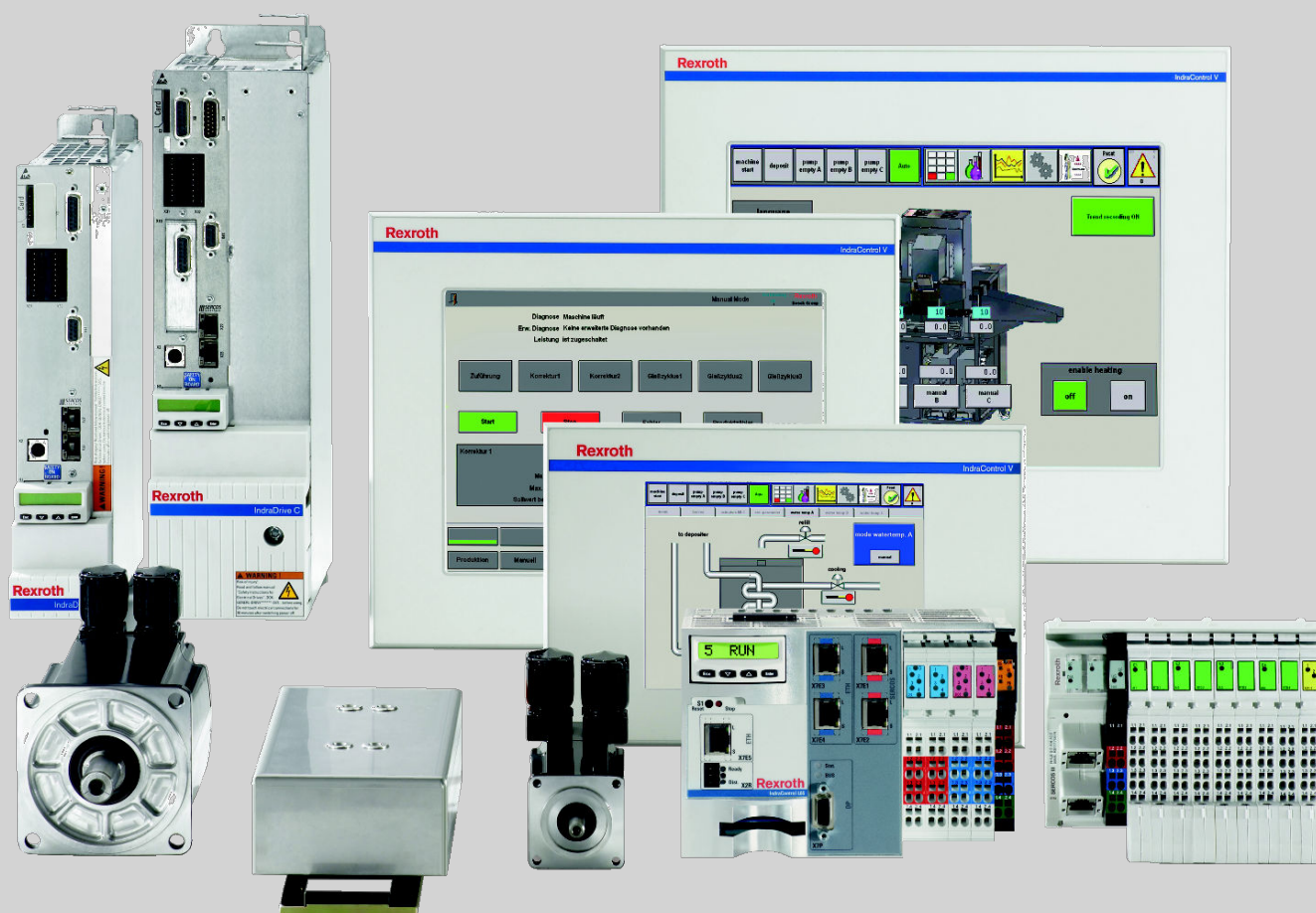


IndraLogic XLC IndraMotion MLC 14VRS

PLCopen Libraries

Library
R911341491

Edition 08



Title	IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries
Type of Documentation	Library
Document Typecode	DOK-XLCMLC-FUNLIB**V14-LI08-EN-P
Internal File Reference	RS-15f349eac3d2ab970a6846a5018a4f71-8-en-US-3
Change Record	Edition 08, 2020-03 Refer to tab. 1-1 "Change Record" on page 1
Copyright	© Bosch Rexroth AG 2020 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 Automation Systems - Solution Integration Control Hardware, PLC and Safety Control (KWA) Technical Editing (MiNi/MePe)

Table of Contents

	Page
1 About this documentation.....	1
1.1 Validity of the documentation.....	1
1.2 Using safety instructions.....	2
1.2.1 Structure of the safety instructions.....	2
1.2.2 Explaining signal words and safety alert symbol.....	2
1.2.3 Symbols used.....	4
1.2.4 Explaining the signal alert symbol on the device.....	4
1.3 Required and supplementing documentation XLC/MLC.....	4
2 Standard interfaces at function blocks.....	11
2.1 Motivation.....	11
2.2 Function block types.....	11
2.3 Block types and their interfaces.....	12
2.4 Inputs and outputs of state-controlled function blocks, final processing.....	13
2.5 Inputs and outputs of edge-controlled function blocks, final processing.....	16
2.6 Inputs and outputs of state-controlled function clocks, permanent processing.....	19
2.7 Inputs and outputs of edge-triggered function blocks, permanent processing.....	22
3 RIL_CommonTypes.library.....	25
3.1 Introduction and overview.....	25
3.2 Global_Variables.....	26
3.2.1 IL_NULL.....	26
3.3 Data types to reference objects of Motion function blocks	26
3.3.1 Referencing objects, introduction and overview.....	26
3.3.2 CONTROLS.....	27
3.3.3 AXIS_REF.....	27
3.3.4 OBJECTS.....	28
3.3.5 AXIS_GROUP_REF.....	29
3.3.6 KIN_OBJECTS.....	30
3.3.7 TOUCHPROBE_REF.....	31
3.3.8 TP_OBJECTS.....	31
3.4 Data types to support cams, MotionProfiles and FlexProfiles	32
3.4.1 Data type to switch between cams, general information.....	32
3.4.2 MC_CAM_ID.....	32
3.4.3 MC_STEP_MODE, MotionProfile extended to FlexProfile	32
3.5 Data types to synchronize in the position-controlled synchronous operation modes	56
3.5.1 Synchronization, introduction and overview.....	56
3.5.2 MC_START_MODE.....	57
3.5.3 MC_SYNC_DIRECTION.....	59
3.6 Data types to diagnose errors in the Motion function blocks	60
3.6.1 ERROR_CODE.....	60
3.6.2 ERROR_STRUCT.....	61
3.6.3 ERROR_TABLE.....	62

