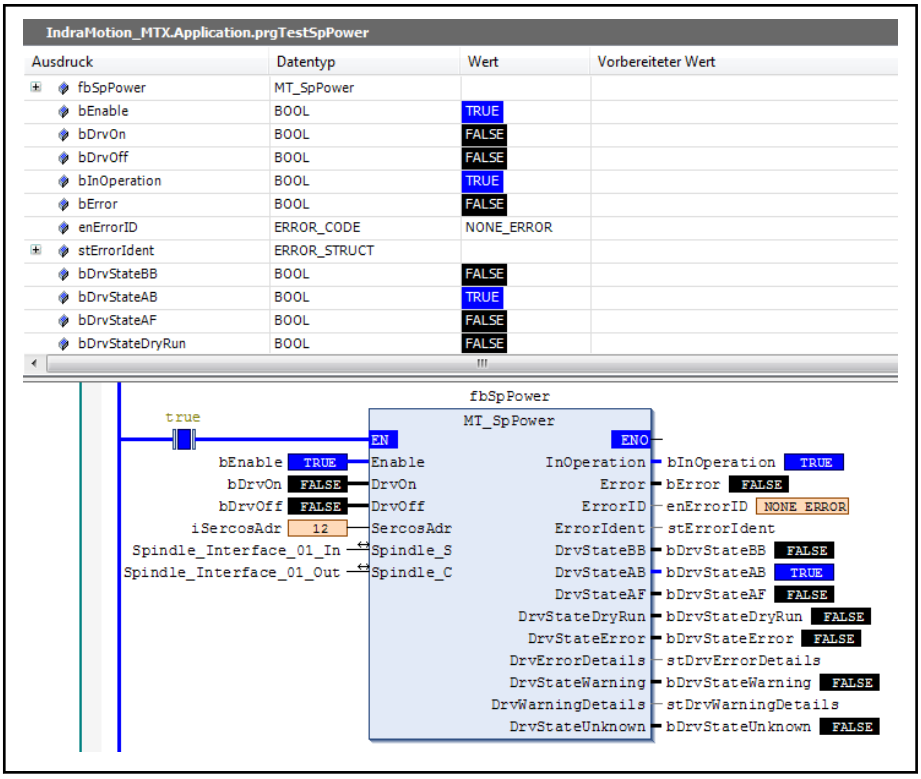


Fig. 14-127: Implementation MT_SpPower

14.6 Tool management

14.6.1 Overview

Overview on tool management functions		
Category	**Function/function block**	**Brief description**
Tool life	MT_TmToolLife, page 330	Determines the tool service life

Tab. 14-113: Overview on tool management functions

14.6.2 MT_TmToolLife, tool, tool life counting

Brief description

The function block calculates the tool life of the active edge of the active tool and displays the state of wear.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-114: Library names according to the target systems

Interface description

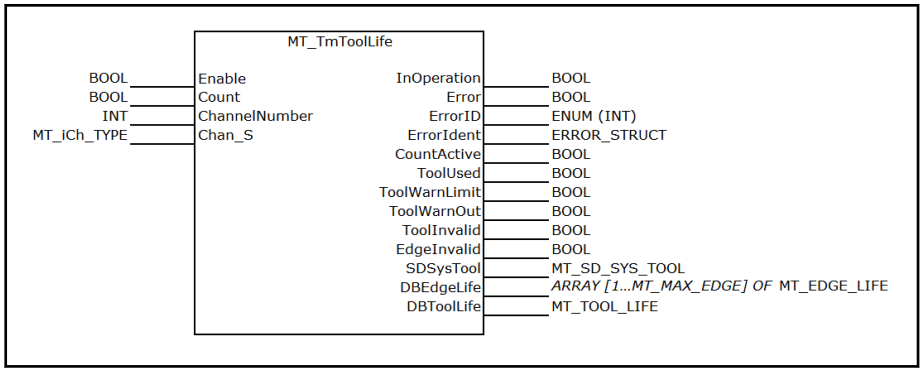


Fig. 14-128: MT_TmToolLife function block

	Name	Type	Init	Description
VAR_INPUT	Enable	BOOL	FALSE	Enable function block
	Count	BOOL	FALSE	Counting/acknowledging the tool life by the function block if completed
	ChannelNumber	INT	1	Channel to be used by tool life counting
VAR_IN_OUT	Chan_S	MT_iCH_TYPE		Channel interface inputs for the channel on which the tool life counting is to be applied to
VAR_OUT-PUT	InOperation	BOOL	FALSE	Processing active
	Error	BOOL	FALSE	Processing incorrect
	ErrorID	ENUM (INT)	NONE_ERROR	Rough error information
	ErrorIdent	ERROR_STRUCT	--	Detailed error information (only valid if "bError" = TRUE)
	CountActive	BOOL	FALSE	
	ToolUsed	BOOL	FALSE	The active tool is already used
	ToolWarnLimit	BOOL	FALSE	The active tool reached the warning limit
	ToolWornOut	BOOL	FALSE	The active tool is worn
	ToolInvalid	BOOL	FALSE	Active tool invalid, "SD.SysTool.ActToolValid=FALSE"
	EdgeInvalid	BOOL	FALSE	Active edge invalid, "SD.SysTool.ActEd<=0 OR >MT_MAX_ED"
	SDSysTool	MT_SD_SYS_TOOL	chapter 14.8.3 "MT_SD_SYS_TOOL, Structure of the SD.SysTool system date" on page 344	Data of the active tool from the system date SD.SysTool

	Name	Type	Init	Description
	DBEdgeLife	ARRAY [1..MT_MAX_EDGE] OF MT_EDGE_LIFE		Check the wear data of all edges of the active tool
	DBToolLife	MT_TOOL_LIFE		Wear of the active tools as well as information on whether it is counted acc. to time or in cycles

Tab. 14-115: Inputs/outputs of the MT_TmToolLife function block

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

Prerequisites:

☞ The basic condition to use the function block is that the system date "SD.SysTool" is written correctly in the tool change program. The standard tool change "TCH" provided by Bosch Rexroth ensures this.

For detailed information on the system date "SD.SysTool", refer to the manual "MTX 14VRS Machine Parameters".

If a valid tool data block "SD.SysTool.ActTool" is saved in by the tool change program, it has to be confirmed by setting "SD.SysTool.ActToolValid". Otherwise, no tool life counting takes place, and the "ToolInvalid" output is TRUE.

Tools can have a different number of edges. The tool with the highest number of edges determines the maximum number of edges. This value is defined in the tool user data structure "dbt1ud.xsd" for the NC and is defined by the global constant "MT_MAX_EDGE" for the PLC. The maximum number of tools for NC and PLC should match! If the edge number entered in "SD.SysTool.ActTool" is zero or greater than the edge number specified in "MT_MAX_EDGE", the output "EdgeInvalid" is set to TRUE and no tool life counting takes place!

☞ The alterable PLC constant "MT_MAX_EDGE" is part of the "MT_Base" library and is contained in the "Constants/MT_ToolManagement_Const" folder.

Operating principle:

This function block counts tool life for the active tool edge.

The active edge is determined from the channel interface signal MT_iCh_Type.iCh_ActFunc09..13 and the active tool is determined from the system date "SD.SysTool".

The function block retrieves from the database (/DBT1/Rec/UD/Tl/ Life) whether the tool life is calculated based on cycles or time. This information is displayed at the "DBToolLife" output. If cycles are used as calculation basis, the tool life is reduced by one with each rising edge at the "Count" input. If the time is used as computation basis, the tool life is reduced by 0.0001 after each 10 ms as long as the "Count" input remains TRUE.

The newly computed tool life for the edges is updated every second in the database and output at the function block output "DBEdgeLife".

There is a tool life for each edge (/DBT1/Rec/UD/Ed[x]/Life/RemLife) as well as a tool life for the entire tool (DBT1/Rec/UD/Tl/ Life/RemLife). The smallest

tool life of an edge corresponds to the tool life of the tool. The function block verifies this and refreshes the tool life accordingly. If there are changes, the result is sent every second to the database and shown at function block output "DBToolLife".

According to the established tool life and the limit values (/DBT1/Rec/UD/Ed[x]/Life/WarnLife & ... MaxLife) contained in the database, the function block checks if the tool has already been used, reached the warning limit or if the tool is worn already. The following applies: If the current tool life is smaller than the maximum tool life (MaxLife), the output "ToolUsed" becomes TRUE. If the current tool life is smaller than the specified warning limit (WarnLife), the "ToolWarnLimit" output is set to TRUE. If the current tool life is zero, the tool is worn and the "ToolWornOut" output is TRUE.

Additionally, the calculated states "Tool used", "Warning limit reached" and "Tool worn" is written to the database every second (/DBT1/Rec/Hd/BQ2).

☞ If the maximum tool life was specified with zero (/DBT1/Rec/UD/Ed[x]/Life/MaxLife=0) in the database, the tool life for this edge is not counted! Thus, it is possible to disable the tool life calculation for individual edges or tools in the HMI without having to adjust the remaining tool life.

All data of the active tool from the system date "SD.SysTool" is additionally output at the function block output "SDSysTool".

Troubleshooting

This function block generates a detailed diagnostics in case of error. A detailed description of the output error code (Additional1, Additional2) can be found in the respective table. This function block can output error codes for the following tables:

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_TmToolLifeControl“

Error code (Add1)	Error code (Add2)	Description
0x00000000	0x00000000	No error code available
16#0009041B	Status messages MT_DbData	Error of the function block MT_DbData, Code in Additional 2 is a status message of the MT_DbData function block, see description MT_DbData
16#00090422	Status message MT_SD_Rd	Error of the function block MT_SD_Rd, Code in Additional 2 is a status message of the MT_SD_Rdfunction block, see description MT_SD_Rd
16#00090423	Status message MT_SD_Wr	Error of the function block MT_SD_Wr, Code in Additional 2 is a status message of the MT_SD_Wr block, see description MT_SD_Wr

Tab. 14-116: Error of “MT_TmToolLifeControl”, valid if “Error”==TRUE

Implementation example

The following example program shows the tool life for edge 2 of the tool on place 1/sector 1 in channel 1. Assign the SD.SysTool system structure with the corresponding values (see functionality).

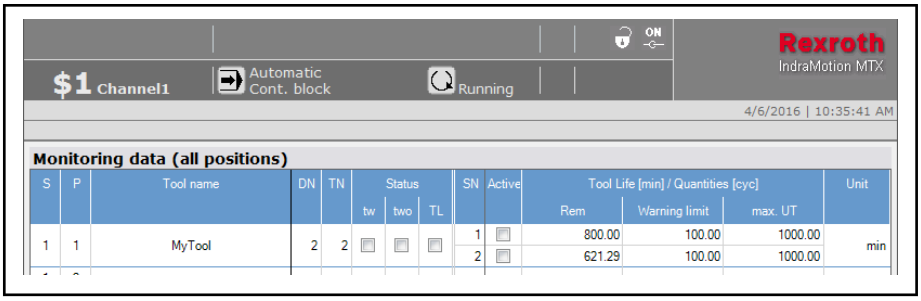


Fig. 14-129: Tool list in IndraWorks Operation OP5

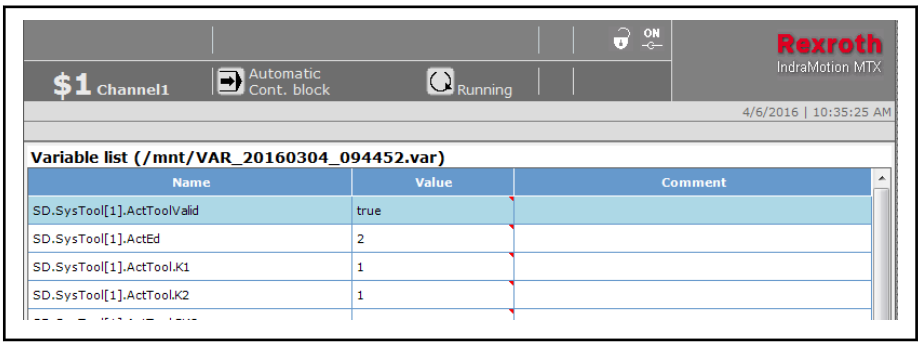


Fig. 14-130: Variable list with SD.SysTool in IndraWorks Operation

PLC program

The example shows the enabling of the tool life counting using the M-function M296 and its disabling using M297.

In network 3, the conditions triggering a counting process is defined.

●

```
PROGRAM pr70000ToolManagment
(*example for tool count
M0296  == enable FB
M0297  == disable FB
*)
VAR
//function block for ToolLife control
//-----------------------------------------------------
    fbToolCount         : MT_TmToolLife;                        //functionbloc
    //input
    bEnableFB           : BOOL:=FALSE;                          //enable FB
    bCount              : BOOL:=FALSE;                          //start count
    iChanNo             : UINT:=1;                              //channel number
    //output
    bError              : BOOL:=FALSE;                          //functionbloc error occur, for detailed infomation view ErrorIdent
    ErrorID             : ERROR_CODE;                           //basic error information
    ErrorIdent          : ERROR_STRUCT;                         //detailed error information
    CountActive         : BOOL := FALSE;                        //Life count active
    ToolUsed            : BOOL := FALSE;                        //Active tool has already been used
    ToolWarnLimit       : BOOL := FALSE;                        //Active tool has reached the warning level
    ToolWornOut         : BOOL := FALSE;                        //Active Tool is worn out
    ToolInvalid         : BOOL := FALSE;                        //no count, Act.Tool is invalid "SD.SysTool.ActToolValid=FALSE"
    EdgeInvalid         : BOOL := FALSE;                        //no count, Act.Tool has no valid edge "SD.SysTool.ActEd<=0 or >MT_MAX_EDGE"
    SDSysTool           : MT_SD_SYS_TOOL;                       //Value SD.SysTool, contains active tool and active edge (basis for database access)
    DBEdgeLife          : ARRAY [0..MT_MAX_EDGE] OF MT_EDGE_LIFE;  //new calculated edge life data
    DBToolLife          : MT_TOOL_LIFE;                         //new calculated tool life data and the cyc/min information
END_VAR
```

Fig. 14-131: Declaration of MT_TmToolLife

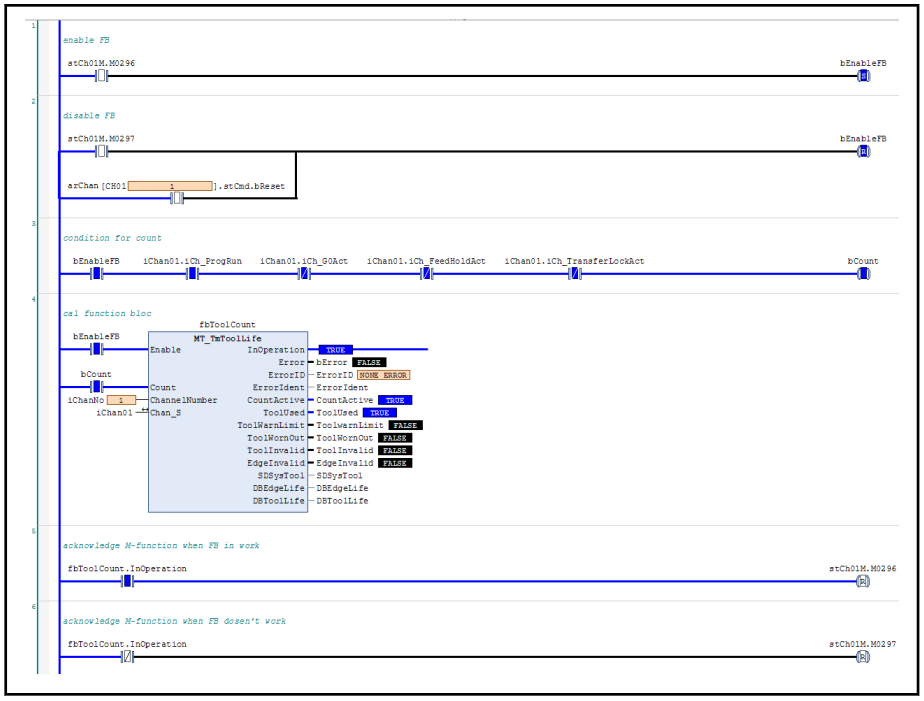


Fig. 14-132: Implementing MT_TmToolLife

fbToolCount	MT_TmToolLife		functionbloc
bEnableFB	BOOL	TRUE	enable FB
bCount	BOOL	TRUE	start count
iChanNo	UINT	1	channel number
bError	BOOL	FALSE	functionbloc error occur, for detailed infomation view ErrorIdent
ErrorID	ERROR_CODE	NONE_ERROR	basic error information
ErrorIdent	ERROR_STRUCT		detailed error information
CountActive	BOOL	TRUE	Life count active
ToolUsed	BOOL	TRUE	Active tool has already been used
ToolWarnLimit	BOOL	FALSE	Active tool has reached the warning level
ToolWornOut	BOOL	FALSE	Active Tool is worn out
ToolInvalid	BOOL	FALSE	no count, Act.Tool is invalid "SD.SysTool.ActToolValid=FALSE"
EdgeInvalid	BOOL	FALSE	no count, Act.Tool has no valid edge "SD.SysTool.ActEd<=0 or >MT_MAX_EDGE"
SDSysTool	MT_SD_SYS_TOOL		Value SD.SysTool, contains active tool and active edge (basis for database acces
ActToolValid	BOOL	TRUE	
ActEd	DINT	1	
ActTool	MT_RECORD_HEADER_DATA		
K1	INT	1	Key1.
K2	INT	1	Key2
SKQ	STRING(31)	'Stufenbohrer 15/5 mm'	String
IKQ1	DINT	0	Integerwert 1 (Schlüssel-Wert)
IKQ2	DINT	1005	Integerwert 1 (Schlüssel-Wert)
IKQ3	DINT	18	Integerwert 1 (Schlüssel-Wert)
IQ1	DINT	0	Integerwert 1
IQ2	DINT	0	Integerwert 1
IQ3	DINT	0	Integerwert 1
BQ1	DWORD	0	Bitfeld 1
BQ2	DWORD	6	Bitfeld 1
BQ3	DWORD	27788289	Bitfeld 1
PreTool	MT_RECORD_HEADER_DATA		
DBEdgeLife	ARRAY [0..MT_MAX_EDGE] OF MT_EDGE_LIFE		new calculated edge life data
DBEdgeLife[0]	MT_EDGE_LIFE		
DBEdgeLife[1]	MT_EDGE_LIFE		
MonOn	BOOL	TRUE	
RemLife	REAL	99.3207	
WarnLife	REAL	20	
MaxLife	REAL	100	
DBEdgeLife[2]	MT_EDGE_LIFE		
MonOn	BOOL	TRUE	
RemLife	REAL	97.57322	
WarnLife	REAL	20	
MaxLife	REAL	100	
DBToolLife	MT_TOOL_LIFE		new calculated tool life data and the cyc/min information
LifeU	STRING(3)	'min'	
RemTlLife	REAL	97.57322	

Fig. 14-133: Example data SDSysTool, DBEdgeLife, DBToolLife if MT_ToolLife is active

14.6.3 MT_TmChGetActiveEdge, Determining active edge of the active tool

Brief description Function to determine the active edge of the active tool.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-117: Library names according to the target systems

Interface description

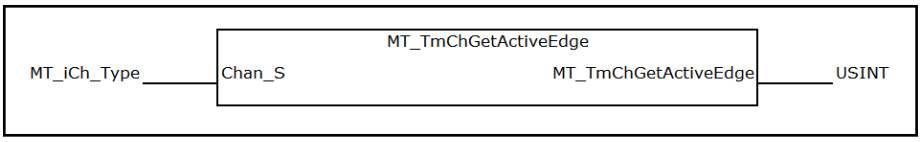


Fig. 14-134: MT_TmChGetActiveEdge function block

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

The active edge of the active tool is controlled via the channel interface (MT_iCh_Type) by the signals "iCh_ActFunc09...13" from the NC to the PLC. This function decodes this information and displays the result at the output of the function.

Troubleshooting

No error handling required for this function.

Implementation example

The operation mode of the first channel is determined in the example program.