Table of Contents

	Page
3.7 Data types to report date and absolute time specifications for Motion function blocks	63
3.7.1 Reporting date and absolute time specification, introduction and overview.....	63
3.7.2 SYS_TIME64.....	63
3.7.3 EXT_SYS_TIME_DATE.....	63
3.7.4 SYS_TIME_DATE.....	64
3.8 Data types to support touch probe functions	64
3.8.1 Data types to support touch probe functions, general information.....	64
3.8.2 MB_DIFF_EVAL.....	65
3.8.3 MB_EXPECT_WINDOW_MODE.....	66
3.8.4 MB_INTERPOLATION_MODE.....	66
3.8.5 MB_POSITION_EVAL.....	67
3.8.6 MB_PROBE_MODULE.....	67
3.8.7 MB_PROBE_INPUT.....	67
3.8.8 MB_PROBE_DATA.....	68
3.8.9 MB_PROBE_CONFIG.....	69
3.8.10 MB_PROBE_CONFIG, special features of Sercos-based axes.....	69
 4 ML_Base.library.....	 71
4.1 Introduction and overview.....	71
4.2 Data types	75
4.2.1 Data types/common types - general types.....	75
MB_IDN.....	75
4.2.2 Data Types/ Flex Structs.....	76
Data types to support Motion and FlexProfile functions, general information.....	76
MB_CHANGESTEP_MODE.....	76
MB_EVENT_ACTION.....	77
MB_EVENT_TRIGGER.....	78
MB_EXECUTION_MODE.....	78
MB_FLEX_EVENT.....	79
MB_MASTER_TYPE.....	79
MB_PROFILE_START_POINT.....	79
MB_PROFILE_STEP.....	81
MB_STEP_TYPE.....	81
MB_SYNC_TYPE.....	82
4.2.3 Data types/read write parameter - parameter access.....	82
MB_SERCOS_DATA_STATUS.....	82
4.2.4 Data types/shared memory - direct access.....	83
Data types to support the data exchange between Motion and PLC.....	83
Axis data via shared memory and function variable access.....	83
Axis data via shared memory and function variable access, general.....	83
AxisData[] - axis data in shared memory.....	85
DV_Axis[] - axis data via function variable access.....	87
GV_LinkAxis[] - axis data via synchronous function variable access in C2C link	90
Control data via shared memory and function variable access.....	92
Control data via shared memory and function variable access, general.....	92
ControlData[] - control data in shared memory.....	93

	Page
DV_Control[] - control data via function variable access	94
Programmable limit switch data via shared memory and function variable access.....	97
Programmable limit switch data via shared memory and function variable access, general.....	97
PlsData[] - PLS data in shared memory.....	98
DV_Pls[] - PLS data via function variable access.....	100
Touch probe data via shared memory and function variable access.....	102
Touch probe data via shared memory and function variable access, general.....	102
ProbeData[] - touch probe data in shared memory.....	103
DV_Probe[] - touch probe data via function variable access.....	105
Oscilloscope data via function variable access.....	107
Oscilloscope data via function variable access, general.....	107
DV_Osci[] - oscilloscope data via function variable access.....	108
ML_AXISDATA_SM.....	110
ML_CONTROLDATA_SM.....	111
ML_PLSDATA_SM.....	112
ML_PROBEDATA_SM.....	113
ML_LINK.....	113
ML_SERCOS.....	114
ML_SYNC.....	114
SM_TYPES.....	114
SM_EXT_TYPES.....	115
4.3 Error/diagnostic memory - supporting user diagnostics and logbook access.....	116
4.3.1 ML_ReadDiagLog.....	116
4.3.2 ML_SetAxisDiagEntry.....	117
4.3.3 ML_SetDiagEntry.....	119
4.3.4 E_LOG_CLASS.....	120
4.3.5 E_LOG_FACILITY.....	121
4.3.6 E_LOG_SEVERITY.....	121
4.3.7 ML_LOGBOOK.....	121
4.4 Execute.....	122
4.4.1 ML_TaskExecuteFile.....	122
4.4.2 ML_TaskExecuteGetActive.....	122
4.4.3 ML_TaskExecuteGetName.....	123
4.4.4 ML_TaskExecuteGetStatus.....	123
4.4.5 ML_TaskExecuteKill.....	124
4.4.6 ML_TASKHANDLE.....	124
4.4.7 ML_ProcessState.....	125
4.5 Global Variable List.....	125
4.5.1 GVL_Base.....	125
4.5.2 GVL_BaseConstant.....	126
4.5.3 Global Variable Lists/Global Variable Index.....	127
GVL_A_GlobalVarIdx.....	127
4.5.4 Parameter constants.....	128
Access to data of an axis, control or another object.....	128
GVL_A_Param_ID.....	130
GVL_C_Param_ID.....	130

Table of Contents

	Page
GVL_M_Param_ID.....	130
GVL_N_Param_ID.....	131
GVL_O_Param_ID.....	131
GVL_P_Param_ID.....	132
GVL_S_Param_ID.....	132
4.6 Interfaces, functions and function blocks.....	133
4.6.1 IL_SercosAttribute.....	133
4.6.2 IL_SercosElementsToParaNo.....	135
4.6.3 IL_SercosStringToParaNo.....	137
4.7 IP settings - IP configuration.....	138
4.7.1 ML_GetGateway.....	138
4.7.2 ML_GetIpAddress.....	139
4.7.3 ML_GetMacAddress.....	141
4.7.4 ML_GetSubnetMask.....	142
4.7.5 ML_SetGateway.....	143
4.7.6 ML_SetIpAddress.....	144
4.7.7 ML_SetSubnetMask.....	145
4.8 SystemTime - system time.....	146
4.8.1 IL_SetDateTime.....	146
4.8.2 ML_GetSystemTimeUTC.....	147
4.8.3 ML_SetSystemTimeUTC.....	149
4.8.4 ML_DAYSOFWEEK.....	150
4.9 Target - Target system.....	151
4.9.1 ML_GetCpuLoad.....	151
4.9.2 ML_GetDiagnosisLog.....	151
4.9.3 ML_GetFirmwareString.....	152
4.9.4 ML_GetHardwareInfoCtrl.....	153
4.9.5 ML_GetInfoApplication.....	153
4.9.6 ML_GetMaxTemperature.....	154
4.9.7 ML_GetNewestDiagnosisIndex.....	155
4.9.8 ML_GetOhcFan.....	155
4.9.9 ML_GetScalingType.....	156
4.9.10 ML_GetSystemSpecialPath.....	156
4.9.11 ML_GetTargetInfo.....	157
4.9.12 ML_GetTemperature.....	157
4.9.13 ML_GetWarningCpuLoad.....	158
4.9.14 ML_SetWarningCpuLoad.....	158
4.9.15 APP_INFO.....	159
4.9.16 E_SCALING_TYPE.....	159
4.9.17 E_SYSTEM_SPECIAL_PATH.....	159
4.9.18 INFO_APPLICATIONS.....	160
4.9.19 ML_DIAGNOSIS_LOG.....	160
4.9.20 TARGET_INFO.....	161
4.10 Transmission.....	161
4.10.1 ML_SystemGetTransmissionInfo.....	161
4.10.2 ML_SystemGetTransmissionStatus.....	162

	Page
4.10.3 ML_SystemStartTransmission.....	162
4.10.4 ML_SystemTftpGetRetransmitCount.....	163
4.10.5 ML_SystemTftpGetTimeout.....	163
4.10.6 ML_SystemTftpSetRetransmitCount.....	164
4.10.7 ML_SystemTftpSetTimeout.....	164
4.10.8 ML_TransmissionInfo.....	164
4.10.9 ML_TransmissionStatus.....	165
5 Function blocks according to PLCopen specification.....	167
6 ML_PLCopen.library.....	171
6.1 Introduction and overview.....	171
6.2 Global_Variables.....	175
6.2.1 ProbeSelect.....	175
6.3 Data types/data types of TouchProbe.....	175
6.3.1 PROBECONFIG.....	175
6.4 State diagram of a real axis.....	176
6.5 Function blocks for single-axis control.....	179
6.5.1 MC_Power.....	179
6.5.2 MC_Reset.....	180
6.5.3 MC_Stop.....	181
6.5.4 MC_MoveAbsolute.....	183
6.5.5 MC_MoveAdditive.....	186
6.5.6 MC_MoveRelative.....	188
6.5.7 MC_MoveVelocity.....	192
6.5.8 MC_TorqueControl.....	195
6.5.9 MB_SetPositionControlMode.....	198
6.5.10 MB_Home.....	199
6.6 Function blocks for multi-axis control.....	200
6.6.1 The MotionControl action chain.....	200
6.6.2 MB_Phasing.....	203
6.6.3 MB_PhasingSlave.....	205
6.6.4 MC_CamIn.....	207
6.6.5 MC_CamIn, special features of Sercos-based axes.....	209
6.6.6 MC_CamOut.....	211
6.6.7 MB_MotionProfile.....	212
6.6.8 MB_MotionProfile, special features of Sercos-based axes.....	215
6.6.9 PLCopen function blocks to support MotionProfiles.....	217
General information.....	217
MB_ChangeProfileSet	218
MB_ChangeProfileSet, special features of Sercos-based axes.....	220
MB_ChangeProfileStep	222
MB_ChangeProfileStep, special features of Sercos-based axes.....	224
MB_ChangeCamData	225
6.6.10 ML_FlexProfile.....	227

Table of Contents

	Page
6.6.11	ML_FlexProfile, special features of Sercos-based axes..... 230
6.6.12	Switching and Synchronizing the FlexProfile 233
	Selecting the Coordinate System..... 233
	Switching and Synchronizing 233
	Processing Types..... 236
6.6.13	PLCopen function blocks for supporting FlexProfiles..... 237
	General information..... 237
	MB_ChangeFlexProfileSet 239
	Characteristics of the Event Control..... 240
	MB_ChangeFlexEventSet 242
6.6.14	MC_GearIn..... 243
6.6.15	MC_GearIn, special features of Sercos-based axes..... 245
6.6.16	MC_GearOut..... 247
6.6.17	MB_GearInPos..... 248
6.6.18	MB_GearInPos, special features of Sercos-based axes..... 249
6.7	Function block for the axis group control..... 251
6.7.1	MC_AddAxisToGroup..... 251
6.7.2	MC_RemoveAxisFromGroup..... 252
6.8	Function blocks to use the cyclic data channel..... 253
6.8.1	General information on using the cyclic data channel..... 253
6.8.2	ML_OpenCyclicAnalogChannel..... 255
6.8.3	ML_WriteCyclicAnalog..... 256
6.8.4	ML_OpenCyclicPositionChannel..... 257
6.8.5	ML_WriteCyclicPosition..... 259
6.8.6	ML_OpenCyclicTorqueChannel..... 261
6.8.7	ML_WriteCyclicTorque..... 263
6.8.8	ML_OpenCyclicVelocityChannel..... 264
6.8.9	ML_WriteCyclicVelocity..... 265
6.9	Function blocks for controlled motion..... 266
6.9.1	General information on controlled motion..... 266
6.9.2	ML_ControlOn..... 267
6.9.3	ML_ControlOff..... 268
6.9.4	ML_ControlWriteActualData..... 269
6.9.5	ML_OpenCyclicAnalogChannelCtrl..... 270
6.9.6	ML_WriteCyclicAnalogCtrl..... 271
6.9.7	ML_OpenCyclicPositionChannelCtrl..... 272
6.9.8	ML_WriteCyclicPositionCtrl..... 273
6.9.9	ML_OpenCyclicTorqueChannelCtrl..... 274
6.9.10	ML_WriteCyclicTorqueCtrl..... 275
6.9.11	ML_OpenCyclicVelocityChannelCtrl..... 276
6.9.12	ML_WriteCyclicVelocityCtrl..... 277
6.10	Parameter access..... 278
6.10.1	Access to data of an axis, control, kinematics (axis group) or another object..... 278
6.10.2	Time required for data transport with read and WriteParameter function blocks..... 280
6.10.3	MB_Command..... 281
6.10.4	MB_Command, special features of Sercos-based axes..... 286