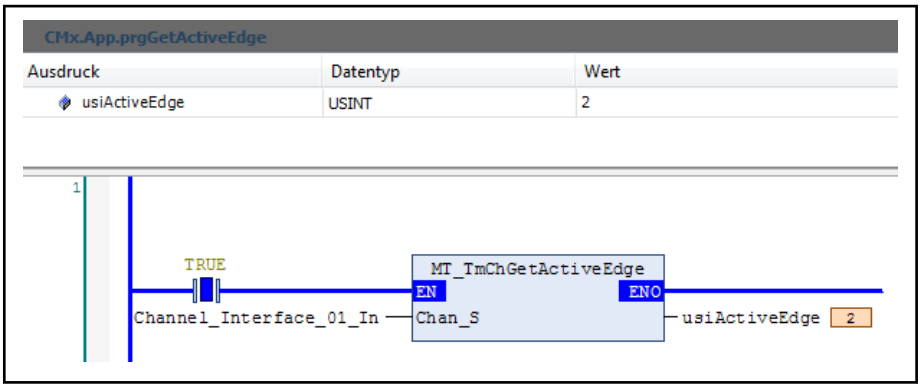


Fig. 14-135: Implementing MT_TmChGetActiveEdge

14.7 Miscellaneous

14.7.1 MT_DistanceAccumulator, Calculates the path distance from the specified position data

Brief description

Function block to determine the traveled distance using position data.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-118: Library names according to the target systems

Interface description

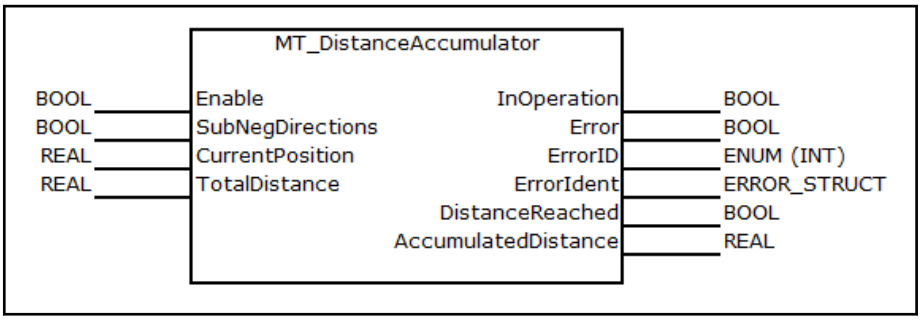


Fig. 14-136: MT_DistanceAccumulator function block

	Name	Type	Description
VAR_INPUT	Enable	BOOL	Enable function block
	SubNegDirections	BOOL	Motions in negative direction "FALSE=add", “TRUE=subtract"
	CurrentPosition	REAL	Current position values
	TotalDistance	REAL	Path distance that can be reached
VAR_OUTPUT	InOperation	BOOL	Processing active
	Error	BOOL	Processing incorrect
	ErrorID	ENUM (INT)	Rough error information
	ErrorIdent	ERROR_STRUCT	Detailed error information (only valid if "bError" = TRUE)
	DistanceReached	BOOL	The path distance was reached specified under "TotalDistance"
	AccumulatedDistance	REAL	Currently traveled path distance

Tab. 14-119: Inputs/outputs function block MT_DistanceAccumulator

Signal-time diagram

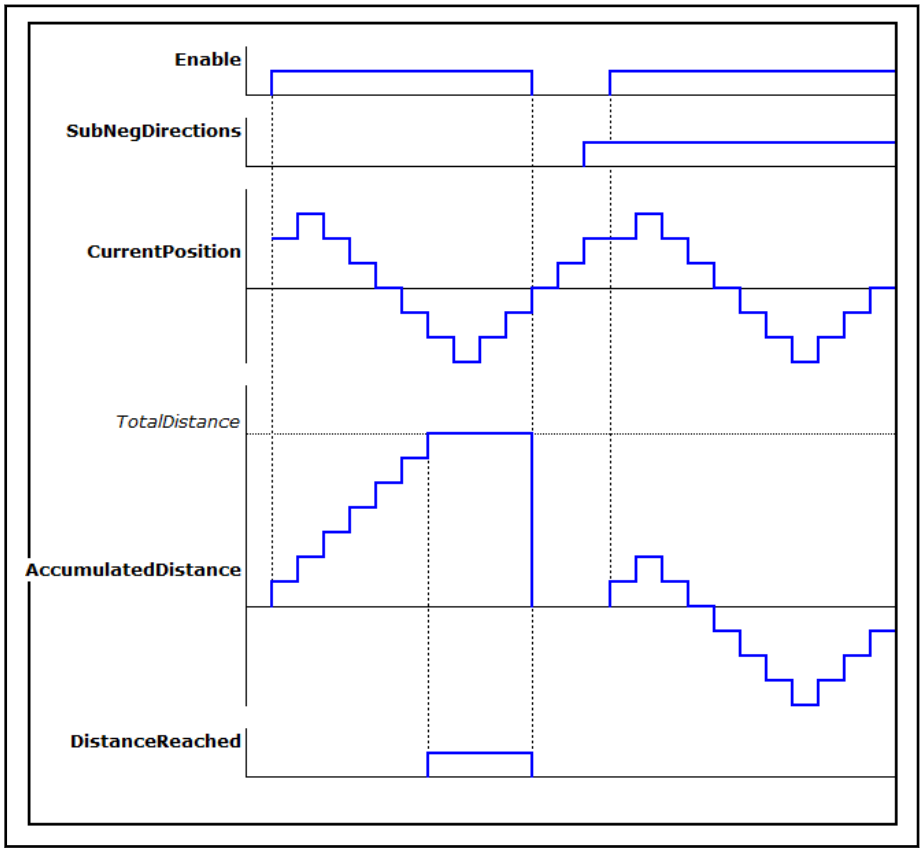


Fig. 14-137: Signal-time diagram MT_DistanceAccumulator

Functional description

This function block calculates the traveled distances from the differences of the position values specified at the "CurrentPosition" input and returns this distance at the "AccumulatedDistance" input. Via the "SubNegDirections" input, it can be specified if the motions are to be added in negative direction or if they have to be subtracted. Via the "TotalDistance" input, how much distance can be traveled ("AccumulatedDistance") before the "DistanceReached" output is set.

The following applies: If the specified total distance ("TotalDistance") is greater or equal to zero, "DistanceReached" is only set to TRUE as soon as the calculated distance ("AccumulatedDistance") is greater or equal to the specified total distance ("TotalDistance").

If the specified total distance ("TotalDistance") is smaller than zero, "DistanceReached" is only set to TRUE as soon as the calculated distance ("AccumulatedDistance") is smaller or equal to the specified total distance ("TotalDistance").

As soon as "TotalDistance" is TRUE, the measurement is completed and "AccumulatedDistance" retains the last determined value. To start a new measurement, the "Enable" input has to be reset (0-->1).

Troubleshooting

Currently, no error codes available for this function block.

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_DistanceAccumulator"

Error code (hex)	Description
0x00000000	No error code available

Tab. 14-120: Error of "MT_DistanceAccumulator" of "ErrorIdent.Additional1", valid if "Error"==TRUE

Implementation example

In the following example, first the position of drive 1 is read by the MTX control. To determined the traveled distance, the position value is subsequently applied to the "CurrentPosition" input of the "MT_DistanceAccumulator" function block.

```
PROGRAM prgTestDistanceAccumulator
VAR
    //Get Drive Position
    fbGetPosData                : MT_ScsPos;            //FB to get the drive position
    //Input
    bGetPosDataStart            : BOOL :=FALSE;         //start
    iGetPosDataChan             : INT  :=1;             //channel
    //Output
    stPositionData              : ARRAY [1..4] OF DINT; //position data
    bGetPosDataActive           : BOOL :=FALSE;         //read position active
    bGetPosDataError            : BOOL :=FALSE;         //error read position
    iGetPosDataErrorNo          : INT  :=0;             //error detail
    iGetPosDataNofElem          : INT  :=0;             //number received elements

    //calculate how far the drive has moved
    fbDistanceAccumulator       : MT_DistanceAccumulator;//FB to calculate how far the drive has moved
    //Input
    bEnable                     : BOOL :=FALSE;         //enable the distance accumulator
    rTotalDistance              : REAL := 0;            //total distance to travel
    rCurrentPosition            : REAL := 0;            //current drive position
    bSubNegDirections           : BOOL := FALSE;        //TRUE=subtract negative difference from accumulated distance,
                                                        //FALSE=add negative difference from accumulated distance

    //Output
    bInOperation                : BOOL :=FALSE;         //FB active
    bError                      : BOOL :=FALSE;         //FB error
    enErrorID                   : ERROR_CODE;           //FB error information
    stErrorIdent                : ERROR_STRUCT;         //FB detailed error information
    bDistanceReached            : BOOL := FALSE;        //Calculations are completed
    rAccumulatedDistance        : REAL := 0.0;          //Distance traveled
END_VAR
```

Fig. 14-138: Declaration MT_DistanceAccumulator

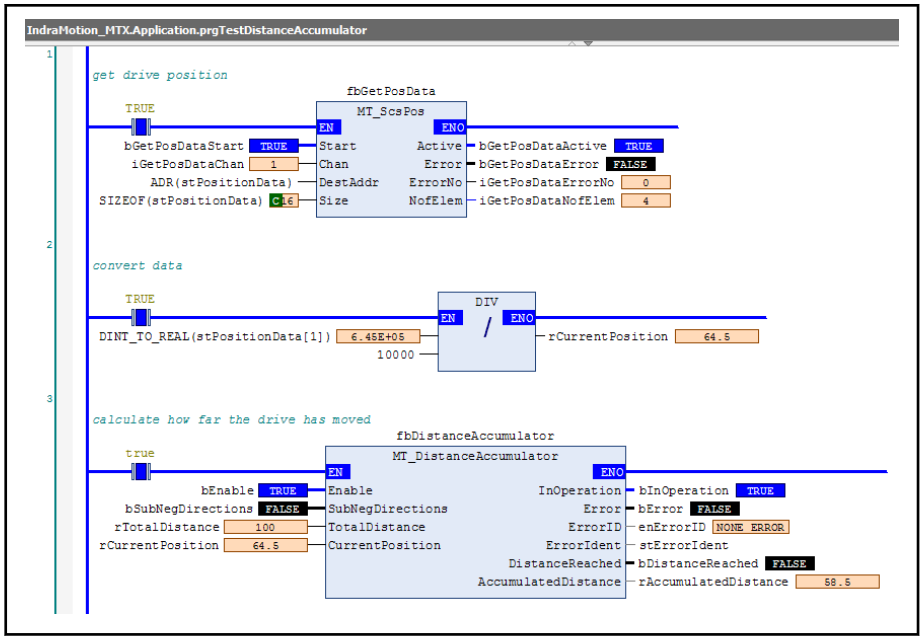


Fig. 14-139: Implementation MT_DistanceAccumulator

14.7.2 MT_TimeAccumulator, Calculates the lapsed time

Brief description

The function block determines the time in which a defined input is in state TRUE.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-121: Library names according to the target systems

Interface description

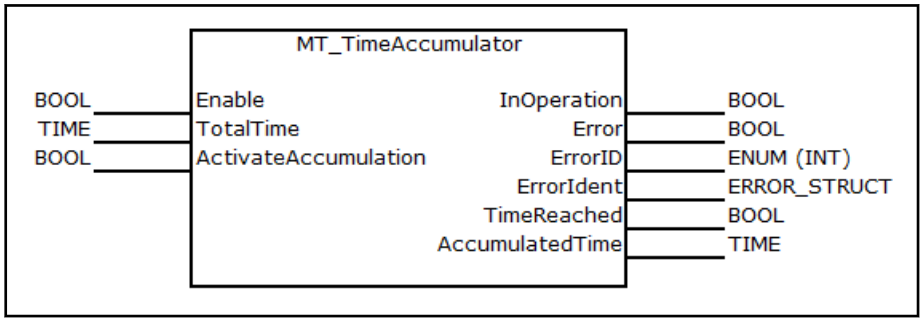


Fig. 14-140: Function block MT_TimeAccumulator

	Name	Type	Init	Description
VAR_INPUT	Enable	BOOL	FALSE	Enable function block
	TotalTime	TIME	T#49d17h2m47s295ms	Time that can be reached Initial value corresponds to maximum value for data type TIME
	ActivateAccumulation	BOOL	FALSE	Time is counted

	Name	Type	Init	Description
VAR_OUTPUT	InOperation	BOOL	FALSE	Processing active
	Error	BOOL	FALSE	Processing incorrect
	ErrorID	ENUM (INT)	NONE_ERROR	Rough error information
	ErrorIdent	ERROR_STRUCT	--	Detailed error information (only valid if "bError" = TRUE)
	TimeReached	BOOL	FALSE	The time specified under "TotalTime" was reached
	AccumulatedTime	TIME	T#0MS	Currently lapsed time

Tab. 14-122: Inputs/outputs function block MT_TimeAccumulator

Signal-time diagram

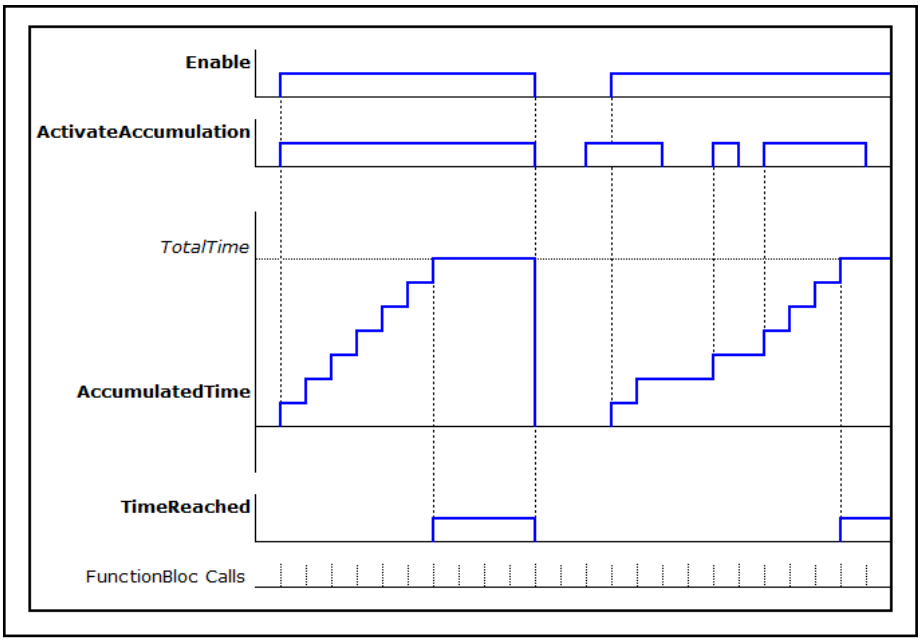


Fig. 14-141: Signal-time diagram MT_TimeAccumulator

Functional description

The function block determines the time in which the input "ActivateAccumulation" is in state TRUE. As long as "ActivateAccumulation" is TRUE, the system time is read upon each function block call. The time that has lapsed between two calls is added to the "AccumulatedTime". Via the "TotalTime" input, it is defined how much time can lapse ("AccumulatedTime") before the "TimeReached" output is set.

As soon as "TimeReached" is TRUE, the measurement is completed and "AccumulatedTime" retains the last determined value. To start a new measurement, the "Enable" input has to be reset (0-->1).

Troubleshooting

Currently, no error codes available for this function block.

Diagnostics

References to the diagnostic table

- MTX_TABLE

Error of "MT_DistanceAccumulator"

Error code (hex)	Description
0x00000000	No error code available

Tab. 14-123: Error of "MT_TimeAccumulator" of "ErrorIdent.Additional1", valid if "Error"==TRUE

Implementation example

The motion time of an axis is monitored in the following example. After 10 s, the axis is lubricated and the monitoring is subsequently restarted.

```
PROGRAM prgTestTimeAccumulator
VAR
    //calculate
    fbTimeAccumulator           : MT_TimeAccumulator ;  //FB to calculate
    //Input
    bEnable                     : BOOL := FALSE;        //enable the distance accumulator
    tTotalTime                  : TIME := T#10S;        //total time
    bActivateAccumulation       : BOOL := FALSE;        //accumulation runs,
    //Output
    bInOperation                : BOOL := FALSE;        //FB active
    bError                      : BOOL := FALSE;        //FB error
    enErrorID                   : ERROR_CODE;           //FB error information
    stErrorIdent                : ERROR_STRUCT;         //FB detailed error information
    bTimeReached                : BOOL := FALSE;        //Calculations are completed
    tAccumulatedTime            : TIME := T#0H0M0S;     //travel time

    //other
    bLubricateAxisStart         : BOOL := FALSE;        //start lubrication
    bLubricateAxisActive        : BOOL := FALSE;        //lubrication runs
END_VAR
```

Fig. 14-142: Declaration MT_TimeAccumulator

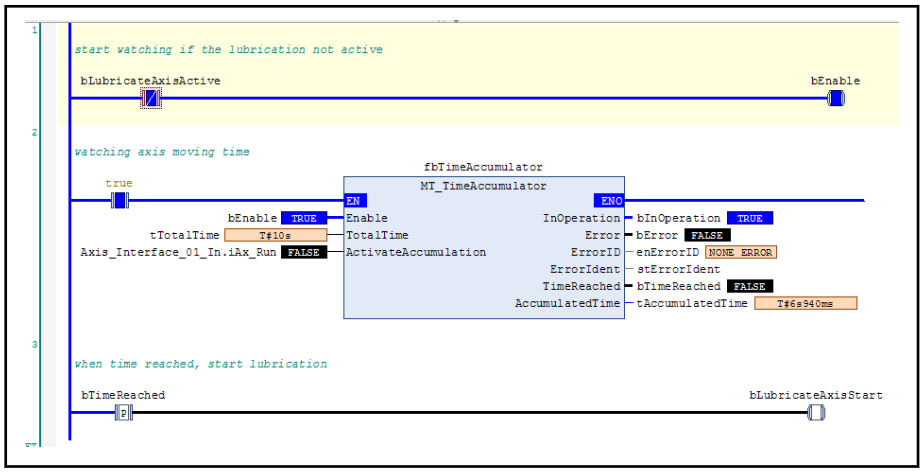


Fig. 14-143: Implementation MT_TimeAccumulator

14.7.3 MT_GetRandomNumber, Generates an integer random number

Brief description

The function calculates an integer random number within the specified minimum and maximum values.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-124: Library names according to the target systems

Interface description

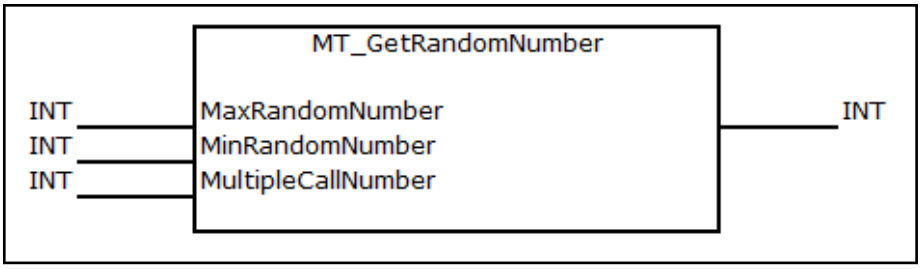


Fig. 14-144: Function block MT_GetRandomNumber

	Name	Type	Init	Description
VAR_INPUT	MaxRandomNumber	INT	32767	Maximum valid random number
	MinRandomNumber	INT	-32768	Lowest valid random number
	MultipleCallNumber	INT	0	Different values during multiple call in a PLC cycle
VAR_OUTPUT	MT_GetRandomNum-ber	INT	0	Random number

Tab. 14-125: Inputs/outputs function block MT_GetRandomNumber

Signal-time diagram

No special behavior that requires a signal-time diagram.

Functional description

This function generates a random number upon each call. The inputs "MaxRandomNumber" and "MinRandomNumber" specify which maximum and minimum value the random number can assume. In case the function is called multiple times in a PLC cycle, the "MultipleCallNumber" input has to have a different value upon each call.

Implementation example

In the following example, the function is called multiple times in a cycle. Upon the first call, a random number between -1000 and +1000 is generated. The second call only generates random numbers between +20 and +90.

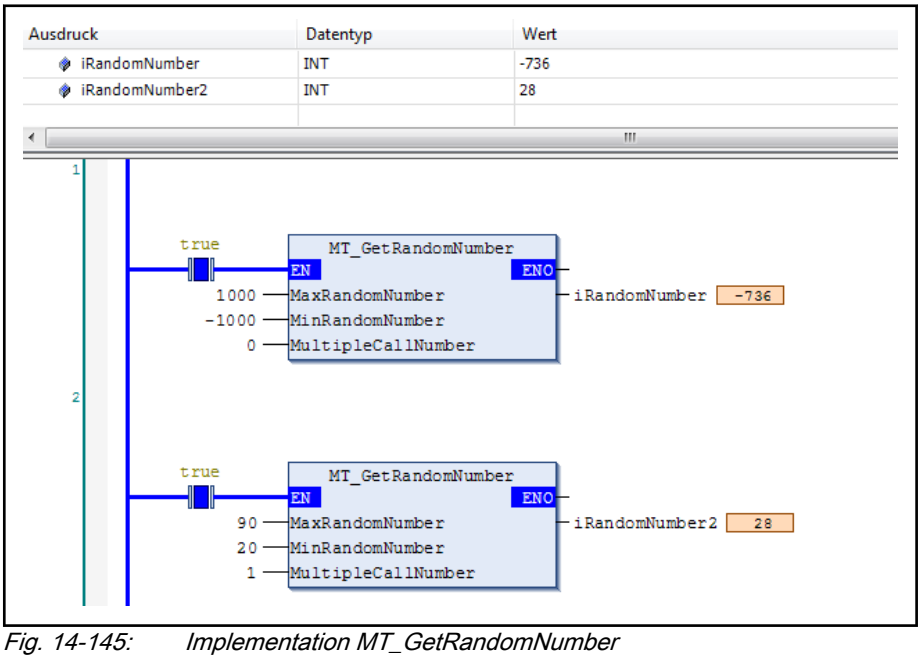


Fig. 14-145: Implementation MT_GetRandomNumber

14.8 Data types

14.8.1 MT_CHANNEL_OPERATION_MODE, Structure with all channel operation modes

Brief description The structure contains all available operation modes of an NC channel.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional

Tab. 14-126: Library names according to the target systems

Type definition

```
TYPE MT_CHANNEL_OPERATION_MODE :
STRUCT
(*main operation mode*)
    Automatic         :BOOL:=FALSE;    (*automatic operation active*)
    Manual            :BOOL:=FALSE;    (*manual operation active*)
    Mdi               :BOOL:=FALSE;    (*MDI operation active (manual input nc block)*)

(*sub operation mode*)
    NoOPMode          :BOOL:=FALSE;(*no operation mode active*)

    (*'manual operation': sub operation modes*)
    ManSetup          :BOOL:=FALSE; (*manual setup, jogging operation active*)
    ManRefPoint       :BOOL:=FALSE; (*manual basingpoint*)
    ManWcsSetup       :BOOL:=FALSE; (*manual setup, jogging operation with tool coordinate system active*)

    (*'automatic operation': sub operation modes*)
    AutoContBlock     :BOOL:=FALSE;  (*automatic, continuous block active*)
    AutoProgBlock     :BOOL:=FALSE;  (*automatic, program block active*)
    AutoSingleStep    :BOOL:=FALSE;  (*automatic, single step active*)
    AutoSingleBlock   :BOOL:=FALSE;  (*automatic, single block active*)
    AutoRetContour    :BOOL:=FALSE;  (*automatic, return to contur active*)
    AutoBlockForerun :BOOL:=FALSE;  (*automatic, block forerun*)

    AutoCplProgBlock :BOOL:=FALSE;  (*automatic, CPL debugger program block active*)
    AutoCplContBlock :BOOL:=FALSE;  (*automatic, CPL debugger continuous block active*)
END_STRUCT
END_TYPE
```

14.8.2 MT_CHANNEL_STATE, Structure contains all channel states

Brief description The structure contains all available states of an NC channel.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional.lib

Tab. 14-127: Library names according to the target systems

Type definition

```
TYPE MT_CHANNEL_STATE :
STRUCT
 Invalid :BOOL;  (*no valid channelstatus identified*)
 Inactiv :BOOL;  (*no process active, any process can be preselected*)
 Ready   :BOOL;  (*a process is selected an ready to start*)
 Reset   :BOOL;  (*process reset is active, nothing orders accepted as far as the process is 'inactive'*)
 Error   :BOOL;  (*process error occurred which can only repaired with channel reset or program deselect*)

 CancelDistancePrepare:BOOL;  (*process canceling distance prepare active*)
 CancelDistanceRunning:BOOL;  (*process is active and processed available buffer new*)

 ProgSelActive   :BOOL;  (*a program is selected and is in preparation (e.g. linking)*)
 ProgNextBlkReady:BOOL;  (*programm ready for next nc block, on buffered nc block selecting there are
                           all blocks processed, waiting for the next presetting*)
 ProgStartReady  :BOOL;  (*a process is selected, the nc program is at the beginning and ready for start*)
 ProgRunning     :BOOL;  (*a process is selected, a nc program is running and processed a nc block*)
```

```
END_STRUCT
END_TYPE
```

14.8.3 MT_SD_SYS_TOOL, Structure of the SD.SysTool system date

Brief description

The NC tool change program (TCH) uses the SD.SysTool system date contained in the functional scope. This structure is the matching image of this system date in the PLC.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional\Data Types\Ch