	Page
6.10.5 MB_ReadLintParameter.....	286
6.10.6 MB_ReadListParameter.....	288
6.10.7 MB_ReadRealListParameter.....	291
6.10.8 MB_ReadRealParameter.....	294
6.10.9 MB_ReadMaxRealValue.....	295
6.10.10 MB_ReadMaxValue.....	297
6.10.11 MB_ReadMinRealValue.....	299
6.10.12 MB_ReadMinValue.....	301
6.10.13 MB_ReadName.....	302
6.10.14 MB_ReadParameter.....	304
6.10.15 MB_ReadRealParameter.....	306
6.10.16 MB_ReadSercosAttribute.....	308
6.10.17 MB_ReadSercosDataStatus.....	309
6.10.18 MB_ReadStringParameter.....	311
6.10.19 MB_ReadStringParameterExt.....	313
6.10.20 MB_ReadUnit.....	315
6.10.21 MB_WriteLintParameter.....	316
6.10.22 MB_WriteListParameter.....	318
6.10.23 MB_WriteLrealParameter.....	321
6.10.24 MB_WriteParameter.....	322
6.10.25 MB_WriteRealParameter.....	324
6.10.26 MB_WriteStringParameter.....	326
6.10.27 MB_WriteStringParameterExt.....	327
6.10.28 MC_ReadActualPosition.....	329
6.10.29 MC_ReadAxisError.....	330
6.10.30 MC_ReadStatus.....	332
6.11 Function blocks to use the touch probe function.....	333
6.11.1 Data and function blocks to use the touch probe function.....	333
6.11.2 ML_AbortTrigger.....	334
6.11.3 ML_InitTouchProbe.....	336
6.11.4 ML_TouchProbe.....	344
6.11.5 ML_TouchProbeContinuous.....	347
6.11.6 ML_WriteExpectWindow.....	349
6.12 Time required to process Motion function blocks in the IndraMotion MLC.....	353
6.13 Axis types and function blocks supported by these types.....	355
6.14 Overview: Diagnostics.....	357
6.14.1 SERCOS_TABLE, ERROR_TABLE 16#0010	357
6.14.2 MLC_TABLE, ERROR_TABLE 16#0030	360
7 RMB_PLCOpenFieldBus.library.....	361
7.1 Overview and introduction.....	361
7.2 Parameter address constants.....	361
7.2.1 Access to data of a (field bus) drive.....	361
7.2.2 GVL_S_Param_ID.....	361
7.2.3 GVL_P_Param_ID.....	362
7.3 Data types.....	362

Table of Contents

	Page
7.3.1 Data types, general.....	362
7.3.2 MB_IDN.....	362
7.3.3 AXIS_REF.....	363
7.4 Function blocks for single-axis control (Motion_FBs).....	365
7.4.1 Motion_FBs, general.....	365
7.4.2 MC_Power.....	367
7.4.3 MC_Reset.....	368
7.4.4 MC_Stop.....	369
7.4.5 MB_Stop.....	371
7.4.6 MB_Home.....	373
7.4.7 MC_MoveAbsolute.....	374
7.4.8 MC_MoveAdditive.....	376
7.4.9 MC_MoveRelative.....	379
7.4.10 MC_MoveVelocity.....	382
7.5 Function blocks for parameter access (Parameter_FBs).....	385
7.5.1 Parameter_FBs, general information.....	385
7.5.2 Access to data of a drive (IndraDrive).....	387
7.5.3 Special Features of Twin or Multi-Axis Modules.....	387
7.5.4 MB_ReadListParameter.....	389
7.5.5 MB_ReadListSegParameterPtr.....	392
7.5.6 MB_ReadMaxRealValue.....	394
7.5.7 MB_ReadMaxValue.....	396
7.5.8 MB_ReadMinRealValue.....	397
7.5.9 MB_ReadMinValue.....	399
7.5.10 MB_ReadParameter.....	400
7.5.11 MB_ReadRealParameter.....	402
7.5.12 MB_ReadSercosAttribute.....	403
7.5.13 MB_ReadStringParameter.....	405
7.5.14 MB_WriteListParameter.....	406
7.5.15 MB_WriteListSegParameterPtr.....	409
7.5.16 MB_WriteParameter.....	411
7.5.17 MB_WriteRealParameter.....	413
7.5.18 MB_WriteStringParameter.....	414
7.5.19 MC_ReadActualPosition.....	415
7.5.20 MC_ReadAxisError.....	417
7.5.21 MC_ReadStatus.....	418
7.6 System_FBs	420
7.7 PLC controls drives via field bus connection (field bus axes).....	420
7.7.1 General information.....	420
7.7.2 Creating field bus axes in the project.....	421
7.7.3 Drive settings for the PLC communication.....	422
General information.....	422
IndraDrive communication settings.....	422
7.7.4 Including required libraries.....	423
7.7.5 Addressing field bus drives in the PLC project.....	424
AXIS_REF, axis addresses of field bus drives.....	424

	Page
AxisData[], axis data.....	426
7.7.6 I/O addressing – RMB_PLCOpenFieldbus.....	426
7.7.7 I/O addressing – RMB_PLCOpenEtherCAT_SoE.....	427
8 RMB_PLCOpenEtherCAT_SoE.library.....	429
8.1 Introduction and overview.....	429
8.2 IndraDrive – Required configurations.....	429
8.2.1 Configuring the operation modes (e.g. with FWA MPB20).....	430
8.2.2 Configuration of signal status word and signal control word.....	430
8.2.3 Configuration using the parameter file.....	431
8.2.4 EtherCAT master – Configuring the drive.....	432
8.2.5 IndraDrive slave – Import.....	433
8.2.6 Configuring cyclic data.....	437
8.3 Data types.....	438
8.3.1 Data types, general.....	438
8.3.2 AXIS_REF.....	438
8.4 Function blocks for single-axis control (Motion_FBs).....	439
8.4.1 Motion_FBs, general information.....	439
8.4.2 MC_Power.....	441
8.4.3 MC_Reset.....	442
8.4.4 MC_Stop.....	444
8.4.5 MB_Stop.....	445
8.4.6 MB_Home.....	447
8.4.7 MC_MoveAbsolute.....	448
8.4.8 MC_MoveAdditive.....	451
8.4.9 MC_MoveRelative.....	453
8.4.10 MC_MoveVelocity.....	456
8.5 Function blocks for parameter access (Parameter_FBs).....	459
8.5.1 Parameter_FBs, general information.....	459
8.5.2 MC_ReadActualPosition.....	460
8.5.3 MC_ReadAxisError.....	461
8.5.4 MC_ReadStatus.....	463
8.6 System_Function blocks	464
8.7 PLC controls drives via field bus connection (field bus axes).....	465
8.7.1 General information.....	465
8.7.2 Creating field bus axes in the project.....	465
8.7.3 Drive settings for the PLC communication.....	466
General information.....	466
IndraDrive communication settings.....	466
8.7.4 Including required libraries.....	467
8.7.5 Addressing field bus drives in the PLC project.....	468
AXIS_REF, axis addresses of field bus drives.....	468
AxisData[], axis data.....	470
8.7.6 I/O addressing – RMB_PLCOpenFieldbus.....	470
8.7.7 I/O addressing – RMB_PLCOpenEtherCAT_SoE.....	471

	Page
9 ML_ControllerBase.library	473
9.1 Introduction and overview	473
9.2 Basic class of the controller	473
9.2.1 Basic class	473
9.2.2 Controller Data	474
9.2.3 Data conversion of a controller	475
9.2.4 Interface IActualValue	476
9.2.5 Interface of the IModuleControl	476
9.2.6 Interface of the IOperationModeHandler	476
9.2.7 Interface of the IOutputData	476
9.3 Controller management	476
9.3.1 Connecting the controller to the controller management	476
9.3.2 Cyclic procedure of the controller management	477
9.3.3 State machine: Operating state	477
9.3.4 Cyclic procedure of the controller	478
9.3.5 Operation mode handling	478
9.3.6 Predefined generators of the operation modes	481
9.3.7 Diagnostic handling	482
10 ML_IPOImpactControl.library	483
10.1 Introduction and overview	483
10.2 ML_IPOImpactControl	484
10.3 ML_IPOImpactControl Methods	485
10.3.1 mSetNewState	485
10.3.2 mGetNewState	485
10.3.3 mSetDirection	485
10.3.4 mGetDirection	486
10.3.5 mSetAcceleration	486
10.3.6 mGetAcceleration	487
10.3.7 mSetBreakBehaviour	487
10.3.8 mGetBreakBehaviour	487
10.3.9 mSetPosition	488
10.3.10 mGetPosition	488
10.3.11 mSetValidValues	488
10.3.12 mGetValidValues	488
10.3.13 mSetVelocity	489
10.3.14 mGetVelocity	489
10.3.15 mGetAxisControlledBy	489
10.4 Using methods	489
11 ML_ParamBase.library	493
11.1 Introduction and overview	493
11.2 Function blocks	494
11.2.1 PARAM_Parametersystem	494
Initialization	494

	Page
Registering parameters.....	494
Functionalities.....	494
General information.....	494
Language selection.....	494
Reading parameters.....	495
Reading parameters.....	495
Backing up parameters.....	495
Restoring parameters.....	496
Loading default values.....	496
Creating parameter lists.....	497
Checksum of the parameters to be backed up.....	498
Synchronizing parameter values.....	498
Cyclic write protection.....	498
11.2.2 PARAM_SercosParameter	498
General information.....	498
Interfaces.....	499
Usage.....	499
Read interface.....	499
Write interface.....	499
Command interface.....	499
Attributes.....	500
Access methods.....	501
Read data status.....	501
Reading IDN.....	501
Reading name.....	501
Reading attribute.....	501
Reading minimum value.....	501
Reading maximum value.....	502
Reading date.....	502
Writing date.....	502
Administration.....	502
Loading default values.....	502
Parameter type.....	503
Cyclic write protection.....	503
11.2.3 PARAM_ExtSercosParameter.....	503
General information.....	503
Attributes.....	503
Access methods.....	504
Reading date.....	504
Writing date.....	505
Administration.....	505
Loading default values.....	505
Parameter type.....	505
11.2.4 PARAM_CmdExecutor.....	505
Brief description.....	505
Methods.....	505

Table of Contents

	Page
11.3 Command handling.....	506
11.4 Data types.....	507
11.4.1 Language.....	507
11.4.2 Data access.....	508
11.5 Auxiliary functions.....	509
11.6 Constants.....	509
 12 Service and support.....	 513
 Index.....	 515

1 About this documentation

Editions of this documentation

Edition	Release Date	Note
01	2014-06	First edition for MLC/XLC 14VRS
02	2015-09	PLCopen: ML_ControlWriteActualData; ML_Base: ML_AXISDATA_SM__bit_a21.31, ML_Task* ML_PlcOpen: MB_ReadSercosAttribute, execute RMB_PlcOpenFieldbus: MB_ReadSercosAttribute, execute
03	2015-11	ML_Base: Transmission
04	2016-05	Update: MC_MoveRelative
05	2017-03	Update: MB_GearInPos
06	2018-05	Update: - MC_START_MODE, see chapter 3.5.2 "MC_START_MODE" on page 57 - MB_GearInPos, see chapter 6.6.17 "MB_GearInPos" on page 248 - ML_WriteCyclicAnalog, see chapter 6.8.3 "ML_WriteCyclicAnalog" on page 256 - ML_WriteCyclicAnalogCtrl, see chapter 6.9.6 "ML_WriteCyclicAnalogCtrl" on page 271 - Axis types and function blocks supported by these axis types, see chapter 6.13 "Axis types and function blocks supported by these types" on page 355
07	2019-03	New: Library RMB_PLCopenEtherCAT_SoE, see chapter 8 "RMB_PLCopenEtherCAT_SoE.library" on page 429
08	2020-03	Updated to 14V22 P5

Tab. 1-1: Change Record

1.1 Validity of the documentation

Target group This documentation is intended to inform on the available PLCopen libraries of the IndraLogic XLC and IndraMotion MLC.