Tab. 14-128: Library names according to the target systems

Type definition

```
{attribute 'pack_mode' := '1'}
TYPE MT_SD_SYS_TOOL :
STRUCT
 ActToolValid :BOOL:=FALSE;           //active tool is valid
 ActEd        :DINT:=0;               //active edge
 ActTool      :MT_RECORD_HEADER_DATA; //active tool header data
 PreTool      :MT_RECORD_HEADER_DATA; //
END_STRUCT
END_TYPE
```

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_Base\Data Types\ToolManagement Default Types\IndraWorks 09Vxx

Tab. 14-129: Library names according to the target systems

Type definition

```
TYPE MT_RECORD_HEADER_DATA :
STRUCT
 K1    :INT;         //Sector
 K2    :INT;         //Place
 SKQ   :STRING(31); //Tool-ID
 IKQ1  :DINT;        //Duplo Number
 IKQ2  :DINT;        //Tool Type
 IKQ3  :DINT;        //Tool Number
 IQ1   :DINT;        //reserve
 IQ2   :DINT;        //reserve
 IQ3   :DINT;        //reserve
 BQ1   :DINT;        //Place Status (Bit Array)
 BQ2   :DINT;        //Tool Status    (Bit Array)
 BQ3   :DINT;        //Tooltype description    (Bit Array)
END_STRUCT
END_TYPE
```

14.8.4 MT_AXIS_ASYNC_OPERATION_MODE, Structure with all axis operation modes

Brief description

The structure contains all available operation modes of an asynchronous axis.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional\Data Types\Drive

Tab. 14-130: Library names according to the target systems

Type definition

```
TYPE MT_AXIS_ASYNC_OPERATION_MODE :
STRUCT
 ProgMode     :BOOL; (*axis can be programed by a channel*)
 ManSetup     :BOOL; (*manual setup, jogging operation active*)
 ManRefPoint  :BOOL; (*manual basingpoint*)
 InvalidState :BOOL; (*no valid status identified*)
END_STRUCT
END_TYPE
```

14.8.5 MT_AXIS_JOG_MODE, Structure with jog distances of an axis

Brief description The structure contains all available jog distances of an axis.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional\Data Types\Drive

Tab. 14-131: Library names according to the target systems

Type definition

```
TYPE MT_AXIS_JOG_MODE :
STRUCT
 NoJogMode          :BOOL:=FALSE;    (*no JogMode aktiv*)
 (*incremental JogModi*)
 JogMode1           :BOOL:=FALSE; (*JogMode 1 increment active*)
 JogMode10          :BOOL:=FALSE; (*JogMode 10 increment active*)
 JogMode100         :BOOL:=FALSE; (*JogMode 100 increment active*)
 JogMode1000        :BOOL:=FALSE; (*JogMode 1000 increment active*)
 JogModeX           :BOOL:=FALSE; (*Jogging respective the increment guidelines in machine parameter active*)
 (*continuous JogModi*)
 JogModeContRapid   :BOOL:=FALSE; (*rapid continuous jogging active*)
 JogModeContFast    :BOOL:=FALSE; (*fast continuous jogging active*)
 JogModeContMiddle  :BOOL:=FALSE; (*middle continuous jogging active*)
 JogModeContSlow    :BOOL:=FALSE; (*slow continuous jogging active*)
END_STRUCT
END_TYPE
```

14.8.6 MT_DRIVE_ERROR_CLASS_1, Structure error details of S-0-0011 of an axis

Brief description The structure contains the error information bit-coded in S-0-0011.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional\Data Types\Drive

Tab. 14-132: Library names according to the target systems

Type definition

```
TYPE MT_DRIVE_ERROR_CLASS_1 :
STRUCT
 //S-0-0011
 OverloadShutdown                  :BIT:=FALSE; //Bit0
 AmplifierOvertemperatureShutdown  :BIT:=FALSE; //Bit1
 MotorShutdownTemperature          :BIT:=FALSE; //Bit2
 ReservedBit03                     :BIT:=FALSE; //Bit3
 ControlVoltageError               :BIT:=FALSE; //Bit4
 EncoderError                      :BIT:=FALSE; //Bit5
 ElectronicCommunicationError      :BIT:=FALSE; //Bit6
 OverCurrentError                  :BIT:=FALSE; //Bit7
 OverVoltageError                  :BIT:=FALSE; //Bit8
 UnderVoltageError                 :BIT:=FALSE; //Bit9
 PhaseErrorInPowerSupply           :BIT:=FALSE; //Bit10
 ExcessiveControllerDevitation     :BIT:=FALSE; //Bit11
 CommunicationError                :BIT:=FALSE; //Bit12
```

```
 PositionLimitValueExceeded       :BIT:=FALSE; //Bit13
 ReservedBit14                    :BIT:=FALSE; //Bit14
 ManufacturerSpecificError        :BIT:=FALSE; //Bit15
END_STRUCT
END_TYPE
```

14.8.7 MT_DRIVE_ERROR_CLASS_2, Structure warning details of S-0-0012 of an axis

Brief description The structure contains the detailed information about a warning bit-coded in S-0-0012.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional\Data Types\Drive

Tab. 14-133: Library names according to the target systems

Type definition

```
TYPE MT_DRIVE_ERROR_CLASS_2 :
STRUCT
 //S-0-0012
 OverloadWarning                  :BIT:=FALSE; //Bit0
 AmplifierOvertemperatureWarning  :BIT:=FALSE; //Bit1
 MotorOvertemperatureWarning      :BIT:=FALSE; //Bit2
 ReservedBit03                    :BIT:=FALSE; //Bit3
 ReservedBit04                    :BIT:=FALSE; //Bit4
 PositioningVelocityGreaterLimit  :BIT:=FALSE; //Bit5
 ReservedBit06                    :BIT:=FALSE; //Bit6
 ReservedBit07                    :BIT:=FALSE; //Bit7
 ReservedBit08                    :BIT:=FALSE; //Bit8
 UndervoltageInDCBus              :BIT:=FALSE; //Bit9
 ReservedBit10                    :BIT:=FALSE; //Bit10
 ReservedBit11                    :BIT:=FALSE; //Bit11
 ReservedBit12                    :BIT:=FALSE; //Bit12
 TargetPositionOutsideOfPositionLimitValues :BIT:=FALSE; //Bit13
 ReservedBit14                    :BIT:=FALSE; //Bit14
 ManufacturerSpecificWarning      :BIT:=FALSE; //Bit15
END_STRUCT
END_TYPE
```

14.8.8 MT_DRIVE_POWER_STATE, Drive status

Brief description The structure contains the drive states provided by the MT_AxPower, MT_SpPower function blocks.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional\Data Types\Drive

Tab. 14-134: Library names according to the target systems

Type definition

```
TYPE MT_DRIVE_POWER_STATE :
STRUCT
 BB             :BOOL:=FALSE; //Status BB - drive ready
 AB             :BOOL:=FALSE; //Status AB - drive enabled for power activation
 AF             :BOOL:=FALSE; //Status AF - drive in operation
 DryRun         :BOOL:=FALSE; //Status DryRun - drive in dryrun mode
 Error          :BOOL:=FALSE; //Status Error   (class1) active - for details read S-0-0011
 ErrorDetails   :MT_DRIVE_ERROR_CLASS_1;//Error detail information
 Warning        :BOOL:=FALSE; //Status Warning (class2) active - for details read S-0-0012
 WarningDetails :MT_DRIVE_ERROR_CLASS_2;//Warning detail information
 Unknown        :BOOL:=FALSE; //Status unknown - drive unknown status
END_STRUCT
END_TYPE
```

14.8.9 MT_DRIVE_SAFE_OP_MODE, Structure contains the safety operating state of a drive

Brief description

The structure maps the safety-engineering operating state of a drive, as it is determined by the "MT_DrvGetSafeState" function block.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional\Data Types\Drive

Tab. 14-135: Library names according to the target systems

Type definition

```
TYPE MT_DRIVE_SAFE_OP_MODE :
STRUCT
 NO    : BOOL:=FALSE; //0: not active, 1: active / Normal operation
 SMES  : BOOL:=FALSE; //0: not active, 1: active / EMERGENCY STOP
 SMST  : BOOL:=FALSE; //0: not active, 1: active / Special mode safe standstill

 SMM1  : BOOL:=FALSE; //0: not active, 1: active / Special mode safe motion 1
 SMM2  : BOOL:=FALSE; //0: not active, 1: active / Special mode safe motion 2
 SMM3  : BOOL:=FALSE; //0: not active, 1: active / Special mode safe motion 3
 SMM4  : BOOL:=FALSE; //0: not active, 1: active / Special mode safe motion 4
 SMM5  : BOOL:=FALSE; //0: not active, 1: active / Special mode safe motion 5
 SMM6  : BOOL:=FALSE; //0: not active, 1: active / Special mode safe motion 6
 SMM7  : BOOL:=FALSE; //0: not active, 1: active / Special mode safe motion 7
 SMM8  : BOOL:=FALSE; //0: not active, 1: active / Special mode safe motion 8
 SMM9  : BOOL:=FALSE; //0: not active, 1: active / Special mode safe motion 9
 SMM10 : BOOL:=FALSE; //0: not active, 1: active / Special mode safe motion 10
 SMM11 : BOOL:=FALSE; //0: not active, 1: active / Special mode safe motion 11
 SMM12 : BOOL:=FALSE; //0: not active, 1: active / Special mode safe motion 12
 SMM13 : BOOL:=FALSE; //0: not active, 1: active / Special mode safe motion 13
 SMM14 : BOOL:=FALSE; //0: not active, 1: active / Special mode safe motion 14
 SMM15 : BOOL:=FALSE; //0: not active, 1: active / Special mode safe motion 15
 SMM16 : BOOL:=FALSE; //0: not active, 1: active / Special mode safe motion 16

 ParkingAxis            : BOOL:=FALSE;    //0: not active, 1: active / parking axis

 SMO_ApplicationActiv   : BOOL:=FALSE;    //0: not active, 1: active / safe motion application
 SMO_TransitionProcess  : BOOL:=FALSE;    //0: not active, 1: active / safe motion transition process
 SMO_ParameterMode      : BOOL:=FALSE;    //0: not active, 1: active / safe motion parameter mode
 SMO_ConfigurationMode  : BOOL:=FALSE;    //0: not active, 1: active / safe motion configuration mode
 SMO_BehaveIfNotActive  : BOOL:=FALSE;    //0: Condition as supplied,
          //1: Functional commissioning; output stage and brake(s) are activated by the standard firmware
        //Behavior with non-active "Safe Motion", only relevant if "SMO_ActivationStatus==FALSE"

    SMO_ActivationStatus   : BOOL:=FALSE; //1: not active, 0: active / safe motion functionality
    SafetyTechnologyError  : BOOL:=FALSE; //1: no error message, 0: Error message/Safety technology error

END_STRUCT
END_TYPE
```

14.8.10 MT_SPINDLE_JOG_SPEED, Structure with spindle jog velocities

Brief description

The structure contains all available jog velocities of a spindle.

Assignment: Target system/library

Target system	Library
IndraMotion_MTX	MT_BaseAdditional\Data Types\Drive

Tab. 14-136: Library names according to the target systems

Type definition

```
TYPE MT_SPINDLE_JOG_SPEED :
STRUCT
 (*Variables for the current SpeedMode*)
 Speed1   :BOOL; (*spindle jogging speed 1*)
 Speed2   :BOOL; (*spindle jogging speed 2*)
 Speed3   :BOOL; (*spindle jogging speed 3*)
```

```
 Speed4   :BOOL; (*spindle jogging speed 4*)
 Speed5   :BOOL; (*spindle jogging speed 5*)
 Speed6   :BOOL; (*spindle jogging speed 6*)
 Speed7   :BOOL; (*spindle jogging speed 7*)
 Speed8   :BOOL; (*spindle jogging speed 8*)
END_STRUCT
END_TYPE
```

15 MT_PLCopen.library

15.1 Introduction and overview

The MT_PLCopen library contains essential function blocks for the implementation of the MTX system for Sercos-based axes.

The data types required by the function blocks at their interface (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT CONSTANT, VAR GLOBAL / EXTERNAL) are covered in the IEC standard EN 61131-3 or defined by the company in the following libraries:

- Refer to the chapter RIL_CommonTypes.library in the documentation "IndraLogic XLC, IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491), data types for general use in all IndraWorks systems.

MT_PLCopen function blocks, functions and data types

Area of application (folder in library)	
POUs	**Description**
POUs (parameter)	
MB_Command, page 351	Allows to execute a command
MB_ReadListParameter, page 356	Returns the value of a user-specific parameter. A pointer addressing a memory space of up to 64 KB is transferred to the function block as defined by the user.
MB_ReadMaxRealValue, page 358	Returns the maximum value allowed of a parameter specified by the user (REAL)
MB_ReadMaxValue, page 360	Returns the maximum value allowed of a parameter specified by the user (DINT)
MB_ReadMinRealValue, page 362	Returns the minimum value allowed of a parameter specified by the user (REAL)
MB_ReadMinValue, page 364	Returns the minimum value allowed of a parameter specified by the user (DINT)
MB_ReadName, page 366	Returns the parameter name specified by the user
MB_ReadParameter, page 367	Returns the value of a specified parameter (DINT)
MB_ReadRealParameter, page 369	Returns the value of a specified parameter (REAL)
MB_ReadSercosAttribute, page 371	Allows reading the Sercos attributes of a parameter
MB_ReadSercosDataStatus, page 372	Returns the status of a parameter specified by the user
MB_ReadStringParameter, page 374	Returns the value of a user-specific parameter (STRING, character string)
MB_ReadUnit, page 376	Returns the measuring unit of a parameter specified by the user
MB_WriteListParameter Refer to the chapter ML_PLCopen.library in the documentation "IndraLogic XLC, IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)	Writes memory content of defined length - starting from the initial address - to a specified parameter
MB_WriteParameter, page 378	Writes a value to a parameter specified by the user (DINT)

MB_WriteRealParameter, page 379	Writes a value to a parameter specified by the user (REAL)
MB_WriteStringParameter, page 381	Writes a character string with up to 255 characters to a user-specific parameter (STRING, character string)

If the respective function block name starts with "MC_", it means that it was synchronized the best possible with the PLCopen template and that there are thus comparable function blocks for different systems such as XLC/MLC, MLD, MTX.

If the respective function block name starts with "MB_", it means that there are comparable function blocks for different systems such as XLC, MLC, MLD, MTX.

If the respective function block name starts with "MT_", it means that it has been developed for the MTX system to support it best possible:

15.2 Global_Variables

15.2.1 Data types

IL_SIII_ATTRIBUTE, Structure contains all Sercos attributes

Brief description The structure contains all available attributes.

Type definition

```
TYPE IL_SIII_ATTRIBUTE :
STRUCT
  WrProtectP4 : BOOL;                //Operation date of the parameter in the operation mode
                                     //(Sercos phase 4 - P4) write-protected
  WrProtectP3 : BOOL;                //Operating date of the parameter in Sercos phase 3
                                     //(P3) write-protected
  WrProtectP2 : BOOL;                //Operation date of the parameter in the parameterization mode
                                     //(Sercos phase 2 - P2) write-protected
  DecPoints   : USINT;               //Number of decimal places of the operating date of the parameter
  DataType    : IL_SIII_DATA_TYPE;   //Data type and display format of the operating date of the parameter
  Cmd         : BOOL;                //Operating date calls for command function
  DataLength  : IL_SIII_DATA_LENGTH; //Data length of the operating date of the parameter
  Conversion  : UINT;                //Evaluation factor of the operating date of the parameter
END_STRUCT
END_TYPE
```

MB_SERCOS_DATA_STATUS, Structure contains all Sercos command states

Brief description The structure contains all available Sercos command states.

Type definition

```
TYPE MB_SERCOS_DATA_STATUS :
(
    MB_SERCOS_DATA_NOT_SET                := 0,            //Not set and not enabled
    MB_SERCOS_DATA_SETTING                := 1,            //Set and enabled
    MB_SERCOS_DATA_END_OK                    := 3,            //Command correctly executed
    MB_SERCOS_DATA_INTERRUPTED := 5,           //Command execution interrupted
    MB_SERCOS_DATA_IN_PROCESS     := 7,           //In process
    MB_SERCOS_DATA_BREAK_ERROR    := 15,        //Error, command execution impossible
    MB_SERCOS_DATA_DATUM_FAULT := 256        //Data invalid

);
END_TYPE
```

15.3 POUs

15.4 MB_Command

Brief description The function block "MB_Command" is used to execute, monitor and cancel commands.

Interface description

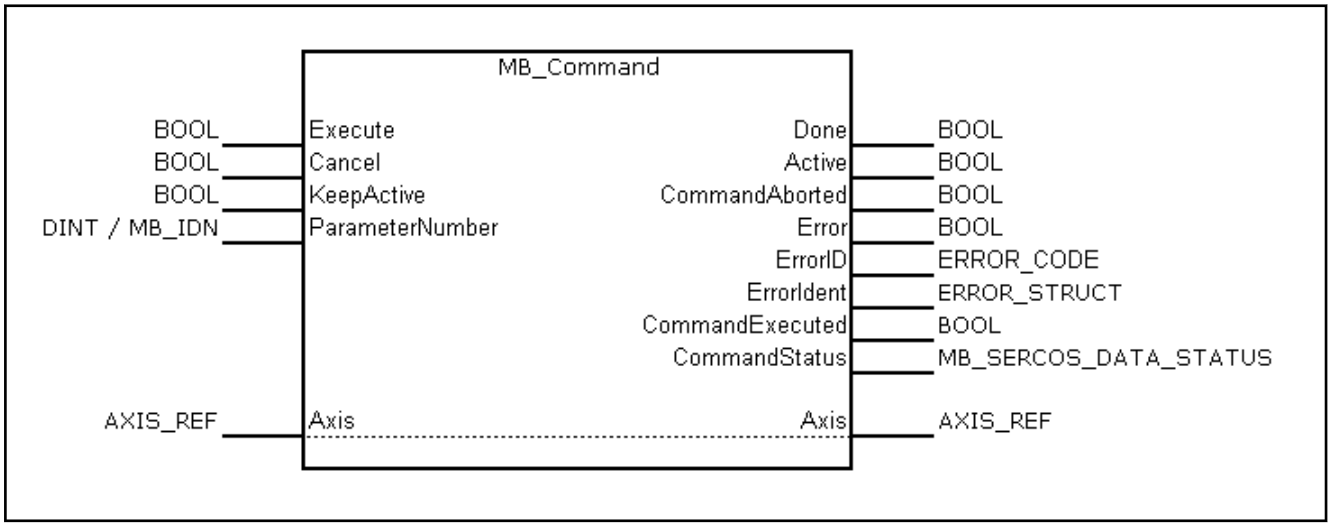


Fig. 15-1: Function block "MB_Command"

Interface variables

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Reference to the axis
VAR_INPUT	Execute	BOOL	Enabling the function block (once, edge-controlled)
	Cancel	BOOL	Canceling the command execution (once, edge-controlled)
	KeepActive	BOOL	Keep the activation of the command active after execution (permanent, state-controlled)
	ParameterNumber	DINT or MB_IDN (from MPx06)	Command

	Name	Type	Comment
VAR_OUTPUT	Done	BOOL	Command successfully executed and reset
	Active	BOOL	Command execution active
	CommandAborted	BOOL	Command successfully aborted and reset
	Error	BOOL	Error of the command execution
	ErrorID	ERROR_CODE	In case of a set "Error" output, this output contains a broad error classification
	ErrorIdent	ERROR_STRUCT	If the "Error" output is set, the output contains detailed error information See "Troubleshooting" on page 356
	CommandExecuted	BOOL	Command successfully completed
	CommandStatus	MB_SERCOS_DATA_STATUS	Status of the command execution

Tab. 15-1: Interface variables of the function block "MB_Command"

Signal-time diagram

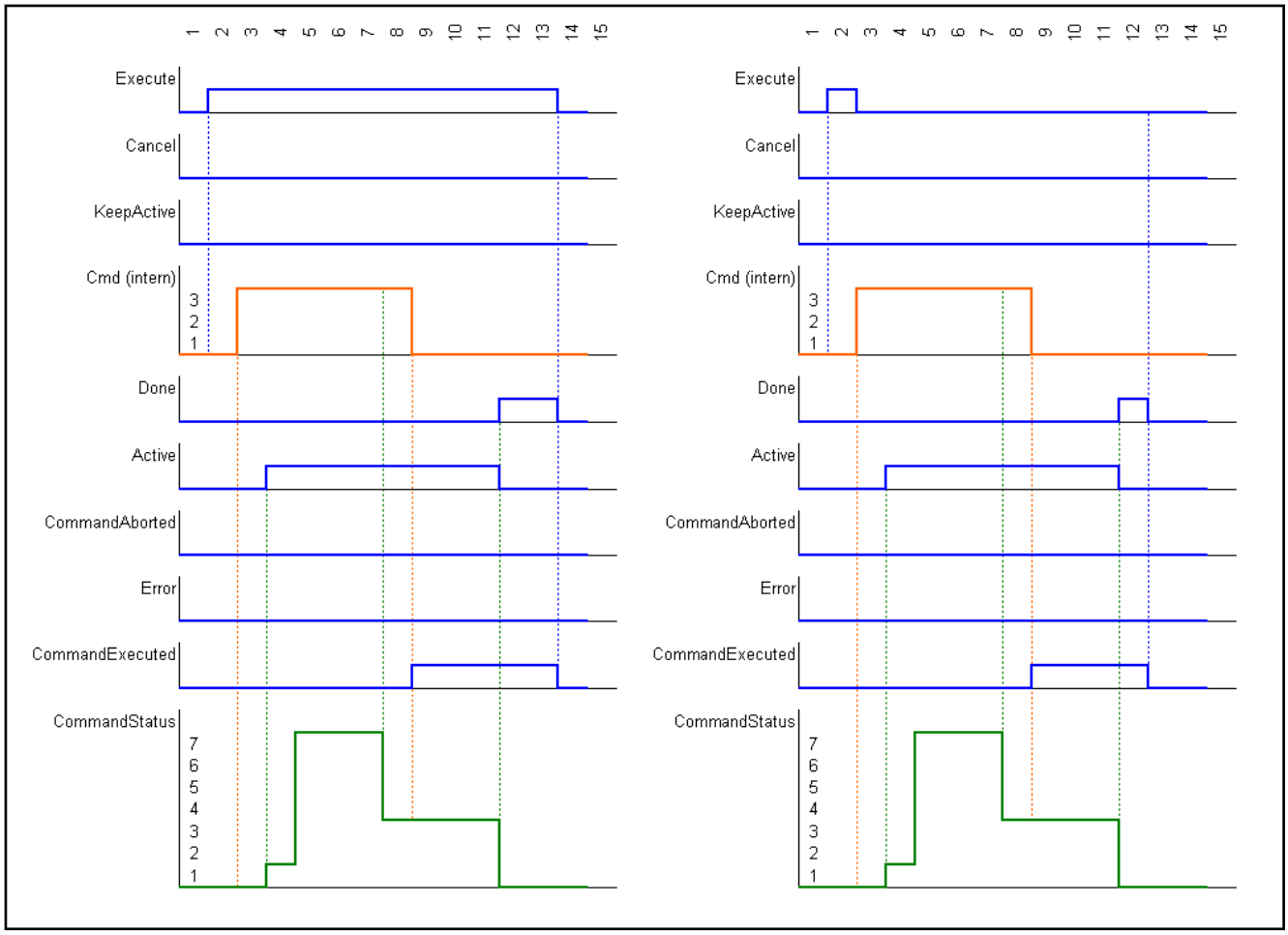


Fig. 15-2: Signal-time diagram for signal flow "Execute-Done"

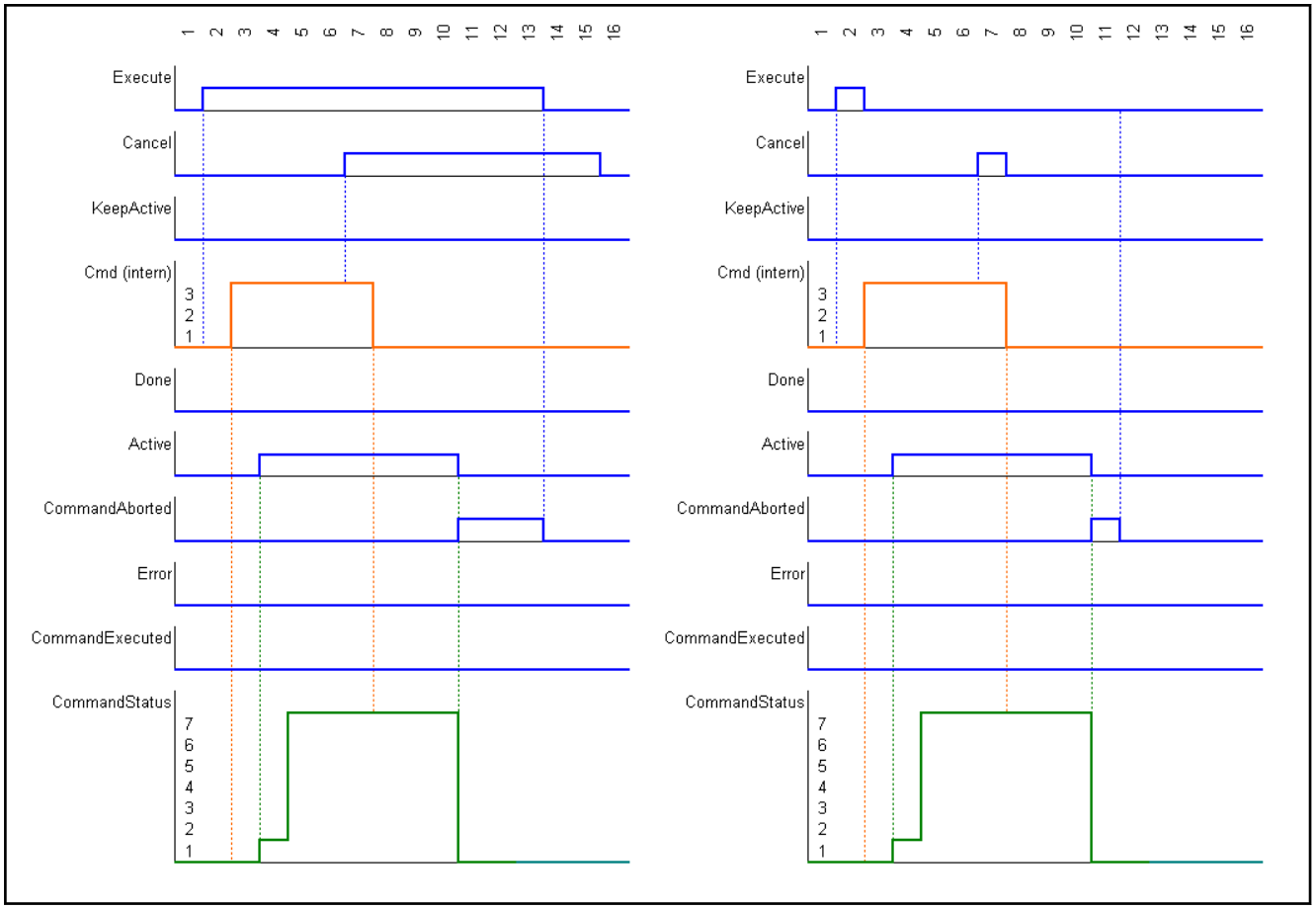


Fig. 15-3: Signal-time diagram for signal flow "Execute-Cancel-CommandAborted"

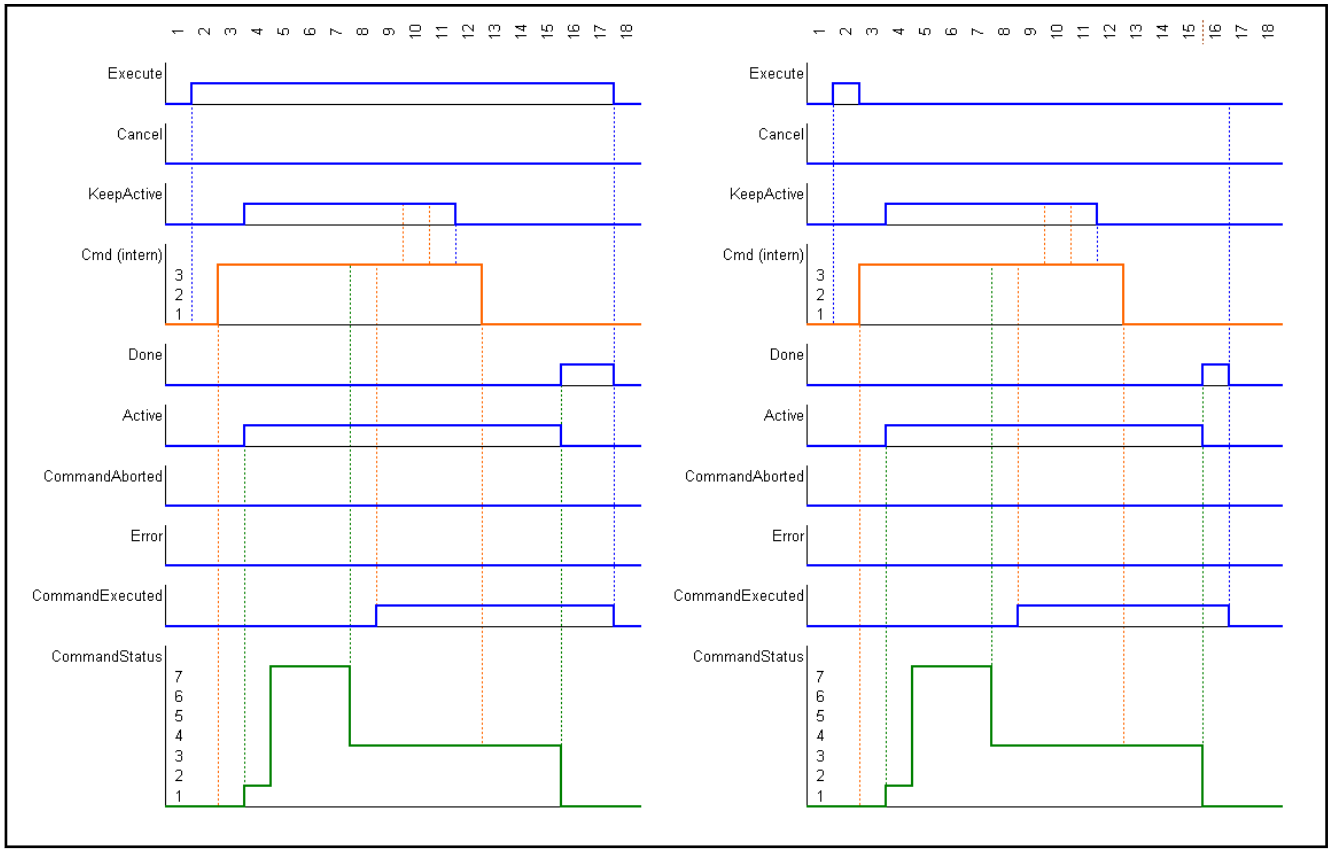


Fig. 15-4: Signal-time diagram for signal flow "Execute-KeepActive-Done"

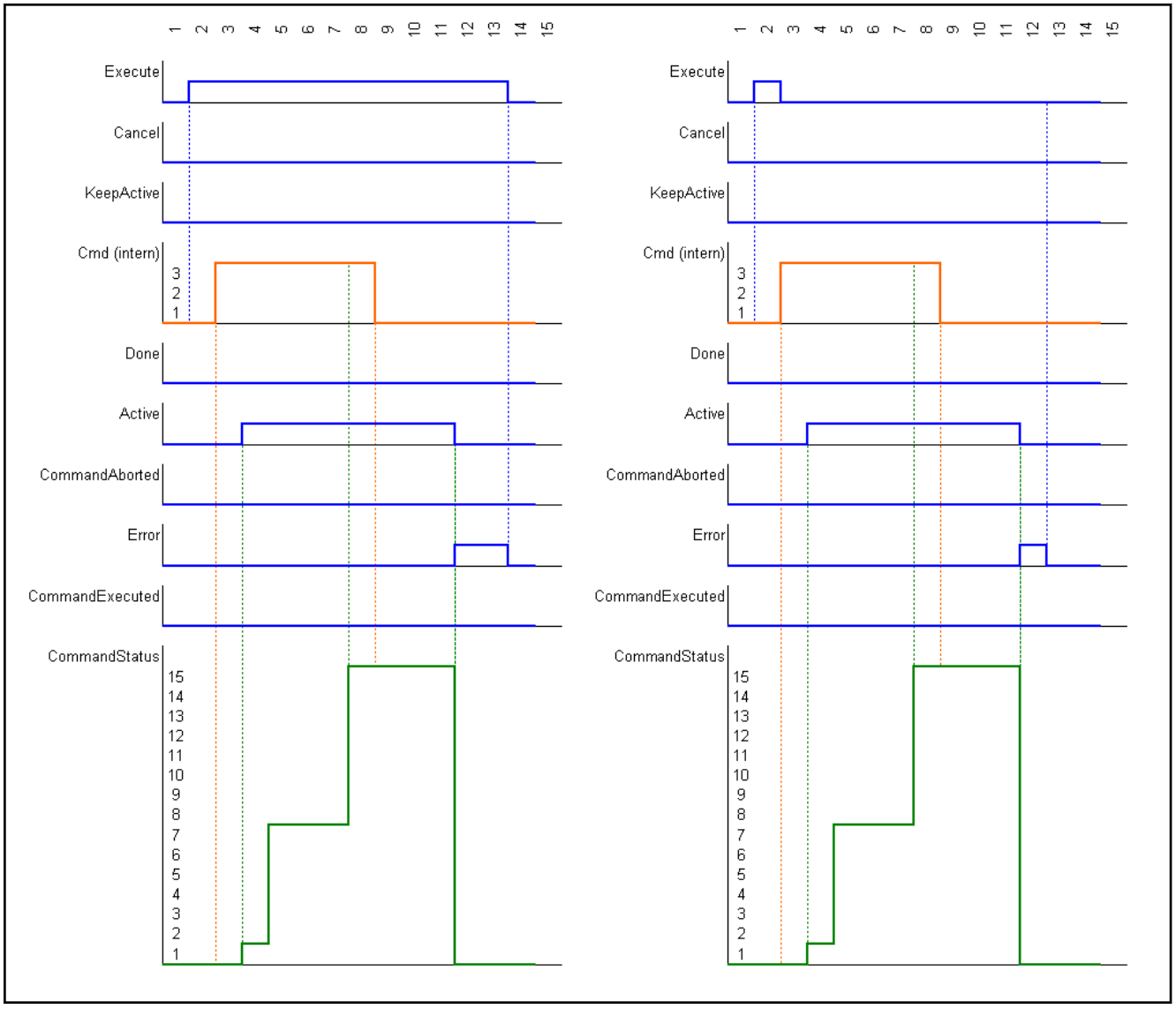


Fig. 15-5: Signal-time diagram for signal flow "Execute-Error"

Functional description

The function block "MB_Command" starts the execution of the command defined via the "Idn" input after having enabled processing via "ParameterNumber". In general, the command is automatically reset after processing. The command has to be specified in the parameter identification number format according to the Sercos standard. Apart from this direct specification, it is also possible to use a correspondingly predefined constant from the range of the global variables of the base library (e. g. "ML_Base.lib") or to use a conversion function from the "RIL_Utilities.lib" library. Depending on the respective command, its execution can be canceled at any time via the "Cancel" input. The "KeepActive" input allows the command to remain active after execution. If the input "KeepActive" is deactivated later on, the command is automatically reset. The successful completion or successful abort of a command is displayed via the outputs "Done" or "CommandAborted" for at least one call. The current status of the command execution is permanently displayed at the "CommandStatus" output and the completion of the command execution is signaled at the "CommandExecuted" output. The active processing of the command will be displayed over the output "Active" as well as with the help of the corresponding date at the output "CommandStatus". If an error

occurs during the processing of a command, this will be signaled over the output "Error" as well as over the error classification elements (ErrorID, ErrorIdent) for a period of at least one call.

Troubleshooting The function block "MB_Command" uses the error table **F_RELATED_TABLE**, 16#0170 and **INDRV_TABLE**, 16#0070. It can generate the errors specified in the following table in Additional1 and Additional2. In addition to this error, the implemented function blocks "MB_WriteParameter", "MB_ReadSercosDataStatus", "MB_ReadParameter" and "MB_ReadSercosAttribute" are able to report other errors.

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR, 16#0006	16#0002	16#0001	Invalid input - Retriggering the command execution is not supported by the function block MB_Command
INPUT_RANGE_ERROR, 16#0006	16#0002	16#0002	Invalid input - The inputs Cancel or KeepActive are not supported by control commands
INPUT_RANGE_ERROR, 16#0006	16#0002	16#0003	Invalid input - The input ParameterNumber does not contain a valid command.
DEVICE_ERROR, 16#0008	16#1701	16#0000	Error of the command execution (control)
STATE_MACHINE_ERROR, 16#0005	16#0006	16#????	Invalid state of the state machine 16#???? specifies the invalid state

Tab. 15-2: Error numbers of the error table "F_RELATED_TABLE" caused by the "MB_Command"

ErrorID	Additional1	Additional2	Description
DEVICE_ERROR, 16#0008	16#?????	16#0000	Command execution error (drive) - 16#???? contains the current value from S-0-0390.

Tab. 15-3: Error numbers of the error table "INDRV_TABLE" caused by the "MB_Command"

15.5 MB_ReadListParameter

Brief description The function block returns the value of a user-specific parameter. A pointer addressing a memory space of up to 64 kB is transferred to the function block as defined **by the user**. In addition, the required use data length is transferred (maximum number of readable bytes).

The return value is the number of bytes actually read.



The data type of the elements of the parameter to be read and the data type of the elements of the ARRAY have to match.

Library	Area
ML_PLCopen.lib	Parameter
15VRS	

Tab. 15-4: Library assignment

Interface description

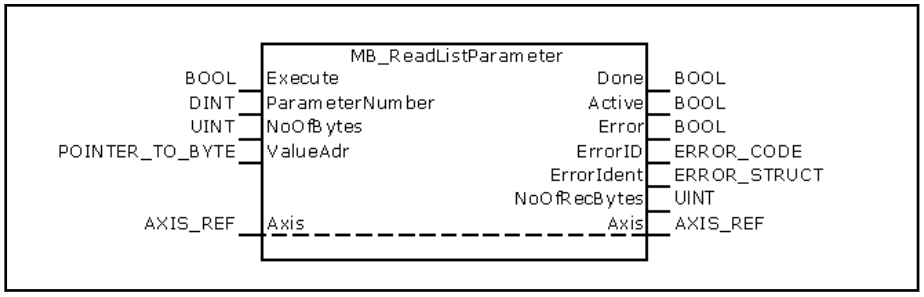


Fig. 15-6: MB_ReadListParameter FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF Refer to the chapter RIL_Common-Types.library in the documentation "IndraLogic XLC, IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)	Contains information on the specific axis
VAR_INPUT	Execute	BOOL	Reading the parameter is triggered by the positive edge at "Execute"
	Parameter number	DINT	Number of the parameter, Refer to the chapter PLCopen Function Blocks for the IndraMotion MLC 02VRS in the documentation "Rexroth IndraMotion MLC02VRS PLCopen Function Blocks and Data Types" Library Description, R911316107
	NoOf-Bytes	UINT	Maximum number of data to be read in bytes
	ValueAdr	POINTER_TO_BYTE	It points to the initial address from where the data read is to be stored (maximum NoOfBytes)
VAR_OUTPUT	Done	BOOL	Valid data is available
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE Refer to the chapter RIL_Common-Types.library in the documentation "IndraLogic XLC, IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)	Brief note on error cause

	Name	Type	Comment
	ErrorIdent	ERROR_STRUCT Refer to the chapter RIL_Common-Types.library in the documentation "IndraLogic XLC, IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)	Detailed error information
	NoOf ReadBytes	UINT	Number of bytes actually read

Tab. 15-5: Interface of the MB_ReadListParameter function block

Signal-time diagram

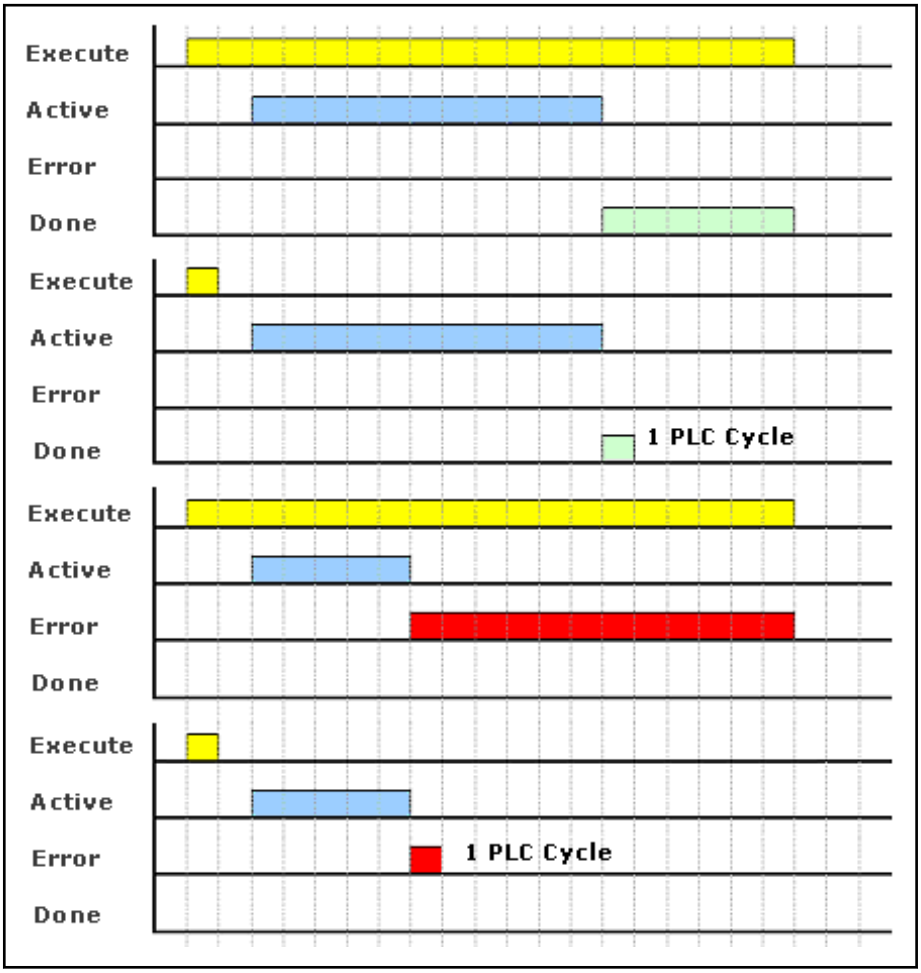


Fig. 15-7: Default signal-time diagram

15.6 MB_ReadMaxRealValue

Brief description

The function block returns the maximum value allowed of a parameter specified by the user. The return value is represented as REAL number.



Any change, e.g. of the parameter number or the axis at the input of the function block, requires another 0/1 transition at "Execute".

Target system	Library/folder
MLC 15VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 15-6: Library assignment

Interface description

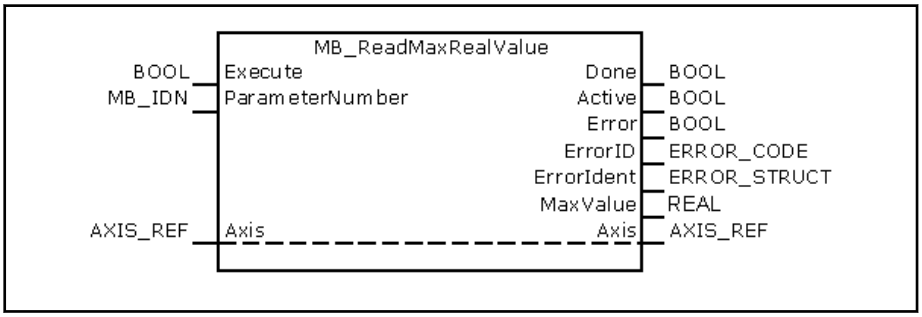


Fig. 15-8: MB_ReadMaxRealValue FB

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	The input variables (0/1 transition) are registered with a positive edge at "Execute". Subsequently, reading runs irrespective of the variable value.
	Parameter number	MB_IDN	Number of the parameter (see chapter ML_Base.library in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library R911341491)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information ("Troubleshooting" on page 360)
	MaxValue	Real	If "Done" = TRUE, "MaxValue" indicates the maximum input value allowed of the parameter selected.
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 15-7: Interface of the MB_ReadMaxRealValue function block

Brief description

Signal-time diagram

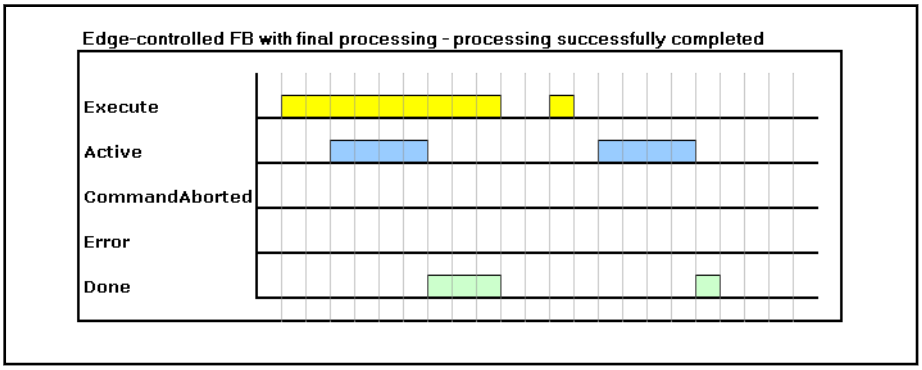


Fig. 15-9: Default signal-time diagram

☞ For the time required for the data transfer, refer to the chapter ML_PLCopen.library, Time requirement for the data transport using Read- and WriteParameter function blocks in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

☞ For the assignment

- of the axis input with A, C, M, N O parameters and/or
- of the axis group input with K-parameters

See chapter ML_PLCopen.library, Accessing data of an axis, control, kinematics (axis group) or another object in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

Troubleshooting The function block MB_ReadMaxRealValue generates the following types of error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- Chapter ML_PLCopen.library; Overview: Diagnostics; SERCOS_TABLE, ERROR_TABLE 16#0010 in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491) - Error when accessing S- or P-parameters (e.g. the parameter is not supported).
- Chapter ML_PLCopen.library; Overview: Diagnostics: MLC_TABLE, ERROR_TABLE 16#0030;” IndraLogic XLC IndraMotion MLC 14VRS; PLCopen Libraries” Library (R911341491) - Error when accessing the axis/the control.