Required qualification: Individual who is able to assess the tasks assigned and to identify possible safety risks owing to qualification in the subject, knowledge and experience. The individual should also be familiar with the standards and regulations.

This documentation describes the function blocks, functions, interfaces and data types of the libraries in addition to the function packages required.

Library	Function package	IndraLogic XLC	IndraMotion MLC
RIL_CommonTypes.library, page 25 , data types for general use in all IndraWorks systems	PLC	x	x
ML_Base.library, page 71 , data types of the IndraLogic XLC and IndraMotion MLC systems for Sercos-based axes	PLC	x	x
ML_PLCopen.library, page 171 , PLCopen function blocks, functions and data types for Sercos-based axes	PLC/MOT	"POUs/Parameter" folder	x

Library	Function package	IndraLogic XLC	IndraMotion MLC
RMB_PLCopenFieldBus.library , page 361, PLCopen function blocks, functions and data types for field bus-based drives	PLC	x	x
ML_ControllerBase.library , page 473 allows to implement a controller into the PLC based on modules.	MOT		x
ML_IPOImpactControl.library , page 483, implementing the alternating control. A functionality allowing direct access to the interpolation of the MLC control.	MOT		x
ML_ParamBase.library , page 493, allows to equip the controller axis with an extended parameterability	MOT		x

PLC

Basic function package

MOT

Motion function package

Tab. 1-2:

Target systems supported by the PLCopen libraries

The documentation also contains

- [Standard interfaces at function blocks, page 11](#)
- [Overview: Diagnostics, page 357..](#)

1.2 Using safety instructions

1.2.1 Structure of the safety instructions

The safety instructions are structured as follows:

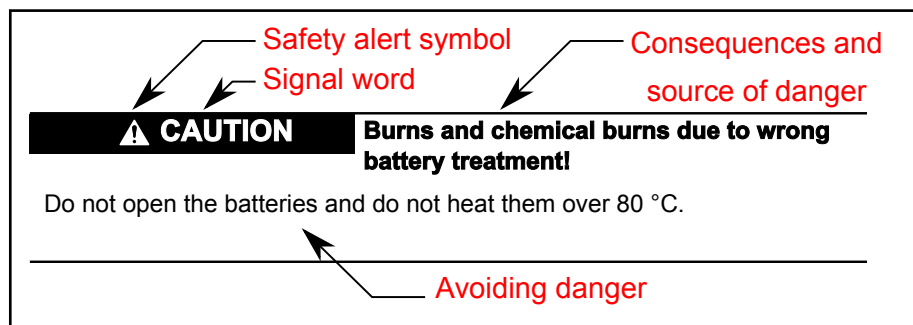


Fig. 1-1: Structure of the safety instructions

1.2.2 Explaining signal words and safety alert symbol

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

The signal word 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.



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.

1.2.3 Symbols used

Pointers are displayed as follows:



This is a note.

Tips are displayed as follows:



This is a tip.

1.2.4 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.

1.3 Required and supplementing documentation XLC/MLC

Documentation titles with type codes and part numbers

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

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

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

Tab. 1-4: XCL/MLC documentation overview - First steps

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

Tab. 1-5: XCL/MLC documentation overview - Engineering

Diagnostics and service
IndraLogic XLC IndraMotion MLC 14VRS Diagnostics DOK-XLCMLC-DIAG***V14-RERS-EN-P, R911341481 This documentation includes all control diagnostics implemented in the control systems IndraLogic XLC and IndraMotion MLC.
IndraLogic XLC IndraMotion MLC 14VRS Parameters DOK-XLCMLC-PARAM***V14-RERS-EN-P, R911341479 This documentation describes the parameters of the XLC/MLC systems as well as the interaction between parameterization and programming. It includes the axis parameters, control parameters, kinematic parameters, touch probe parameters and programmable limit switch parameters.
IndraLogic XLC IndraMotion MLC 14VRS Commissioning DOK-XLCMLC-STARTUP*V14-CORS-EN-P, R911341502 This documentation describes the steps to commission and service the IndraMotion MLC and IndraLogic XLC systems. It includes checklists for frequent tasks and a detailed description of the steps.
WebAssistant DOK-GENERL-WEB*ASSIST*-APRS-EN-P, R911381469 This documentation describes the WebAssistant. The WebAssistant is a webbased diagnostic tool used to access a control system via an Ethernet highspeed connection. The WebAssistant allows OEMs, end users and service engineers to access and to remotely diagnose a system. A PC with Internet Explorer version 11 or higher, Firefox version 45 or above or Google Chrome version 54 or above is required. The following control variants are supported: • IndraMotion MLC L45/L65/L75/XM21/XM22/XM42/VPx • IndraLogic XLC L45/L65/L75/VEP
IndraWorks IndraMotion Service Tool DOK-IWORKS-IMST*****-APRS-EN-P, R911341383 This documentation describes the IndraMotion Service Tool (IMST). IMST is a web-based diagnostic tool used to access a control system via an Ethernet high-speed connection. The IMST allows OEMs, end users and service engineers to access and remotely diagnose a system. The PC has to use at least Internet Explorer 8, Firefox 3.5 or a higher version. The following control variants are supported: • IndraMotion MLC L45/L65/L75/XM21/XM22/XM42/VPx • IndraLogic XLC L45/L65/L75

Tab. 1-6: XCL/MLC documentation overview - Diagnostics and service

Firmware
Rexroth IndraDrive MPx-20, Functions ¹⁾ DOK-INDRV*-MP*-20VRS**-AP01-EN-P, R911345608 This documentation describes all functional properties of the IndraDrive firmware in the variants MPB-20, MPM-20, MPC-20 and MPE-20.
Rexroth IndraDrive MPx-16 to MPx-20 und PSB, Parameters DOK-INDRV*-GEN1-PARA**-RERS-EN-P, R911328651 This documentation contains the descriptions of all parameters implemented in the firmware for drive controllers of the IndraDrive range. It supports the drive controller parameterization.
Rexroth IndraDrive MPx-16 to MPx-20 and PSB, Diagnostic Messages DOK-INDRV*-GEN1-DIAG**-RERS-EN-P, R911326738 This documentation contains the descriptions of all diagnostic messages implemented in the following firmware products: • Drive controller firmware variants MPx-16 to MPx-20 • PSB firmware variants for "HMU05" and "KMV03" type supply units • Firmware variants for "HMF01" type supply units It assists machine operators and installation programmers with troubleshooting.
Hardware
Rexroth IndraDrive Control Sections CSB02, CSE02, CSH02, CDB02 DOK-INDRV*-Cxx02*****-PRRS-EN-P, R911338962
Rexroth IndraDrive Cs Drive Systems with HCS01 DOK-INDRV*-HCS01*****-PRRS-EN-P, R911322210
Rexroth IndraDrive Mi Drive Systems with KCU02, KSM02, KMS02/03, KMV03 DOK-INDRV*-KCU02+KSM02-PRxx-EN-P, R911335703
Rexroth IndraDrive ML Drive Systems with HMU05 DOK-INDRV*-Hxx05*****-PRxx-EN-P, R911344279

1)

In preparation

Tab. 1-7:

Overview of documentations for drive controllers

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

Tab. 1-8:

XCL/MLC documentation overview - PLC

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

xx Corresponding edition
Tab. 1-9: XCL/MLC documentation overview - Field buses

HMI
IndraWorks 14VRS HMI DOK-IWORKS-HMI*****V14-APRS-EN-P, R911343569 This documentation describes the functions, configuration and operation of the user interfaces IndraWorks HMI Engineering and IndraWorks HMI Operation.
IndraWorks 14VRS WinStudio 7.4 DOK-IWORKS-WINSTUD*V14-APRS-EN-P, R911341585 This "User Manual and Technical Reference Book" facilitates working with the "Rexroth WinStudio" software for optimal results. This document provides technical information and step-by-step instructions to create web-enabled HMI/SCADA programs.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS HMI Connection DOK-XLCMLC-HMI*****V14-APRS-EN-P, R911341497 This documentation describes the visualization systems supported by the IndraLogic XLC and IndraMotion MLC and their connection.

Tab. 1-10: XCL/MLC documentation overview - HMI

Technology
IndraLogic XLC IndraMotion MLC 14VRS Generic Application Template DOK-XLCMLC-TF*GAT**V14-APRS-EN-P, R911341487 This documentation provides a structured template to the IndraLogic PLC programmer. This template can be used to add and edit the PLC programming code. It includes the template, the template wizard and example applications.
IndraMotion MLC 14VRS Technology Libraries DOK-MLC***-TF*LIB**V14-LIRS-EN-P, R911341511 This documentation describes the function blocks, functions and data types of the "ML_TechInterface.library", "ML_TechMotion.library", "RMB_TechCam.library" and "ML_TechBase.library". It also includes libraries for the winder functionality, register controller functionality and CrossCutter functionality.
IndraWorks 14VRS Energy Efficiency Management DOK-IWORKS-4EE*****V14-APRS-EN-P, R911339229 This documentation describes the HMI connection of the energy management system and the function blocks, functions and data types of the "RIL_4EE " library. The documentation also includes the error reactions of the function blocks.
Rexroth IndraMotion MLC 14VRS RegisterControl (Library) DOK-MLC***-REGI*CO*V14-LIRS-EN-P, R911341509 This documentation describes the inputs and outputs of the individual function blocks and provides notes on their use.
IndraMotion MLC 14VRS RegisterControl (Application Description) DOK-MLC***-REGI*CO*V14-APRS-EN-P, R911341507 This documentation describes the application of the integrated register control for a rotogravure printing machine. The components of the mark stream sensor, the HMI application and the error recovery options are described. This instruction provides information on how to operate the register control, react to errors and query diagnostics. This documentation is written for machine setters and machine operators.
Rexroth IndraMotion MLC 14VRS Winder Function Application DOK-MLC***-TF*WIND*V14-APRS-EN-P, R911341513 This application-related system documentation describes the application of the winder technology functions.
Rexroth IndraWorks CamBuilder 13VRS DOK-IWORKS-CAMBUIL*V13-APRS-EN-P, R911336291 This documentation describes the basic principles and operation of the CamBuilder, the cam editing tool.
IndraMotion MLC 14VRS Robot Control V2 DOK-MLC***-ROCO****V14-RERS-EN-P, R911341588 This documentation provides information about the Robot Control V2. The focus is on PLCopen programming. The program structure, variables, functions, motion statements and the required system parameters are described. This documentation also includes the description of the "ML_Robot" library. Robot Control V2 is supported by a kinematic interface. The components of this interface are described in the "ML_KinTech" library.

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

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

Tab. 1-12: XCL/MLC documentation overview - Open Core Engineering - OCE

SafeLogic
IndraWorks SafeLogic 14VRS First Steps DOK-IWORKS-SL*STEP*V14-CORS-EN-P, R911341520 This documentation describes the initial commissioning of the Safety function module and its external Safety periphery. Taking the example of a project, all required steps to commission a simple Safety application are executed.
IndraLogic XLC IndraMotion MLC 14VRS SafeLogic System Overview DOK-XLCMLC-SL**SYS*V14-PRRS-EN-P, R911341696 This documentation describes the project planning, configuration, creation and commissioning of safety-related devices.
IndraWorks 14VRS SafeLogic - Project Configuration, Version 1.2.0.0. DOK-IWORKS-SL**PRJ*V14-APRS-EN-P, R911341694 This documentation describes the configuration of Safety applications.
IndraControl CFL01.1-F1 Safety Function Module DOK-CTRL-SL**FM*****-ITRS-EN-P, R911336576 This documentation is provided with the hardware and describes the commissioning, the installation, the decommissioning and disposal of the hardware.

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

Hardware
Rexroth IndraControl L45/L65/L85 Control DOK-CTRL-ICL45L65L85-PRxx-EN-P, R911332116 This documentation describes the IndraControl L45/L65/L85 controls.
Rexroth IndraControl Lxx 14VRS Function Modules DOK-CTRL-FM*LXX**V14-APRS-EN-P, R911341583 This documentation describes all function modules of the Lxx controls including engineering and diagnostics.
IndraControl XM21, XM22 Controls DOK-CTRL-IC*XM2*****-ITRS-EN-P, R911340667 This documentation describes the IndraControl XM21/XM22 controls.
IndraControl XM42 Controls DOK-CTRL-XM4X*CTRL**-ITRS-EN-P, R911345566 This documentation describes the IndraControl XM42 controls.
IndraControl XFE 01.1 Extension Modules Profibus, RT-Ethernet, Sercos, CAN DOK-CTRL-XFE**EXTMOD-ITRS-EN-P, R911345570 This documentation describes all extension modules of the XM controls.

xx Corresponding edition
Tab. 1-14: XCL/MLC documentation overview - Hardware

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

Tab. 1-15: XCL/MLC documentation overview - Hydraulics

2 Standard interfaces at function blocks

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.