15.7 MB_ReadMaxValue

Brief description The function block returns the maximum value allowed of a parameter specified by the user. The return value is represented as DINT number.

☞ Any change, e.g. of the parameter number or the axis at the input of the function block, requires another 0/1 transition at "Execute".

Target system	Library/folder
MLC 15VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 15-8: Library assignment

Interface description

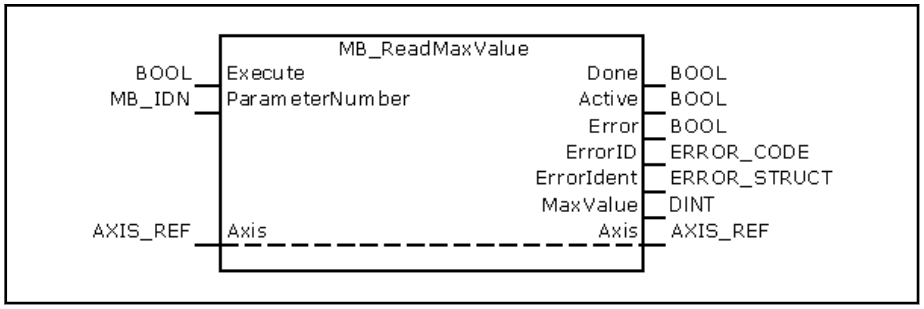


Fig. 15-10: MB_ReadMaxValue FB

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	The input variables (0/1 transition) are registered with a positive edge at "Execute". Subsequently, reading runs irrespective of the variable value
	Parameter number	MB_IDN	Number of the parameter, see chapter ML_PLCopen.library; Parameter access; Access data of an axis... in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information ("Troubleshooting" on page 362)
	MaxValue	DINT	If "Done" = TRUE, "MaxValue" indicates the maximum input value allowed of the parameter selected
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 15-9: Interface of the MB_ReadMaxValue FB

Brief description

Signal-time diagram

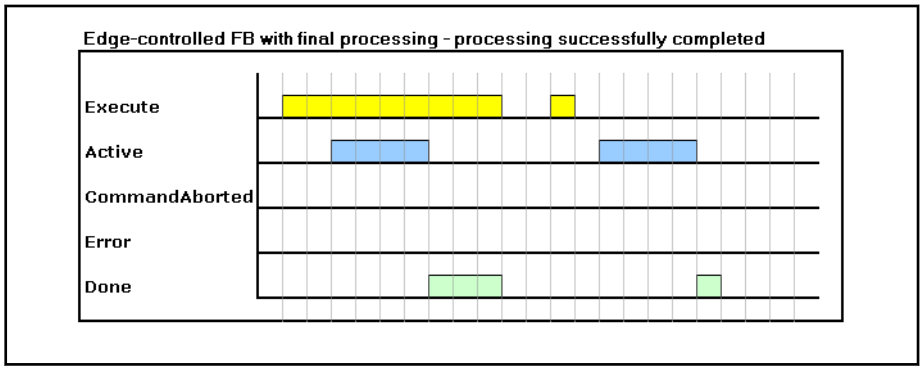


Fig. 15-11: Default signal-time diagram

For the time required for the data transfer, refer to the chapter ML_PLCopen.library, Time requirement for the data transport using Read- and WriteParameter function blocks in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

☞ For the assignment

- of the axis input with A, C, M, N O parameters and/or
- of the axis group input with K-parameters

See chapter ML_PLCopen.library, Accessing data of an axis, control, kinematics (axis group) or another object in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

Troubleshooting The function block MB_ReadMaxValue generates the following types of error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- Chapter ML_PLCopen.library; Overview: Diagnostics; SERCOS_TABLE, ERROR_TABLE 16#0010 in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491) - Error when accessing S- or P-parameters (e.g. the parameter is not supported).
- Chapter ML_PLCopen.library; Overview: Diagnostics: MLC_TABLE, ERROR_TABLE 16#0030;" IndraLogic XLC IndraMotion MLC 14VRS; PLCopen Libraries" Library (R911341491) - Error when accessing the axis/the control.

15.8 MB_ReadMinRealValue

Brief description The function block returns the minimum value allowed of a parameter specified by the user. The return value is represented as REAL number.

☞ Any change, e.g. of the parameter number or the axis at the input of the function block, requires another 0/1 transition at "Execute".

Target system	Library/folder
MLC 15VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 15-10: Library assignment

Interface description

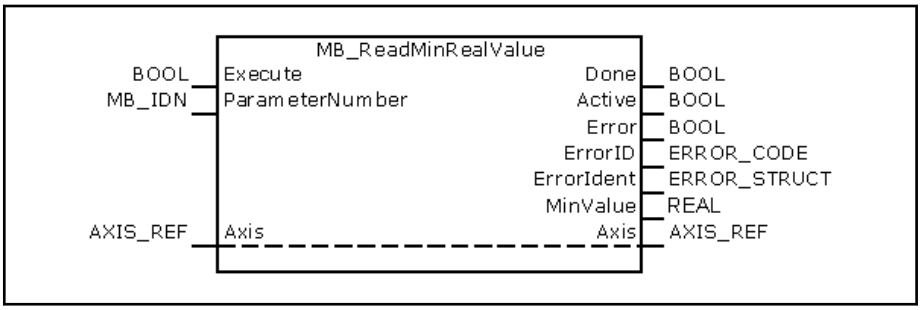


Fig. 15-12: MB_ReadMinRealValue FB

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	The input variables (0/1 transition) are registered with a positive edge at "Execute". Subsequently, reading runs irrespective of the variable value
	Parameter number	MB_IDN	Number of the parameter, see chapter ML_PLCopen.library; Parameter access; Access data of an axis... in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information ("Troubleshooting" on page 363)
	MinValue	REAL	If "Done" = TRUE, "MinValue" indicates the minimum input value allowed of the parameter selected.
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 15-11: Interface of the MB_ReadMinRealValue function block

Signal-time diagram

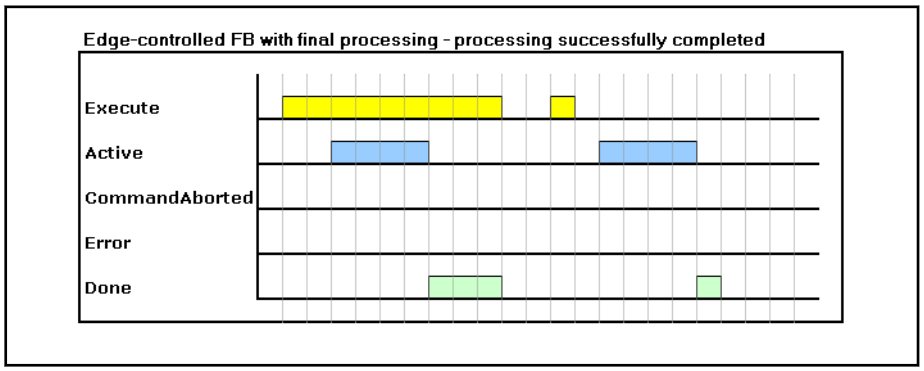


Fig. 15-13: Default signal-time diagram

For the time required for the data transfer, refer to the chapter ML_PLCopen.library, Time requirement for the data transport using Read- and WriteParameter function blocks in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

For the assignment

- of the axis input with A, C, M, N O parameters and/or
- of the axis group input with K-parameters

See chapter ML_PLCopen.library, Accessing data of an axis, control, kinematics (axis group) or another object in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

Troubleshooting

The function block MB_ReadMinRealValue generates the following types of error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- Chapter ML_PLCopen.library; Overview: Diagnostics; SERCOS_TABLE, ERROR_TABLE 16#0010 in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491) - Error when accessing S- or P-parameters (e.g. the parameter is not supported).
- Chapter ML_PLCopen.library; Overview: Diagnostics: MLC_TABLE, ERROR_TABLE 16#0030;" IndraLogic XLC IndraMotion MLC 14VRS; PLCopen Libraries" Library (R911341491) - Error when accessing the axis/the control.

15.9 MB_ReadMinValue

Brief description

The function block returns the minimum value allowed of a parameter specified by the user. The return value is represented as DINT number.

Any change, e.g. of the parameter number or the axis at the input of the function block, requires another 0/1 transition at "Execute".

Target system	Library/folder
MLC 15VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 15-12: Library assignment

Interface description

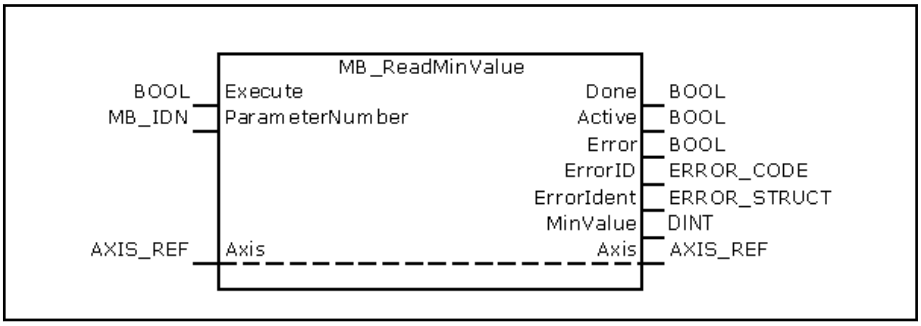


Fig. 15-14: MB_ReadMinValue FB

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	The input variables (0/1 transition) are registered with a positive edge at "Execute". Subsequently, reading runs irrespective of the variable value
	Parameter number	MB_IDN	Number of the parameter, see chapter ML_PLCopen.library; Parameter access; Access data of an axis... in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred when the function block instance was called

	Name	Type	Comment
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information ("Troubleshooting" on page 365)
	MinValue	DINT	If "Done" = TRUE, "MinValue" indicates the minimum input value allowed of the parameter selected.
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 15-13: Interface of the MB_ReadMinValue FB

Signal-time diagram

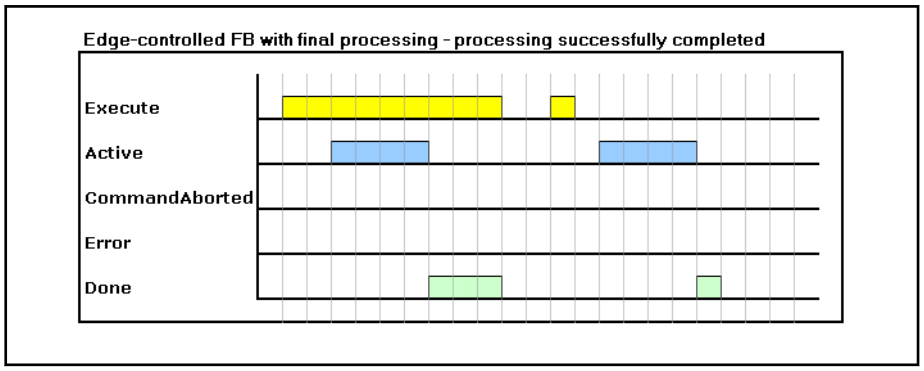


Fig. 15-15: Default signal-time diagram

For the time required for the data transfer, refer to the chapter ML_PLCopen.library, Time requirement for the data transport using Read- and WriteParameter function blocks in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

For the assignment

- of the axis input with A, C, M, N O parameters and/or
- of the axis group input with K-parameters

See chapter ML_PLCopen.library, Accessing data of an axis, control, kinematics (axis group) or another object in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

Troubleshooting

The function block MB_ReadMinValue generates the following types of error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- Chapter ML_PLCopen.library; Overview: Diagnostics; SERCOS_TABLE, ERROR_TABLE 16#0010 in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491) - Error when accessing S- or P-parameters (e.g. the parameter is not supported).
- Chapter ML_PLCopen.library; Overview: Diagnostics: MLC_TABLE, ERROR_TABLE 16#0030;" IndraLogic XLC IndraMotion MLC 14VRS; PLCopen Libraries" Library (R911341491) - Error when accessing the axis/the control.

15.10 MB_ReadName

Brief description

The function block returns the name of a parameter specified by the user. The return value is represented as a text.

Any change, e.g. of the parameter number or the axis at the input of the function block, requires another 0/1 transition at "Execute".

Target system	Library/folder
MLC 15VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 15-14: Library assignment

Interface description

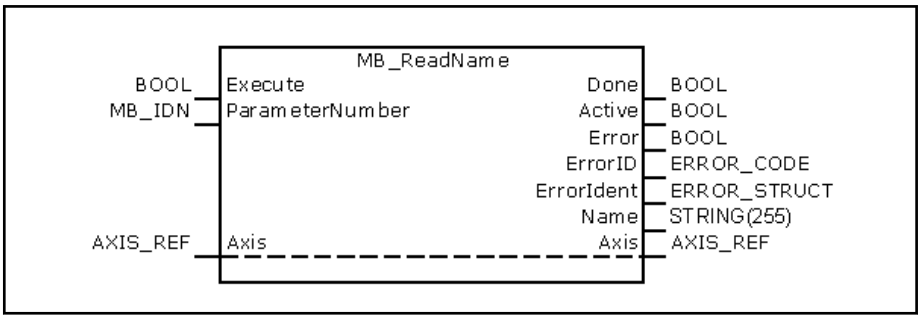


Fig. 15-16: MB_ReadName FB

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	The input variables (0/1 transition) are registered with a positive edge at "Execute". Subsequently, reading runs irrespective of the variable value
	Parameter number	MB_IDN	Number of the parameter, see chapter ML_PLCopen.library; Parameter access; Access data of an axis... in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information ("Troubleshooting" on page 367)
	Name	STRING(255)	If "Done" = TRUE, "MinValue" indicates the name of the parameter selected
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 15-15: Interface of the MB_ReadName function block

Signal-time diagram

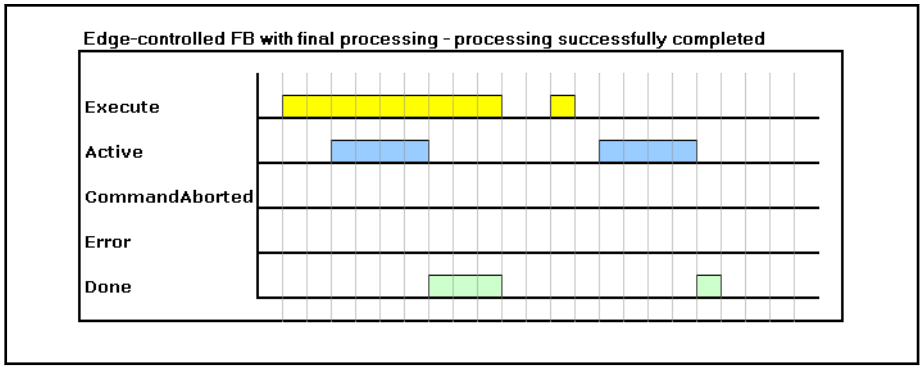


Fig. 15-17: Default signal-time diagram

For the assignment

- of the axis input with A, C, M, N O parameters and/or
- of the axis group input with K-parameters

See chapter ML_PLCopen.library, Accessing data of an axis, control, kinematics (axis group) or another object in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

Troubleshooting

The function block MB_ReadName generates the following types of error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- Chapter ML_PLCopen.library; Overview: Diagnostics; SERCOS_TABLE, ERROR_TABLE 16#0010 in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491) - Error when accessing S- or P-parameters (e.g. the parameter is not supported).
- Chapter ML_PLCopen.library; Overview: Diagnostics: MLC_TABLE, ERROR_TABLE 16#0030;" IndraLogic XLC IndraMotion MLC 14VRS; PLCopen Libraries" Library (R911341491) - Error when accessing the axis/the control.

15.11 MB_ReadParameter

Brief description

The function block returns the value of a user-specific parameter. The return value is represented as DINT number.

Any change, e.g. of the parameter number or the axis at the input of the function block, requires another 0/1 transition at "Enable".

Target system	Library/folder
MLC 15VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 15-16: Library assignment

Interface description

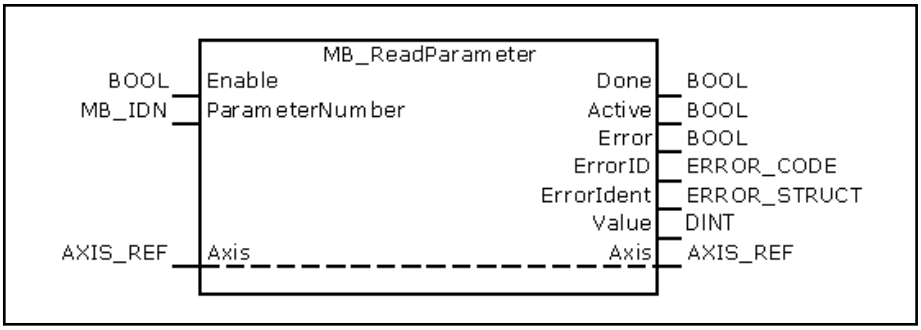


Fig. 15-18: MB_ReadParameter FB

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Continuous reading of the parameter as long as "Enable" is TRUE
	Parameter number	MB_IDN	Number of the parameter, see chapter ML_PLCopen.library; Parameter access; Access data of an axis... in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information ("Troubleshooting" on page 369)
	Value	DINT	4 byte value, represented as DINT
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the actual axis and/or the addressed object.

Tab. 15-17: Interface of the MB_ReadParameter FB

Signal-time diagram

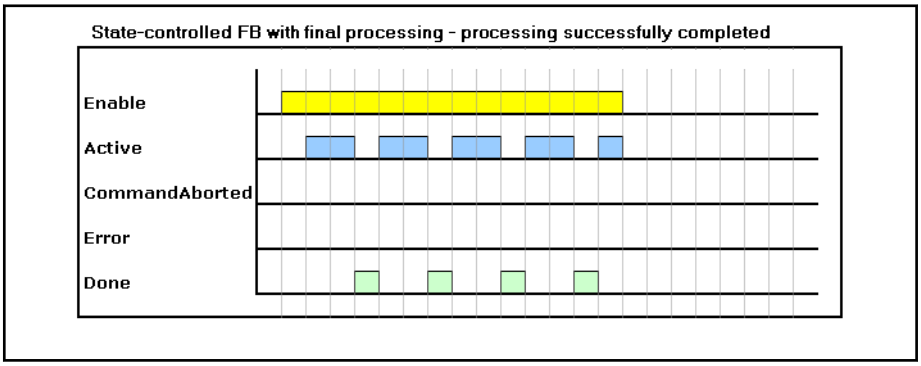


Fig. 15-19: Default signal-time diagram

For the time required for the data transfer, refer to the chapter ML_PLCopen.library, Time requirement for the data transport using Read- and WriteParameter function blocks in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

☞ For the assignment

- of the axis input with A, C, M, N O parameters and/or
- of the axis group input with K-parameters

See chapter ML_PLCopen.library, Accessing data of an axis, control, kinematics (axis group) or another object in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

Troubleshooting The function block MB_ReadParameter generates the following types of error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- Chapter ML_PLCopen.library; Overview: Diagnostics; SERCOS_TABLE, ERROR_TABLE 16#0010 in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491) - Error when accessing S- or P-parameters (e.g. the parameter is not supported).
- Chapter ML_PLCopen.library; Overview: Diagnostics: MLC_TABLE, ERROR_TABLE 16#0030;" IndraLogic XLC IndraMotion MLC 14VRS; PLCopen Libraries" Library (R911341491) - Error when accessing the axis/the control.

15.12 MB_ReadRealParameter

Brief description The function block returns the value of a user-specific parameter. The return value is represented as REAL number.

☞ Any change, e.g. of the parameter number or the axis at the input of the function block, requires another 0/1 transition at "Enable".

Target system	Library/folder
MLC 15VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 15-18: Library assignment

Interface description

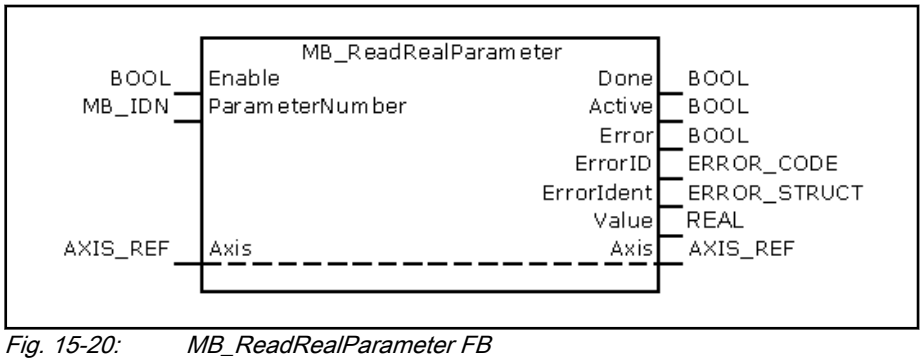


Fig. 15-20: MB_ReadRealParameter FB

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Continuous reading of the parameter as long as "Enable" is TRUE
	Parameter number	MB_IDN	Number of the parameter, see chapter ML_PLCopen.library; Parameter access; Access data of an axis... in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information ("Troubleshooting" on page 370)
	Value	REAL	4 byte value represented as REAL number
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it

Tab. 15-19: Interface of the MB_ReadRealParameter FB

Signal-time diagram

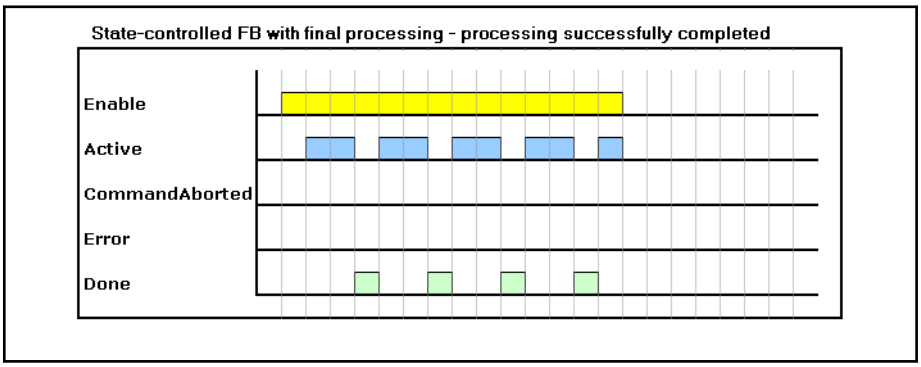


Fig. 15-21: Default signal-time diagram

For the time required for the data transfer, refer to the chapter ML_PLCopen.library, Time requirement for the data transport using Read- and WriteParameter function blocks in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

For the assignment

- of the axis input with A, C, M, N O parameters and/or
- of the axis group input with K-parameters

See chapter ML_PLCopen.library, Accessing data of an axis, control, kinematics (axis group) or another object in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

Troubleshooting

The function block MB_ReadRealParameter generates the following types of error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- Chapter ML_PLCopen.library; Overview: Diagnostics; SERCOS_TABLE, ERROR_TABLE 16#0010 in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491) - Error when accessing S- or P-parameters (e.g. the parameter is not supported).
- Chapter ML_PLCopen.library; Overview: Diagnostics: MLC_TABLE, ERROR_TABLE 16#0030;" IndraLogic XLC IndraMotion MLC 14VRS; PLCopen Libraries" Library (R911341491) - Error when accessing the axis/the control.

15.13 MB_ReadSercosAttribute

Brief description

The function block allows for reading the Sercos attributes of a parameter. The return value is represented as a DWORD.



Any change, e.g. of the parameter number or the axis at the input of the function block, requires another 0/1 transition at "Execute".

Target system	Library/folder
MLC 15VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 15-20: Library assignment

Interface description

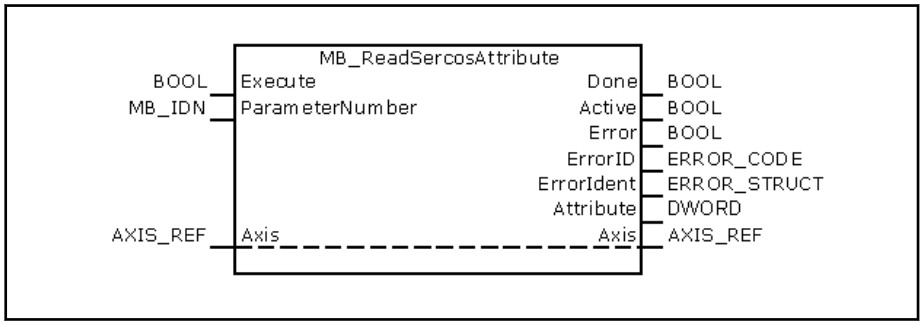


Fig. 15-22: MB_ReadSercosAttribute FB

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Reading the parameter attribute at transition 0-1
	Parameter number	MB_IDN	Number of the parameter, see chapter ML_PLCopen.library; Parameter access; Access data of an axis... in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information ("Troubleshooting" on page 372)

	Name	Type	Comment
	Attributes	DWORD	If "Done"=TRUE, "Attribute" indicates the attribute of the selected parameter
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 15-21: Interface of the MB_ReadSercosAttribute function block

Decoding the attribute

The IL_SercosAttribute function block is provided in the "ML_Base" library for decoding purposes.

Signal-time diagram

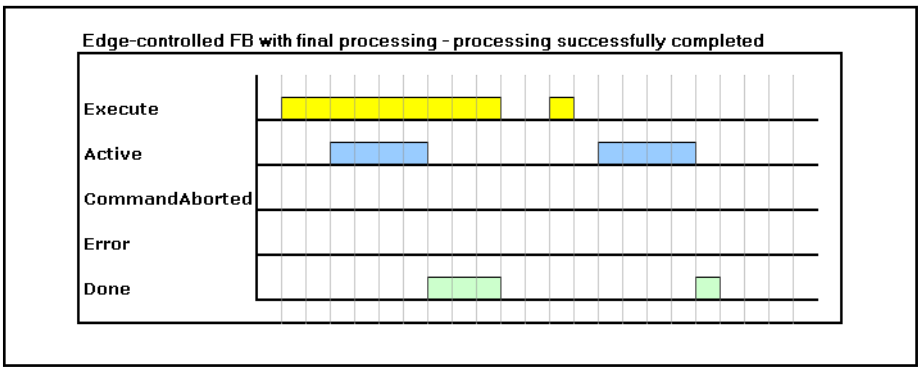


Fig. 15-23: Default signal-time diagram

For the assignment

- of the axis input with A, C, M, N O parameters and/or
- of the axis group input with K-parameters

See chapter ML_PLCopen.library, Accessing data of an axis, control, kinematics (axis group) or another object in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

Troubleshooting

The function block MB_ReadSercosAttribute generates the following types of error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- Chapter ML_PLCopen.library; Overview: Diagnostics; SERCOS_TABLE, ERROR_TABLE 16#0010 in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491) - Error when accessing S- or P-parameters (e.g. the parameter is not supported).
- Chapter ML_PLCopen.library; Overview: Diagnostics: MLC_TABLE, ERROR_TABLE 16#0030;" IndraLogic XLC IndraMotion MLC 14VRS; PLCopen Libraries" Library (R911341491) - Error when accessing the axis/the control.

15.14 MB_ReadSercosDataStatus

Brief description

The function block returns the status of a parameter specified by the user. The return value is represented as an ENUM value (INT number).

Any change, e.g. of the parameter number or the axis at the input of the function block, requires another 0/1 transition at "Enable".

Target system	Library/folder
MLC 15VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 15-22: Library assignment

Interface description

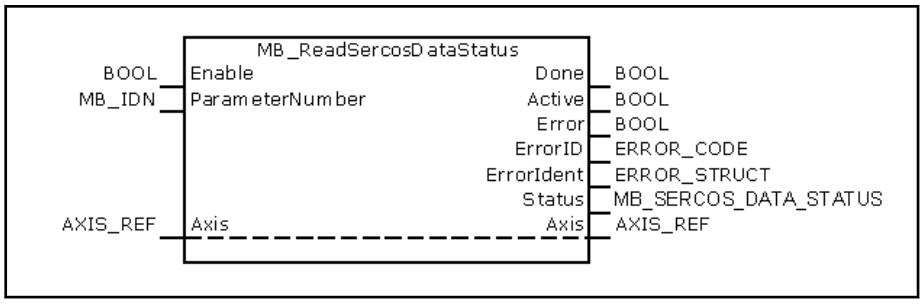


Fig. 15-24: FB MB_ReadSercosDataStatus

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Continuous reading of the parameter as long as "Enable" is TRUE
	Parameter number	MB_IDN	Number of the parameter, see chapter ML_PLCopen.library; Parameter access; Access data of an axis... in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information ("Troubleshooting" on page 374)
	Status	MB_SERCOS_DATA_STATUS	Status of the selected parameter chapter "MB_SERCOS_DATA_STATUS, Structure contains all Sercos command states" on page 350
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 15-23: Interface of the MB_ReadSercosDataStatus function block

Brief description

Using this function block and a MB_WriteParameter, page 378,, drive commands can be triggered and their processing tracked:

- 0 - Command not set and not released
- 1 - Command set and released
- 3 - Command processed correctly
- 5 - Command processing interrupted
- 7 - Command processing running
- 15 - Error, command processing impossible
- 256 - Value of the parameter beyond its minimum/maximum range

Signal-time diagram

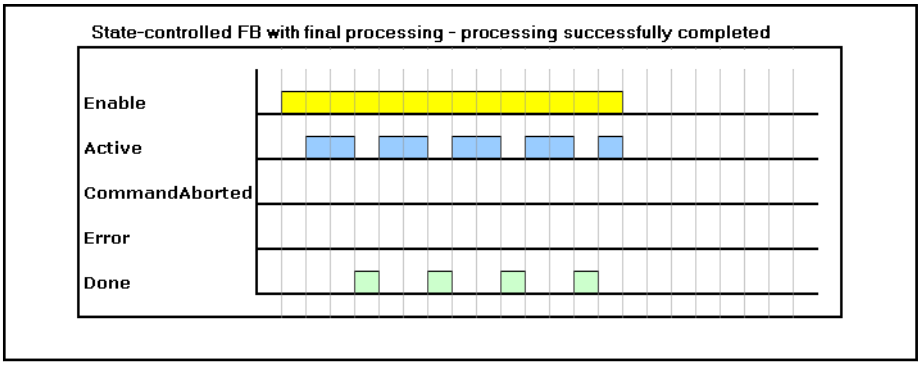


Fig. 15-25: Default signal-time diagram

For the assignment

- of the axis input with A, C, M, N O parameters and/or
- of the axis group input with K-parameters

See chapter ML_PLCopen.library, Accessing data of an axis, control, kinematics (axis group) or another object in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

Troubleshooting

The function block MB_ReadSercosDataStatus generates the following types of error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- Chapter ML_PLCopen.library; Overview: Diagnostics; SERCOS_TABLE, ERROR_TABLE 16#0010 in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491) - Error when accessing S- or P-parameters (e.g. the parameter is not supported).
- Chapter ML_PLCopen.library; Overview: Diagnostics: MLC_TABLE, ERROR_TABLE 16#0030;" IndraLogic XLC IndraMotion MLC 14VRS; PLCopen Libraries" Library (R911341491) - Error when accessing the axis/the control.

15.15 MB_ReadStringParameter

Brief description

The function block returns the value of a user-specific parameter. The return value is displayed as a character string (STRING) according to with its actual length with a maximum of 255 characters.

Assignment: Target system/library

Target system	Library/folder
MLC 15VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 15-24: Library assignment

Interface description

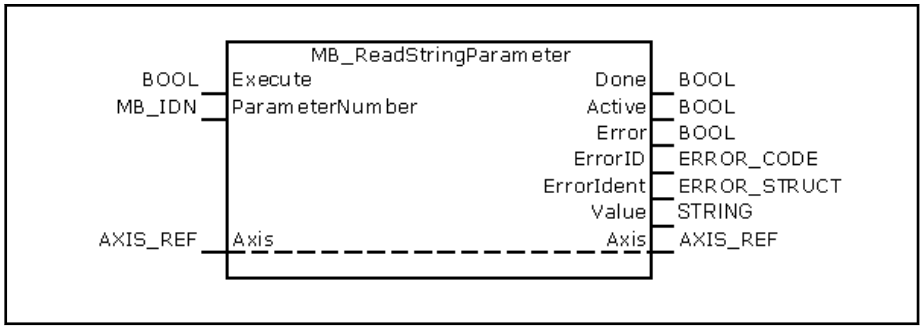


Fig. 15-26: MB_ReadStringParameter FB

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	It starts reading the text of the parameter with a rising edge
	Parameter number	MB_IDN	Number of the parameter, see chapter ML_PLCopen.library; Parameter access; Access data of an axis... in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information ("Troubleshooting" on page 376)
	Value	STRING	Character string with up to 255 characters
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 15-25: Interface of the MB_ReadStringParameter FB

Signal-time diagram

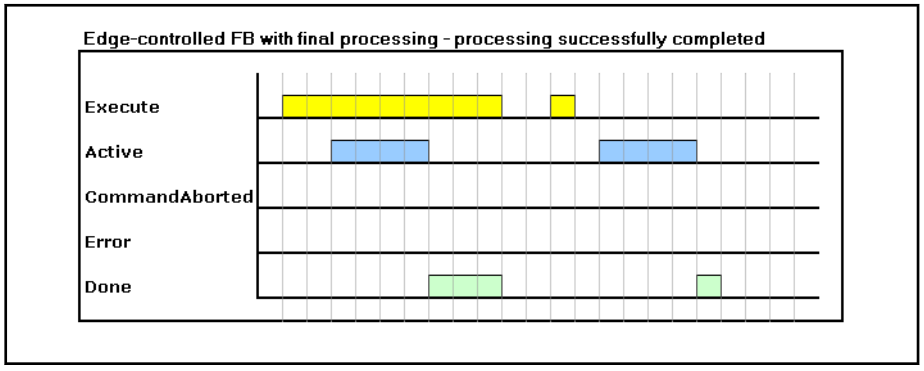


Fig. 15-27: Default signal-time diagram

For the time required for the data transfer, refer to the chapter ML_PLCopen.library, Time requirement for the data transport using Read- and WriteParameter function blocks in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

☞ For the assignment

- of the axis input with A, C, M, N O parameters and/or
- of the axis group input with K-parameters

See chapter ML_PLCopen.library, Accessing data of an axis, control, kinematics (axis group) or another object in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

Troubleshooting The function block MB_ReadStringParameter generates the following types of error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- Chapter ML_PLCopen.library; Overview: Diagnostics; SERCOS_TABLE, ERROR_TABLE 16#0010 in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491) - Error when accessing S- or P-parameters (e.g. the parameter is not supported).
- Chapter ML_PLCopen.library; Overview: Diagnostics: MLC_TABLE, ERROR_TABLE 16#0030;" IndraLogic XLC IndraMotion MLC 14VRS; PLCopen Libraries" Library (R911341491) - Error when accessing the axis/the control.

15.16 MB_ReadUnit

Brief description The function block returns the measuring unit of a parameter specified by the user. The return value is represented as a text.

☞ Any change, e.g. of the parameter number or the axis at the input of the function block, requires another 0/1 transition at "Execute".

Target system	Library/folder
MLC 15VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 15-26: Library assignment

Interface description

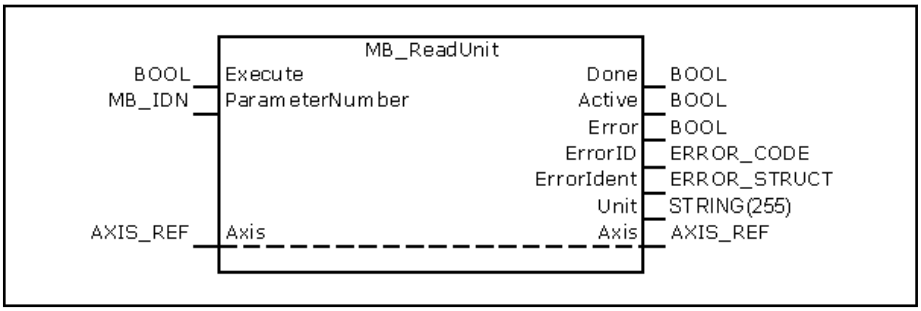


Fig. 15-28: MB_ReadUnit FB

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	The input variables (0/1 transition) are registered with a positive edge at "Execute". Subsequently, reading runs irrespective of the variable value.
	Parameter number	MB_IDN	Number of the parameter, see chapter ML_PLCopen.library; Parameter access; Access data of an axis... in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Unit	STRING[255]	If "Done" = TRUE, "Unit" indicates the measuring unit of the parameter selected.
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 15-27: Interface of the MB_ReadName function block

Signal-time diagram

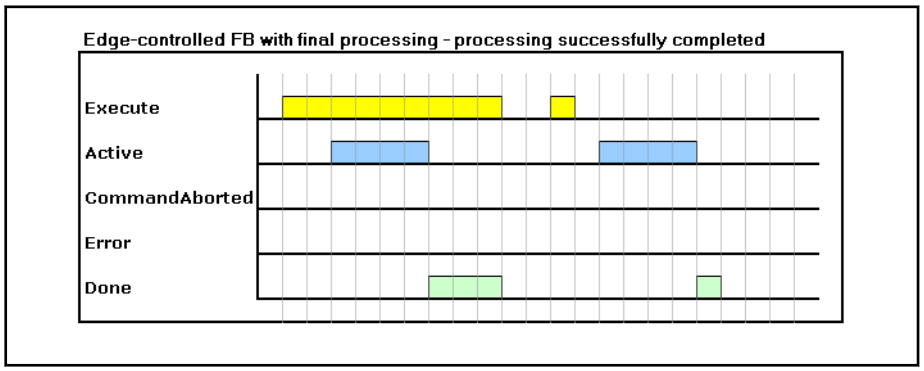


Fig. 15-29: Default signal-time diagram

For the assignment

- of the axis input with A, C, M, N O parameters and/or
- of the axis group input with K-parameters

See chapter ML_PLCopen.library, Accessing data of an axis, control, kinematics (axis group) or another object in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

Troubleshooting

The function block MB_ReadUnit generates the following types of error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- Chapter ML_PLCopen.library; Overview: Diagnostics; SERCOS_TABLE, ERROR_TABLE 16#0010 in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491) - Error when accessing S- or P-parameters (e.g. the parameter is not supported).
- Chapter ML_PLCopen.library; Overview: Diagnostics: MLC_TABLE, ERROR_TABLE 16#0030;" IndraLogic XLC IndraMotion MLC 14VRS; PLCopen Libraries" Library (R911341491) - Error when accessing the axis/the control.

15.17 MB_WriteParameter

Brief description

The function block writes a value in a parameter specified by the user.

Target system	Library/folder
MLC 15VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 15-28: Library assignment

Interface description

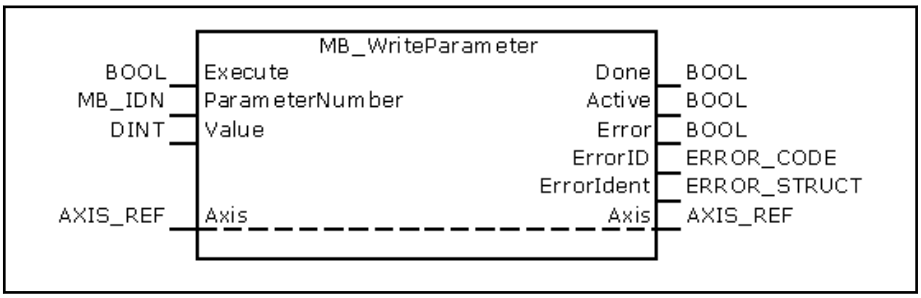


Fig. 15-30: MB_WriteParameter FB

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	It starts writing the value that occurs at the rising edge to the parameter
	Parameter number	MB_IDN	Number of the parameter, see chapter ML_PLCopen.library; Parameter access; Access data of an axis... in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)
	Value	DINT	Value to be written to the parameter
VAR_OUTPUT	Done	BOOL	The value has been written successfully to the control/drive
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information ("Troubleshooting" on page 379)
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it

Tab. 15-29: Interface of the MB_WriteParameter FB

Signal-time diagram

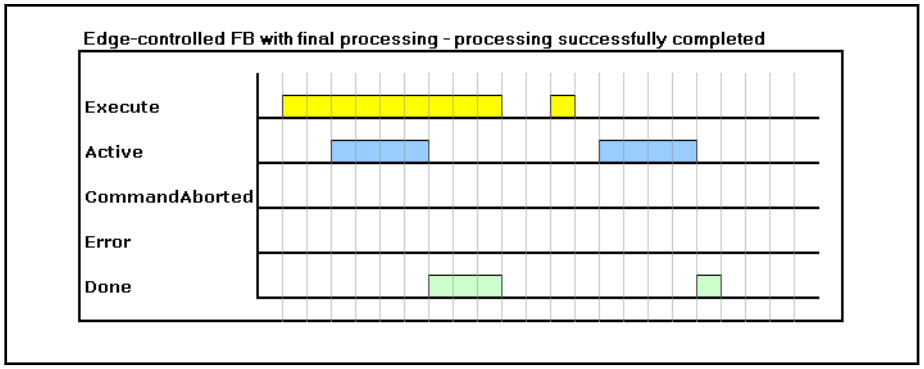


Fig. 15-31: Default signal-time diagram

For the time required for the data transfer, refer to the chapter ML_PLCopen.library, Time requirement for the data transport using Read- and WriteParameter function blocks in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

For the assignment

- of the axis input with A, C, M, N O parameters and/or
- of the axis group input with K-parameters

See chapter ML_PLCopen.library, Accessing data of an axis, control, kinematics (axis group) or another object in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

Troubleshooting

The function block MB_WriteParameter generates the following types of error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- Chapter ML_PLCopen.library; Overview: Diagnostics; SERCOS_TABLE, ERROR_TABLE 16#0010 in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491) - Error when accessing S- or P-parameters (e.g. the parameter is not supported).
- Chapter ML_PLCopen.library; Overview: Diagnostics: MLC_TABLE, ERROR_TABLE 16#0030;" IndraLogic XLC IndraMotion MLC 14VRS; PLCopen Libraries" Library (R911341491) - Error when accessing the axis/the control.

15.18 MB_WriteRealParameter

Brief description

This function block writes a value to a user-specific parameter.

Target system	Library/folder
MLC 15VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 15-30: Library assignment

Interface description

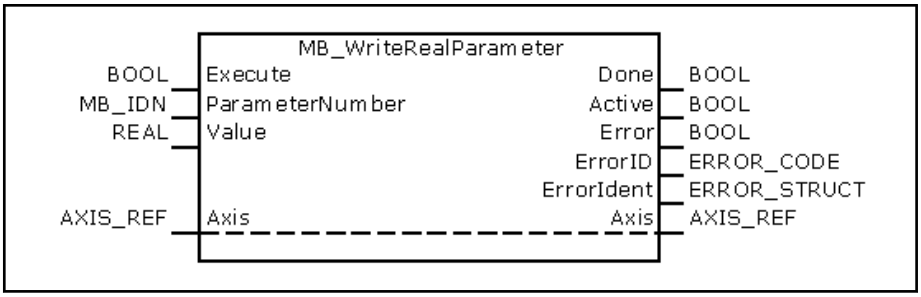


Fig. 15-32: MB_WriteRealParameter FB

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	It starts writing the value that occurs at the rising edge to the parameter
	Parameter number	MB_IDN	Number of the parameter, see chapter ML_PLCopen.library; Parameter access; Access data of an axis... in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)
	Value	REAL	Value to be written to the parameter
VAR_OUTPUT	Done	BOOL	The value has been written successfully to the control/drive
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information ("Troubleshooting" on page 381)
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it

Tab. 15-31: Interface of the MB_WriteRealParameter FB

Signal-time diagram

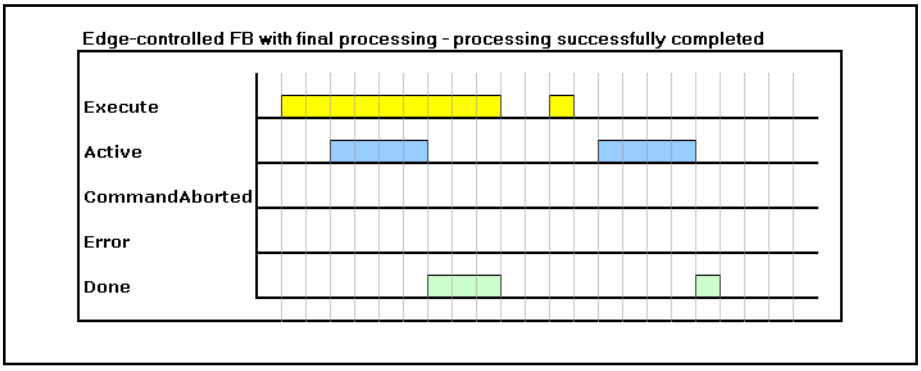


Fig. 15-33: Default signal-time diagram

For the time required for the data transfer, refer to the chapter ML_PLCopen.library, Time requirement for the data transport using Read- and WriteParameter function blocks in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

☞ For the assignment

- of the axis input with A, C, M, N O parameters and/or
- of the axis group input with K-parameters

See chapter ML_PLCopen.library, Accessing data of an axis, control, kinematics (axis group) or another object in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

Troubleshooting The MB_WriteRealParameter FB generates the following types of error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- Chapter ML_PLCopen.library; Overview: Diagnostics; SERCOS_TABLE, ERROR_TABLE 16#0010 in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491) - Error when accessing S- or P-parameters (e.g. the parameter is not supported).
- Chapter ML_PLCopen.library; Overview: Diagnostics: MLC_TABLE, ERROR_TABLE 16#0030;" IndraLogic XLC IndraMotion MLC 14VRS; PLCopen Libraries" Library (R911341491) - Error when accessing the axis/the control.

15.19 MB_WriteStringParameter

Brief description The function block writes a character string with a maximum of 255 characters to a user-specific parameter.

Assignment: Target system/library

Target system	Library/folder
MLC 15VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 15-32: Library assignment

Interface description

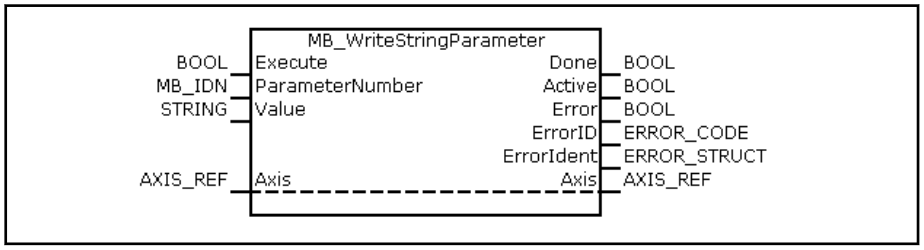


Fig. 15-34: MB_WriteStringParameter FB

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	It starts writing the value that occurs at the rising edge to the parameter
	Parameter number	MB_IDN	Number of the parameter, see chapter ML_PLCopen.library; Parameter access; Access data of an axis... in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491)
	Value	STRING[255]	Value to be written to the parameter

	Name	Type	Comment
VAR_OUTPUT	Done	BOOL	The value has been written successfully to the control/drive
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information ("Troubleshooting" on page 382)
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it

Tab. 15-33: Interface of the MB_WriteStringParameter FB

Signal-time diagram

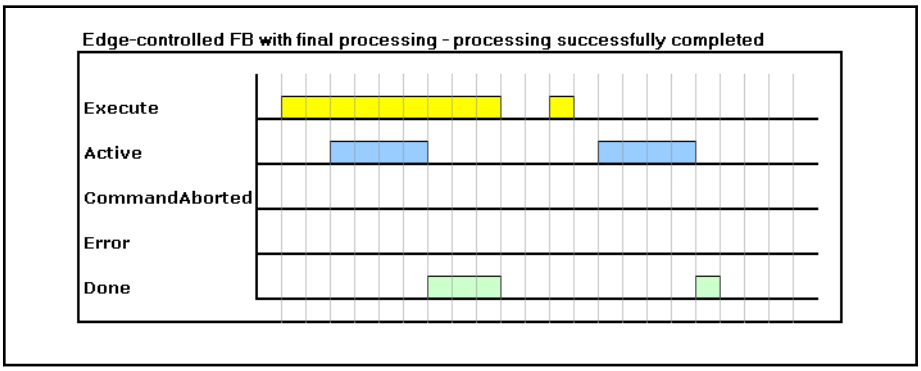


Fig. 15-35: Default signal-time diagram

For the time required for the data transfer, refer to the chapter ML_PLCopen.library, Time requirement for the data transport using Read- and WriteParameter function blocks in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

For the assignment

- of the axis input with A, C, M, N O parameters and/or
- of the axis group input with K-parameters

See chapter ML_PLCopen.library, Accessing data of an axis, control, kinematics (axis group) or another object in the documentation "IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries" Library (R911341491).

Troubleshooting

The function block MB_WriteStringParameter generates the following types of error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- Chapter SERCOS_TABLE, ERROR_TABLE 16#0010: IndraLogic XLC / IndraMotion MLC 14VRS; PLCopen Libraries, Library (R911341491) - Error when accessing S- or P-parameters (e.g. the parameter is not supported).

- Chapter MLC_TABLE, ERROR_TABLE 16#0030;” IndraLogic XLC IndraMotion MLC 14VRS; PLCopen Libraries” Library (R911341491) - Error when accessing the axis/the control.

16 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

0 … 9

3 iSp_DrvChangeClass3..... 115

A

About this documentation..... 1
Information representation..... 5
Acknowledging the crash state from NC to PLC..... 34
Acknowledging the crash state from PLC to NC (IPO)..... 36
Activate MZA messages..... 221
Active..... 227
ANSI Z535.6-2006..... 6
Automatic Sercos III I/O exchange (MT_S3AutoAdapt)
Configuration example..... 201
Function block parameters..... 201
General information..... 201
Status messages..... 201
Auxiliary functions..... 117
Axes..... 273
Overview..... 273
Axis-related input signals (PLC --> NC)
Axis in front of end point iAx_DistCtrl..... 85
Axis in position iAx_InPos..... 84
Axis on grid position iAx_NotchPos..... 84
Axis override 0% iAx_Override0..... 87
Axis override 100% iAx_Override100..... 88
Axis reset iAx_Reset..... 85
Axis running iAx_Run..... 84
Axis velocity reached iAx_ProgSpReach..... 85
Change in state class 2 iAx_DrvChangeClass2..... 90
Change in state class 3 iAx_DrvChangeClass3..... 91
Compensable Gantry command value offset exceeded iAx_CmdOffsExceed..... 89
Customer output 1 ... 8 iAx_Custom1 ... 8..... 88
Delayed error reaction iAx_DelayErrClass1..... 89
Drive in operation iAx_DrvAct..... 92
Drive ready iAx_DrvReady..... 92
Enabled for power activation iAx_DrvPower..... 92
Error of state class 1 iAx_DrvErrClass1..... 89
Fixed stop active iAx_FxStopAct..... 85
Fixed stop reached iAx_FxStopReached..... 85
Gantry command value offset active iAx_CmdOffsExst..... 89
Known reference point iAx_RefKnown..... 82
Lagging error iAx_CoupleLag..... 89
Negative traversing direction iAx_TrvDirNeg..... 84
Position switching point 1 ... 8 iAx_PosSwitch1 ... 8..... 86
Reference point reached iAx_RefReached..... 83
SCS signal status 0 ...15 iAx_ScsState00 ... 15..... 88
Standstill error iAx_TrqExceed..... 89
Status of safe position iAx_SafStatePos..... 86
Status output of controller iAx_SafCtrlOutputState..... 86
Test mode (dry run) iAx_DryRun..... 92
Torque reduced iAx_TrqLim..... 91
Travel command iAx_TrvCmd..... 83
Axis-related interface signals..... 67
Overview on axis-related interface signals..... 67
PLC input signals - Overview..... 26
PLC output signals - overview..... 24
Axis-related output signals (PLC --> NC)
Apply actual value offset qAx_TakeActOffs..... 80
Axis operation mode bit 0 and 1 qAx_OpModeSel_00 ... 01..... 71
Axis override 100% qAx_Override100..... 78
Axis reset qAx_Reset..... 73
Customer input 1 ... 8 qAx_Custom1 ... 8..... 78
Discharge axis qAx_Discharge..... 80
Drive inhibit qAx_SafDrvLock..... 74
Drive on qAx_DrvOn..... 81
Enabling button qAx_SafEnablCtrl..... 75
Feed inhibit qAx_DrvLock..... 81
Freeze target position qAx_FrzIpoPos..... 80
Gantry on master position qAx_MasterPos..... 79
General information..... 70
Handwheel direction of rotation qAx_HandwDir..... 76
Handwheel selection bit 0 and bit 1 qAx_HandwSel_00 ... 01..... 76
Hide coupling error qAx_LagErrOff..... 79
Hide end limit switch qAx_SwLimOff..... 74
Hide standstill error qAx_TrqErrOff..... 78
Incremental step as diameter qAx_JogDia..... 73
Incremental step in inch qAx_JogInch..... 73
Limit switch range 0 ... 1 qAx_TrvLim_00 ... 01..... 74
Manual feed bit 0 ... bit 3 qAx_ManFeed_00 ... 03..... 77
Manual+ qAx_JogPlus, Manual- qAx_JogMinus..... 71
Next grid position qAx_NextNotch..... 73
Operation mode selection qAx_SafOpModeSwitch..... 74
Override, bit 0 ... ... bit 15 qAx_Override_00 ... 15..... 78
Reduce delayed error reaction qAx_RedDelayErrClass1..... 80
reduce transmission time qAx_SafRedTransTime..... 76
Reference switch qAx_HomeSwitch..... 78
Release fixed stop qAx_FxStopRel..... 74

Si-switch 1 (S2) qAx_SafSwitch1..... 75
Si-switch 2 (S2) qAx_SafSwitch2..... 75
Status of safety signals qAx_SafTechState... 75
Torque reduction qAx_TrqLi..... 80
Axis-related output signals (PLC--> NC)
Check input : safety status qAx_SafCheckInputState..... 75
AxisNo..... 137

B
Backup database table (MT_DbSave)
Configuration example..... 170
Function..... 168
Status messages..... 171
BCE code..... 119
Binary-coded active channel states..... 58
Bit-coded auxiliary functions..... 118

C
Calculates the lapsed time..... 339
Calculates the path distance from the specified position data..... 336
Chan..... 139
Chanel-related interface signals
Input signals (NC -> PLC)..... 53
Change data blocks..... 162
Change feed for an asynchronous axis..... 301
Change record..... 1
Change the channel feed..... 254
Change tool data..... 152
Changing and determining the active jog distance..... 289
Changing and determining the active jog velocity..... 317
Changing the feed for a spindle..... 312
Changing the handwheel..... 278
Changing the JodMode..... 285
Changing the NC main operation modes..... 259, 268
Changing the NC operation modes..... 263
Changing the operation mode of an asynchronous axis..... 295
Channel restart (MT_ChanRestart)
Configuration example..... 135
Function..... 135
Status messages..... 136
Channel structure..... 37
Channel-related input signals (PLC --> NC)
Channel number bit 0 ... bit 5 iAx_ChIndex_00 ... 05..... 87
Index master axis bit 0 ... bit 7 iAx_MasterAxindex_00 ... 07..... 89
Si operation mode bit 0 ... bit 3 iAx_SafOpMode_00 ... 03..... 85
Channel-related interface signals..... 37
Overview on channel-related signals..... 37
PLC channel-related output signals..... 41
PLC channel-related output signals - overview..... 20, 37
PLC input signals - overview..... 22, 39
Signal description..... 41
Channel-related output signals (PLC --> NC)
Asynchronous subroutine 1 ... 8 qCh_ASub1 ... 8..... 47
Automatic reselection off qCh_ReSelOff..... 44
Automatic restart qCh_Restart..... 43
Block transfer inhibit qCh_TransferLock..... 43
Cancel distance to go qCh_CancDist..... 45
Conditional jump qCh_OptJump..... 48
Control reset qCh_CtrlReset..... 44
Coupling off qCh_CoordCoupleOff..... 51
Customer input 1 ... 8 qCh_Custom1 ... 8..... 50
Delete syntax errors qCh_CancSyntaxError..... 45
Direction of online correction..... 50
Fast retract qCh_Retract..... 47
Feed hold qCh_FeedHold..... 44
Feed inhibit qCh_FeedStop..... 44
Handwheel direction of rotation qCh_HandwDirWcs..... 48
Handwheel selection bit 0 qCh_HandwSelWcs_00 and handwheel selection bit 1 qCh_HandwSelWcs_01..... 47
Machine function qCh_MachineFunction..... 44
NC start qCh_NCStart..... 43
Online correction enabled..... 50
Operation mode of PLC qCh_OpModePlc..... 43
Operation mode selection qCh_OpModeSel_00 ... 03..... 41
Optional stop qCh_OptStop..... 48
Override 100% qCh_Override100..... 49
Override, bit 0 ... ... bit 15 qCh_Override_00 ... ... 15..... 49
Path motion backward qCh_RetraceMovRev..... 51
Path motion forward qCh_RetraceMovFwd... 50
Position handwheel qAx_HandwPosMode.... 77
Position handwheel qCh_HandwPosMode.... 48
Raid traverse override, bit 0 ... ... bit 15 qCh_Override_00 ... ... 15..... 51
Rapid traverse 100% qCh_Rapid100..... 49
Reduced rapid traverse qCh_RedRap..... 49
Retrace mode qCh_Retrace..... 50
Return to contour qCh_RetCont..... 46
Second feed qCh_Feedrate2..... 52
Skip block qCh_BlkSlash..... 48
Test feed qCh_TestFeed..... 51
Test rapid traverse qCh_TestRap..... 52
TTL released qCh_TangTRotRel..... 51
WCS Manual + qCh_JogPlusWcs and WCS Manual - qCh_JogMinusWcs..... 47
Channel-related output signals (PLC--> NC)
Switch to next block qCh_NextBlk..... 46
Checking channel number..... 240
Clearing Sercos I/O device errors..... 32

CommandAborted..... 227
Communication structure..... 13
Complaints..... 7
CondMask..... 156
Configurations..... 15
Configure PLC-NC bit interface..... 15
Configure machine state display (MZA)
Configure MZA bit interface..... 219
Enable messages..... 221
General information..... 219
MZA file structure..... 220
Parameterize machine state display (MZA)..... 219
Supplementing notes..... 221
Configure PLC-specific data in IndraWorks..... 219
Configure machine status display (MZA)..... 219
Configuring PLC-NC bit interface
Axis interface..... 16
Channel interface..... 15
General information..... 15
Global interface (general interface)..... 15
Spindle interface..... 17
Connect digital I/Os..... 123
Controller enable and axis status..... 308
Controller enable and spindle status..... 327
Coordinate coupling..... 51
Coupling distance..... 89
CPL variable..... 179
Crash state from NC to PLC..... 36
Crash state from PLC to NC (IPO)..... 34
Criticism..... 7
Customer Feedback..... 7

D
Data types..... 343, 350
Delete data blocks..... 156
Delete error information..... 32
Delete tool data blocks..... 148
Determine active operation mode..... 256
Determine current channel status..... 242
Determine drive status..... 302
Determine the drive status..... 320
Determining active edge of the active tool..... 335
Determining active feed override..... 255
Determining Active JogMode..... 283
Determining active NC program..... 244
Determining current program progress..... 246
Determining the active jog velocity..... 313
Determining the active Safety Engineering operating state..... 274
Determining the operation mode of an asynchronous axis..... 294
Digital I/Os..... 123
Documentation
Revision history..... 1
Done..... 227
DP master interface..... 123
Drive status..... 346
Drive status incl. Determine errors and warnings..... 303, 322
Drive switch-off with lock..... 113

E
Edge-controlled signals..... 70
Edit data block lists (MT_DbRecList)
Configuration example..... 159
Function..... 156
Status messages..... 161
Edit data blocks (MT_DbData)
Configuration example..... 166
Function..... 162
Status messages..... 167
Edit data blocks (MT_P_DbData)
Configuration example..... 153
Function..... 152
Status messages..... 155
Edit tool lists (MT_P_DbRecList)
Configuration example..... 150
Function..... 147
Status messages..... 151
ElemPath..... 162
Enable..... 227
End block..... 129
Error code..... 125
Execute..... 227
ExecuteLock..... 227
Extended NC block specification (MT_NcBlkExt)
Configuration example..... 134
Function..... 132
Status messages..... 134

F
Feedback..... 7
Function block types..... 225
Function block types and their interfaces..... 226

G
Gear in idle position..... 111
Gear range selection..... 110
Gear switching function..... 110
General Interface..... 15
Generates an integer random number..... 341
Global interface..... 15
Global interface signals..... 31
Overview on global interface signals..... 31
PLC input signals..... 35
PLC input signals - Overview..... 19
PLC output signals..... 32
PLC output signals - Overview..... 19
Signal description..... 32

H
Handwheel data (MT_Handw)
Configuration example..... 141

Function..... 140
Status messages..... 141
Hardware warning..... 35
Hazard warnings..... 5
Helpdesk..... 385
Hotline..... 385

I
iAx_ChIndex_00 ... 05..... 87
iAx_CmdOffsExceed..... 89
iAx_CmdOffsExst..... 89
iAx_CoupleLag..... 89
iAx_Custom1 ... 8..... 88
iAx_DelayErrClass1..... 89
iAx_DistCtrl..... 85
iAx_DrvAct..... 92
iAx_DrvChangeClass2..... 90
iAx_DrvChangeClass3..... 91
iAx_DrvErrClass1..... 89
iAx_DrvPower..... 92
iAx_DrvReady..... 92
iAx_DryRun..... 92
iAx_FxStopAct..... 85
iAx_FxStopReached..... 85
iAx_InPos..... 84
iAx_MasterAxindex_00 ... 07..... 89
iAx_NotchPos..... 84
iAx_Override0..... 87
iAx_Override100..... 88
iAx_PosSwitch1 ... 8..... 86
iAx_ProgSpReach..... 85
iAx_RefKnown..... 82
iAx_RefReached..... 83
iAx_Reset..... 85
iAx_Run..... 84
iAx_SafCtrlOutputState..... 86
iAx_SafOpMode_00 ... 03..... 85
iAx_SafStatePos..... 86
iAx_ScsState00 ... 15..... 88
iAx_TrqExceed..... 89
iAx_TrqLim..... 91
iAx_TrvCmd..... 83
iAx_TrvDirNeg..... 84
iCh_ActFunc01..... 62
iCh_ActFunc02..... 62
iCh_ActFunc03..... 62
iCh_ActFunc04..... 62
iCh_ActFunc05..... 62
iCh_ActFunc06..... 62
iCh_ActFunc07..... 62
iCh_ActFunc08..... 63
iCh_ActFunc09 ... 13..... 63
iCh_ActFunc18..... 64
iCh_ASub1 ... 8..... 57
iCh_BlkSlash..... 58
iCh_CoordCoupleAct..... 61
iCh_Cpl01 ... 16..... 60
iCh_Custom1 ... 8..... 60
iCh_DryRun..... 53
iCh_FeedHoldAct..... 55
iCh_Feedrate2..... 64
iCh_G0Act..... 60
iCh_G41G141Act..... 61
iCh_G42G142Act..... 61
iCh_InPosAct..... 60
iCh_NCReady..... 54
iCh_OpMode_00 ... 03..... 53
iCh_OptJump..... 58
iCh_OptStop..... 58
iCh_Override0..... 60
iCh_Override100..... 60
iCh_ProgRun..... 54
iCh_ProgStopM0..... 55
iCh_ProgStopM30..... 55
iCh_RapOverride0..... 64
iCh_RapOverride100..... 64
iCh_ReadyReEnter..... 56
iCh_ReEnterAct..... 57
iCh_RemoveFinish..... 56
iCh_Reset..... 56
iCh_RetContPath..... 56
iCh_RetpaAct..... 56
iCh_RetraceAct..... 61
iCh_SRunAct..... 59
iCh_SRunReEnter..... 60
iCh_SRunRepos..... 60
iCh_State_00 .. 04..... 57
iCh_TangTRotCmd..... 61
iCh_TransferLockAct..... 55
Idling position of the gear..... 111
iGen_CrashAckn..... 36
iGen_CrashStoreData..... 36
iGen_HardwareState..... 35
iGen_NoStroke..... 35
iGen_Reset..... 35
iGen_SercosIO_Reset..... 35
iGen_StrokeIntend..... 35
IL_SIII_ATTRIBUTE,..... 350
Inch increments..... 73
Incremental steps in diameter..... 73
Information representation
Names and Abbreviations..... 7
InGear..... 226
InOperation..... 226
Input signals (NC --> PLC)
Activate optional stop iCh_OptStop..... 58
Activate skip block iCh_BlkSlash..... 58
Active operation mode iCh_OpMode_00 ... 03..... 53
Asynchronous subroutine 1 ... 8 iCh_ASub1 ... 8..... 57
Axis transformation 2 inactive iCh_ActFunc04..... 62
Block pre-run active iCh_SRunAct..... 59

Block transfer inhibit active iCh_Transfer-LockAct..... 55
Channel reset iCh_Reset..... 56
Channel state bit 0 ... 4 iCh_State_00 .. 04.... 57
Feed 100% active iCh_ActFunc02..... 62
Feed hold active iCh_FeedHoldAct..... 55
G41/141 active iCh_G41G141Act, G42/142 active iCh_G42G142Act..... 61
G70 active iCh_ActFunc01..... 62
G95 active iCh_ActFunc03..... 62
G96 active iCh_ActFunc18..... 64
iCh_OptJump..... 58
Inpos range 2 active iCh_InPosAct..... 60
Leaving contour completed iCh_Remove-Finish..... 56
NC ready iCh_NCReady..... 54
Override 0% iCh_Override0..... 60
Override 100% iCh_Override100..... 60
Overview on interface signals for block pre-run..... 59
Program end M30 iCh_ProgStopM30..... 55
Program position active iCh_ActFunc05..... 62
Program running iCh_ProgRun..... 54
Program stop M0 iCh_ProgStopM0..... 55
Rapid traverse active iCh_G0Act..... 60
Rapid traverse override 0% iCh_RapO-verride0..... 64
Rapid traverse override 100% iCh_Ra-pOverride100..... 64
Re-entry active iCh_ReEnterAct..... 57
Re-entry active iCh_SRunReEnter..... 60
Ready to return to contour iCh_Ready-ReEnter..... 56
Retrace mode active iCh_RetraceAct..... 61
Return to contour active iCh_SRunRepos..... 60
Returning to approach point active iCh_RetpaAct..... 56
Returning to jog path iCh_RetContPath..... 56
Second feed iCh_Feedrate2..... 64
Tapping active iCh_ActFunc07..... 62
Thread cutting active iCh_ActFunc08..... 63
Thread cycle active iCh_ActFunc06..... 62
Tool correction active bit 0 ... bit 4 iCh_ActFunc09 ... 13..... 63
Tool rotation (TangTool (TTL)) iCh_Tang-TRotCmd..... 61
Input Signals (NC --> PLC)
Retrace mode active iCh_RetraceAct..... 61
Test mode (dry run) iCh_DryRun..... 53
Input signals (NC --> SPS)..... 35
CPL customer output 1 ... 16 iCh_Cpl01 ... 16..... 60
Customer output 1 ... 8 iCh_Custom1 ... 8..... 60
Inputs and outputs of edge-controlled function blocks, final processing..... 231
Inputs and outputs of edge-triggered function blocks, permanent processing..... 237
Inputs and outputs of state-controlled function blocks, final processing..... 228
Inputs and outputs of state-controlled function clocks, permanent processing..... 234
InSync..... 226
Intended use
Areas of application..... 9
Introduction..... 9
Use cases..... 9
InTorque..... 226
InVelocity..... 226
iSp_CAxAct..... 106
iSp_CAxSwitch..... 106
iSp_CoupleErr..... 113
iSp_CoupleIndex_00 ... 02..... 112
iSP_Custom1 ... 8..... 112
iSp_DrvAct..... 116
iSp_DrvChangeClass2..... 114
iSp_DrvErrClass1..... 113
iSp_DrvPower..... 116
iSp_DrvReady..... 116
iSp_DryRun..... 115
iSp_Gear1Sel ... 4Sel..... 110
iSp_GearChange..... 110
iSp_GearIdleSel..... 111
iSp_IdleSpeed..... 110
iSp_InPos..... 107
iSp_Master..... 112
iSp_OrientateAct..... 108
iSp_OrientateFinish..... 108
iSp_Override0..... 111
iSp_Override100..... 111
iSp_PosCtrl..... 107
iSp_ProgSpReach..... 108
iSp_Reset..... 107
iSp_SafCtrlOutputState..... 109
iSp_SafOpMode_00 ... 03..... 109
iSp_SafStatePos..... 109
iSp_ScsState00 ... 15..... 112
iSp_SpLim..... 108
iSp_Stop..... 108
iSp_Synchr1..... 113
iSp_Synchr2..... 113
iSp_ToolLimAct..... 111
iSp_TurnCmd..... 107
iSp_TurnDirM4..... 107

L

Lent axis..... 87
Library
MT_BaseAdditional.library..... 223
MT_PLCopen..... 349
Load database table (MT_DbLoad)
Configuration example..... 173
Function..... 172
Status messages..... 174
Log entry in the action recorder (MT_LogData)

Configuration example 205
Function block parameters..... 204
General information..... 203
Status messages..... 206

M
Machine state display..... 219
Manual-..... 71
Manual+..... 71
MB_Command..... 351
MB_ReadListParameter..... 356
MB_ReadMaxRealValue..... 358
MB_ReadMaxValue..... 360
MB_ReadMinRealValue..... 362
MB_ReadMinValue..... 364
MB_ReadName..... 366
MB_ReadParameter..... 367
MB_ReadRealParameter..... 369
MB_ReadSercosAttribute..... 371
MB_ReadSercosDataStatus..... 372
MB_ReadStringParameter..... 374
MB_ReadUnit..... 376
MB_SERCOS_DATA_STATUS,..... 350
MB_WriteParameter..... 378
MB_WriteRealParameter..... 379
MB_WriteStringParameter..... 381
Messages of operating states..... 115
Miscellaneous..... 336
Monitoring the probe axis..... 280
MT_AxAsyncGetOPMode..... 294
MT_AxAsyncOPMode..... 295, 298
MT_AxAsyncSetOverride..... 301
MT_AxGetChIndex..... 278
MT_AxGetJogMode..... 283
MT_AxGetPowerState..... 302
MT_AxGetPowerStateExt..... 303
MT_AxHandwheelAcs..... 278
MT_AXIS_ASYNC_OPERATION_MODE..... 344
MT_AXIS_JOG_MODE..... 345
MT_AxJogMode..... 289
MT_AxPower..... 308
MT_AxProbeMonitor..... 280
MT_AxSetJogMode..... 285
MT_AxSetPowerOff..... 306
MT_AxSetPowerOn..... 305
MT_BaseAdditional.library..... 223
Axes..... 273
Data types..... 343
Introduction and overview..... 223
Miscellaneous..... 336
NC channel..... 239
Spindles..... 311
Tool management..... 330
MT_CfgData..... 138
MT_CHANNEL_OPERATION_MODE..... 343
MT_CHANNEL_STATE..... 343
MT_ChanRestart..... 135
MT_ChGetActiveProgram..... 244
MT_ChGetFeedOverride..... 255
MT_ChGetOPMode..... 256
MT_ChGetState..... 242
MT_ChNumberValidation..... 240
MT_ChOPMode..... 268
MT_ChSetFeedOverride..... 254
MT_ChSetMainOPMode..... 259
MT_ChSetOPMode..... 263
MT_ChSetReentryBreakPoint..... 251
MT_CplData..... 179
MT_CplDataConv..... 182
MT_DbData..... 162
MT_DbLoad..... 172
MT_DbRecList..... 156
MT_DbSave..... 168
MT_DiagCode..... 191
MT_DiagText..... 195
MT_DistanceAccumulator..... 336
MT_DRIVE_ERROR_CLASS_1..... 345
MT_DRIVE_ERROR_CLASS_2..... 346
MT_DRIVE_POWER_STATE..... 346
MT_DRIVE_SAFE_OP_MODE..... 347
MT_DrvGetSafeState..... 274
MT_FaDataRead..... 211
MT_FaDataWrite..... 215
MT_GetRandomNumber..... 341
MT_Handw..... 140
MT_LogData..... 203
MT_MCoDS..... 199
MT_MT_ChNcProgProgress..... 246
MT_NcBlk..... 131
MT_NcBlkExt..... 132
MT_P_DbData..... 152
MT_P_DbRecList..... 147
MT_PlcOpen.library
Data types..... 350
MT_PLCopen.library..... 349
MT_ProgSel..... 127
MT_RingTop..... 202
MT_S3AutoAdapt..... 201
MT_ScsData..... 142
MT_ScsPos..... 125
MT_SD_RD..... 185
MT_SD_Rd_Seg..... 206
MT_SD_SYS_TOOL..... 344
MT_SD_WR..... 188
MT_SD_Wr_Seg..... 209
MT_SpGetJogSpeed..... 313
MT_SpGetPowerState..... 320
MT_SpGetPowerStateExt..... 322
MT_SPINDLE_JOG_SPEED..... 347
MT_SpJogSpeed..... 317
MT_SpPower..... 327
MT_SpSetJogSpeed..... 314
MT_SpSetOverride..... 312
MT_SpSetPowerOff..... 323, 325
MT_TCorr..... 175

MT_TempComp..... 136
MT_TimeAccumulator..... 339
MT_TmChGetActiveEdge..... 335
MT_TmToolLife..... 330
MZA file..... 220
MZA text..... 220

N
NC block specification (MT_NcBlk)
Configuration example..... 131, 132
Function..... 131
NC channel..... 239
Overview..... 239
NmbOfValues..... 137
NofElem..... 156
Non-bit-coded channel-dependent auxiliary functions..... 120
Non-bit-coded channel-independent auxiliary functions..... 119

O
Operation mode of an asynchronous axis..... 278, 298
Orient spindle..... 99
Original channel..... 87
Output signals (PLC --> NC)..... 32
Overview on axis-related interface signals
Axis-related input signals (PLC --> NC)..... 82
Axis-related output signals (PLC -> NC)..... 70
General description..... 70
General information..... 67
Overview on axis-related input signals (NC --> PLC)..... 69
Overview on axis-related output signals (PLC --> NC)..... 67
Overview on channel-related interface signals
General information..... 37
Overview on global interface signals
General information..... 31
PLC input signals - Overview..... 31
PLC output signals - Overview..... 31
Overview on spindle-related interface signals
General information..... 93
Overview on target groups and product phases... 1

P
Parameterize machine state display..... 219
PLC channel-related output signals..... 41
PLC input signals..... 53
Global interface signals 35
PLC output signals
Global interface signals 32
Profibus DP
I/O configuration..... 123
Profibus DP interface..... 123
Program change..... 127
Program function blocks..... 125
Automatic Sercos III I/O exchange (MT_S3AutoAdapt)..... 201
Backup database table (MT_DbSave)..... 168
Channel restart (MT_ChanRestart)..... 135
Edit data block lists (MT_DbRecList)..... 156
Edit data blocks (MT_P_DbData)..... 152
Edit data records (MT_DbData)..... 162
Edit tool lists (MT_P_DbRecList)..... 147
Extended NC block specification (MT_NcBlkExt)..... 132
General structure of function blocks..... 125
Handwheel data (MT_Handw)..... 140
Load database table (MT_DbLoad)..... 172
Log entry in the action recorder (MT_LogData)..... 203
NC block specification (MT_NcBlk)..... 131
Program selection/deselection..... 127
Read actual axis values (MT_ScsPos)..... 125
Read diagnostic data (MT_DiagCode)..... 191
Read diagnostic texts (MT_DiagText)..... 195
Read fast action input data (MT_FaDataRead)..... 211
Read machine parameters (MT_CfgData)..... 138
Read Motion Control Data Services (MT_MCoDS)..... 199
Read system data (MT_SD_RD)..... 185
Read system data is segments (MT_SD_Rd_Seg)..... 206
Read/write permanent CPL variables (MT_CplData)..... 179
Read/Write permanent CPL variables with type conversion (MT_CplDataConv)..... 182
Sercos parameters (MT_ScsData)..... 142
Sercos ring topology (MT_RingTop)..... 202
Temperature compensation..... 136
Tool correction (MT_TCorr)..... 175
Write fast action input data (MT_FaDataWrite)..... 215
Write system data (MT_SD_WR)..... 188
Write system data is segments (MT_SD_Wr_Seg)..... 209
Program selection/deselection (MT_ProgSel)
Configuration example..... 129
Function..... 127
Status messages..... 130
ProgramName..... 129
Punching HS logic..... 33
Purpose..... 1

Q
qAx_Custom1 ... 8..... 78
qAx_Discharge..... 80
qAx_DrvLock..... 81
qAx_DrvOn..... 81
qAx_FrzIpoPos..... 80
qAx_FxStopRel..... 74

qAx_HandwDir ... 76
qAx_HandwPosMode ... 77
qAx_HandwSel_00 ... 01 ... 76
qAx_HomeSwitch ... 78
qAx_JogDia ... 73
qAx_JogInch ... 73
qAx_JogMinus ... 71
qAx_JogPlus ... 71
qAx_LagErrOff ... 79
qAx_ManFeed_00 ... 03 ... 77
qAx_MasterPos ... 79
qAx_NextNotch ... 73
qAx_OpModeSel_00 ... 01 ... 71
qAx_Override_00 ... 15 ... 78
qAx_Override100 ... 78
qAx_RedDelayErrClass1 ... 80
qAx_Reset ... 73
qAx_SafCheckInputState ... 75
qAx_SafDrvLock ... 74
qAx_SafEnablCtrl ... 75
qAx_SafOpModeSwitch ... 74
qAx_SafRedTransTime ... 76
qAx_SafSwitch1 ... 75
qAx_SafSwitch2 ... 75
qAx_SafTechState ... 75
qAx_SwLimOff ... 74
qAx_TakeActOffs ... 80
qAx_TrqErrOff ... 78
qAx_TrqLim ... 80
qAx_TrvLim_00 ... 01 ... 74
qCh_ASub1 ... 8 ... 47
qCh_BlkSlash ... 48
qCh_CancDist ... 45
qCh_CancSyntaxError ... 45
qCh_CoordCoupleOff ... 51
qCh_CtrlReset ... 44
qCh_Custom1 ... 8 ... 50
qCh_FeedHold ... 44
qCh_Feedrate2 ... 52
qCh_FeedStop ... 44
qCh_HandwDirWcs ... 48
qCh_HandwPosMode ... 48
qCh_HandwSelWcs_00 ... 47
qCh_JogPlusWcs ... 47
qCh_MachineFunction ... 44
qCh_NCStart ... 43
qCh_NextBlk ... 46
qCh_OnlCorrWcs ... 50
qCh_OnlCorrWcsDir ... 50
qCh_OpModePlc ... 43
qCh_OpModeSel_00 ... 03 ... 41
qCh_OptJump ... 48
qCh_OptStop ... 48
qCh_Override_00 ... 15 ... 49, 51
qCh_Override100 ... 49
qCh_Rapid100 ... 49
qCh_RedRap ... 49
qCh_ReSelOff ... 44
qCh_Restart ... 43
qCh_RetCont ... 46
qCh_Retrace ... 50
qCh_RetraceMovFwd ... 50
qCh_RetraceMovRev ... 51
qCh_Retract ... 47
qCh_TangTRotRel ... 51
qCh_TestFeed ... 51
qCh_TestRap ... 52
qCh_TransferLock ... 43
qGen_Crash ... 34
qGen_CrashStoreDataAckn ... 34
qGen_DelErrInfo ... 32
qGen_Reset ... 32
qGen_SercosIO_Reset ... 32
qGen_StrokeInhibit ... 33
qGen_StrokeRel ... 33
qGen_StrokeReserv ... 33
qSp_CAxOff ... 97
qSp_CAxOn ... 96
qSp_Custom1 ... 8 ... 104
qSp_DrvLock ... 105
qSp_DrvOn ... 104
qSp_Gear1Act ... 4Act ... 102
qSp_GearIdleAct ... 102
qSp_HomeSwitch ... 103
qSp_JogMinus ... 97
qSp_JogPlus ... 97
qSp_ManSpeed_00 ... 02 ... 102
qSp_Orientate ... 99
qSp_Override_00 ... 15 ... 104
qSp_Override100 ... 104
qSp_RedDelayErrClass1 ... 104
qSp_Reset ... 98
qSp_SafCheckInputState ... 101
qSp_SafDrvLock ... 100
qSp_SafEnablCtrl ... 101
qSp_SafOpModeSwitch ... 100
qSp_SafRedTransTime ... 102
qSp_SafSwitch1 ... 101
qSp_SafSwitch2 ... 101
qSp_SafTechState ... 101
qSp_Stop ... 98
qSp_SValueSD ... 103
qSp_TurnCCW ... 98
qSp_TurnCW ... 98

R

Rapid traverse G0 ... 49
Read actual axis values (MT_ScsPos)
Configuration example ... 126
Function ... 125
Status messages ... 127
Read data blocks ... 162
Read diagnostic data (MT_DiagCode)
Configuration example ... 195
Function block parameters ... 192

General information..... 191
Status messages..... 195
Read diagnostic texts (MT_DiagText)
Configuration example..... 199
Function block parameters..... 196
General information..... 195
Status messages..... 199
Read Fast Action input data (MT_FaDataRead)
Configuration example..... 212
Function block parameters..... 212
General information..... 211
Status messages..... 214
Read machine parameters (MT_CfgData)
Configuration example..... 139
Function..... 138
Status messages..... 140
Read Motion Control Data Services (MT_MCoDS)
Configuration example..... 200
Function block parameters..... 200
General information..... 199
Status messages..... 200
Read parameter value..... 142
Read system data (MT_SD_RD)
Configuration example..... 187
Function..... 185
Status messages..... 187
Read system data in segments (MT_SD_Rd_Seg)
Configuration example..... 207
Function block parameters..... 207
General information..... 206
Status messages..... 208
Read tool data..... 152
Read/write permanent CPL variables (MT_CplData)
Configuration example..... 181
Function..... 179
Status messages..... 182
Read/write permanent CPL variables with type conversion (MT_CplDataConv)
Configuration example..... 184
Function..... 182
Status messages..... 185

S

Safe absolute position..... 86
Safety instructions..... 5
Saving breakpoint..... 251
SD qSp_SpeedLimit..... 103
Search for data blocks..... 156
Search for tool data blocks..... 147
SelMask..... 168, 172
Sercos I/O device errors cleared..... 35
Sercos parameter (MT_ScsData)
Configuration example..... 143
Status messages..... 146
Sercos parameters (MT_ScsData)
Function..... 142
Sercos ring topology (MT_RingTop)
Configuration example..... 203
General information..... 202
Status messages..... 203
Sercos Ring topology (MT_RingTop)
Function block parameters..... 202
Service hotline..... 385
Set correction values..... 175
SetPos offset..... 62
Sets controller enable for a spindle..... 325
Sets the controller enable for a spindle..... 323
Sets the controller enable for an axis..... 305, 306
Shutdown..... 227
Signal alert symbol..... 6
Signal description
Output signals (PLC --> NC)..... 41
Spindle-related input signals (NC--> PLC)..... 106
Spindle-related output signals (PLC--> NC)..... 96
Signal words..... 6
Sp_DelayErrClass1..... 113
Specify parameter value..... 142
Spindle clockwise..... 98
Spindle counterclockwise..... 98
Spindle stop..... 98
Spindle-related input signals (NC --> PLC)
C-axis active iSp_CAxAct..... 106
C-axis switching iSp_CAxSwitch..... 106
Change in state class 2 iSp_DrvChange-Class2..... 114
Change in state class 3 iSp_DrvChange-Class3..... 115
Command of rotation iSp_TurnCmd..... 107
Coupling error iSp_CoupleErr..... 113
Customer output 1 ... 8 iSP_Custom1 ... 8..... 112
Delayed error reaction Sp_DelayErr-Class1..... 113
Direction of rotation M4 iSp_TurnDirM4..... 107
Drive in operation iSp_DrvAct..... 116
Drive ready iSp_DrvReady..... 116
Enabled for power activation iSp_DrvPower..... 116
Error of state class 1 iSp_DrvErrClass1..... 113
General information..... 106
GTS switching iSp_GearChange..... 110
Idling speed reached iSp_IdleSpeed..... 110
Orient spindle active iSp_OrientateAct..... 108
Position control active iSp_PosCtrl..... 107
SCS signal status 0 ...15 iSp_ScsState00 ... 15..... 112
Selection gear-idling iSp_GearIdleSel..... 111
Selection GTS 1 ... 4 iSp_Gear1Sel ... 4Sel..... 110
Speed limited iSp_SpLim..... 108
Speed reached iSp_ProgSpReach..... 108
Spindle in position iSp_InPos..... 107

Spindle is master iSp_Master..... 112
Spindle oriented iSp_OrientateFinish..... 108
Spindle override 0% iSp_Override0..... 111
Spindle override 100% iSp_Override100..... 111
Spindle reset iSp_Reset..... 107
Spindle stop iSp_Stop..... 108
Status of safe position iSp_SafStatePos..... 109
Status output of controller iSp_SafCtrlOutputState..... 109
Synchronous run 1 iSp_Synchr1..... 113
Synchronous run 2 iSp_Synchr2..... 113
Test mode (dry run) iSp_DryRun..... 115
Spindle-related input signals (NC--> PLC)
No. of coupling bit 0 ... bit 2 iSp_CoupleIndex_00 ... 02..... 112
Si operation mode bit 0 ... bit 3 iSp_SafOpMode_00 ... 03..... 109
Tool limitations active..... 111
Spindle-related interface signals..... 93
Overview on spindle-related interface signals..... 93
PLC input signals - overview 29, 95
PLC output signals - overview 27, 93
Signal description..... 96
Spindle-related output signals (PLC --> NC)
Acknowledgement GTS 1...4 qSp_Gear1Act ... 4Act..... 102
Acknowledgement idling qSp_GearIdleAct..... 102
C-axis off qSp_CAxOff..... 97
C-axis on qSp_CAxOn..... 96
Check input of safety status qSp_SafCheckInputState..... 101
Drive inhibit qSp_SafDrvLock..... 100
Drive on qSp_DrvOn..... 104
Enabling button qSp_SafEnablCtrl..... 101
General information..... 96
Jog speed bit 0 ... bit 2 qSp_ManSpeed_00 ... 02..... 102
Jog spindle M3 qSp_JogPlus..... 97
Jog spindle M4 qSp_JogMinus..... 97
Operation mode selection qSp_SafOpModeSwitch..... 100
Override bit 0 ... ... bit 15 qSp_Override_00 ... 15..... 104
Override Bit 0 ... Bit 15 qSp_Override_00 ... 15..... 104
reduce transmission time qSp_SafRedTransTime..... 102
Reduced delayed error reaction qSp_RedDelayErrClass1..... 104
Reference switch qSp_HomeSwitch..... 103
S-value specification via SD qSp_SValueSD..... 103
Safety switch 1 (S2) qSp_SafSwitch1..... 101
Safety switch 2 (S2) qSp_SafSwitch2..... 101
Speed limitation SD qSp_SpeedLimit..... 103
Spindle inhibit qSp_DrvLock..... 105
Spindle M3 manual qSp_TurnCW..... 98
Spindle M4 manual qSp_TurnCCW..... 98
Spindle M5 manual qSp_Stop..... 98
Spindle M19 manual qSp_Orientate..... 99
Spindle override 100 % qSp_Override100..... 104
Spindle reset qSp_Reset..... 98
Status of safety signals qSp_SafTechState..... 101
Spindles..... 311
Overview..... 311
Standard interfaces at function blocks..... 225
Starting block..... 129
Static signals..... 70
Stroke inhibit..... 33
Stroke intended..... 35
Stroke not running..... 35
Stroke on..... 33
Stroke reservation..... 33
Structure contains all channel states..... 343
Structure contains all Sercos attributes..... 350
Structure contains all Sercos command states..... 350
Structure contains the safety operating state of a drive..... 347
Structure error details of S-0-0011 of an axis... 345
Structure of the SD.SysTool system date..... 344
Structure warning details of S-0-0012 of an axis..... 346
Structure with all axis operation modes..... 344
Structure with all channel operation modes..... 343
Structure with jog distances of an axis..... 345
Structure with spindle jog velocities..... 347
Suggestions..... 7
Support..... 385
Switch off prewarning..... 114
Symbols..... 7
System reset..... 32
System reset executed..... 35

T

TangTool..... 61
Target groups..... 1
Temperature compensation (MT_TempComp)
Configuration example..... 137
Function..... 136
Status messages..... 138
ThreadSet..... 62
Tool correction (MT_TCorr)
Status messages..... 179
Tool corrections (MT_TCorr)
Configuration example..... 177
Function..... 175
Tool management..... 330
Overview..... 330
Tool, tool life counting..... 330
TypeId..... 162

U

Unintended use..... 10
Consequences, disclaimer..... 9
UTF8 character coding..... 221

W

Warnings..... 5
Write fast action output data (MT_FaDataWrite)
General information..... 215
Write Fast Action output data (MT_FaDataWrite)
Configuration example..... 216
General information..... 215
Status messages..... 215
Write system data..... 188
Write system data (MT_SD_WR)
Configuration example..... 189
Function..... 188
Status messages..... 190
Write system data in segments (MT_SD_Wr_Seg)
Configuration example..... 210
Function block parameters..... 210
General information..... 209
Status messages..... 210

X

XPath..... 186, 188

Notes

Notes

rexroth
A Bosch Company

Bosch Rexroth AG
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr a.Main
Germany
Tel. +49 9352 18 0
Fax +49 9352 18 8400
www.boschrexroth.com/electrics

R911400172

DOK-MTX***-PLC*INT*V15-PR04-EN-P