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_WriteParameter
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. 2-1: Overview on function block types

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. 2-2: Overview on standardized identifiers

2.4 Inputs and outputs of state-controlled function blocks, final processing

Control/ processing	I/O	Variable name	Description
State-con- trolled/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 order and is then in a final state. Data outputs are valid now. "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 processing its actual task and is then in an intermediate state. Possible preprocessing are not marked 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 has been interrupted and is now 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 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 interrupted due to an error and is now in a final state. "Error" remains TRUE until the control input "Enable" is disabled. An additional reset input is 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. 2-3: 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).

Standard interfaces at function blocks

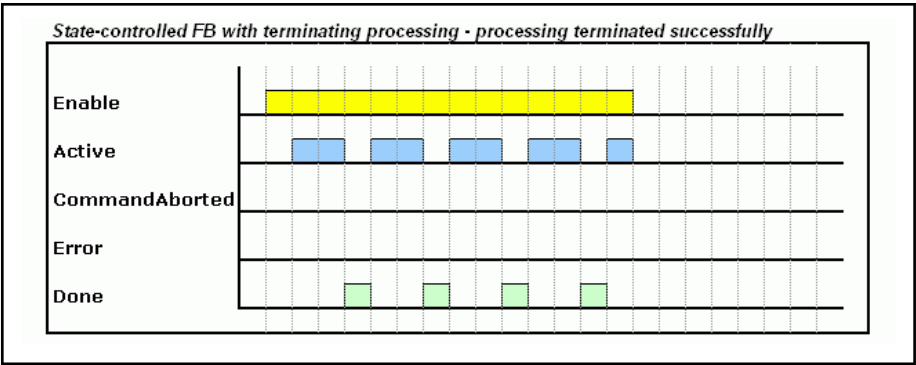


Fig. 2-1: I/O of state-controlled function blocks, final processing without "Shut-down". Processing successfully completed

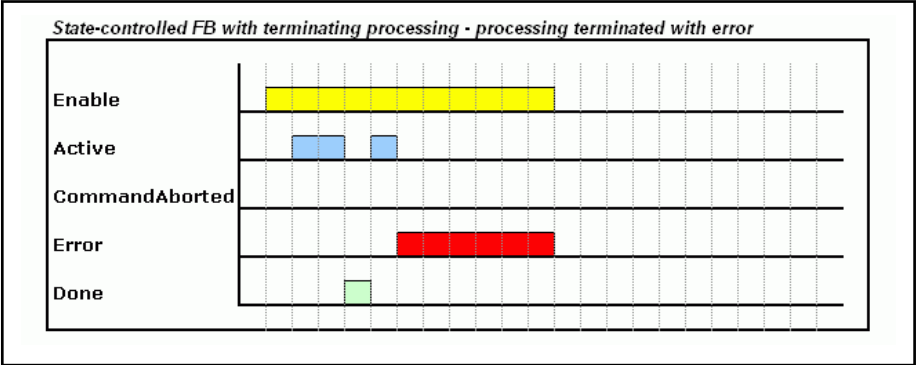


Fig. 2-2: I/O of state-controlled function blocks, final processing without "Shut-down". Processing completed with error

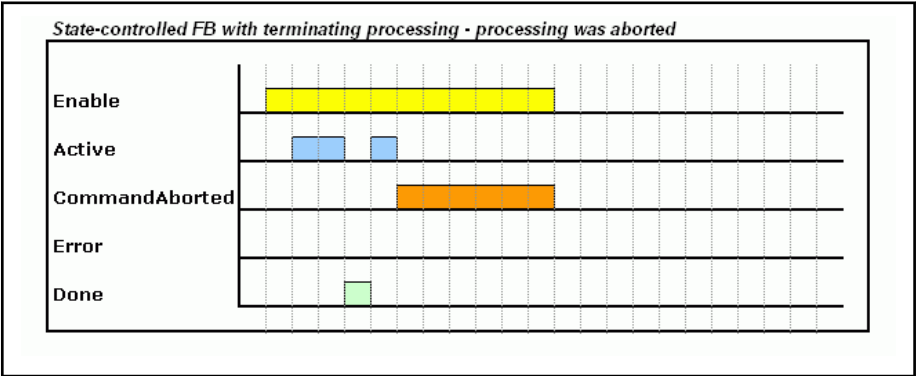


Fig. 2-3: I/O of state-controlled function blocks, final processing without "Shut-down". Processing interrupted

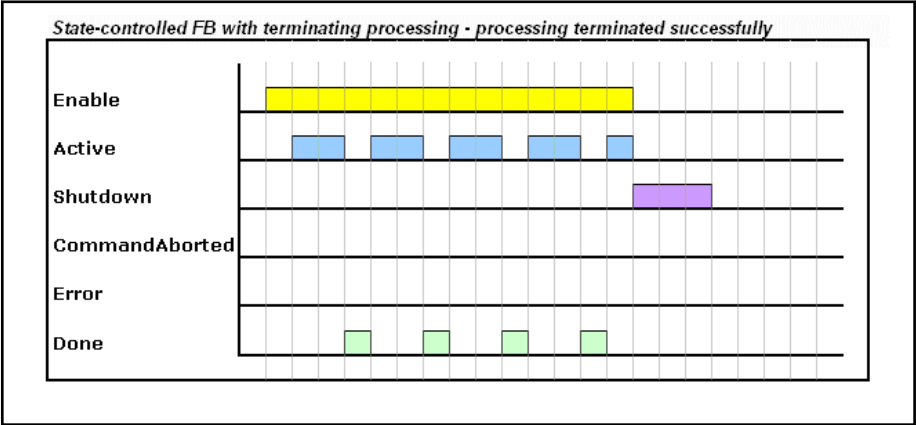


Fig. 2-4: I/O of state-controlled function blocks, final processing with "Shut-down". Processing successfully completed

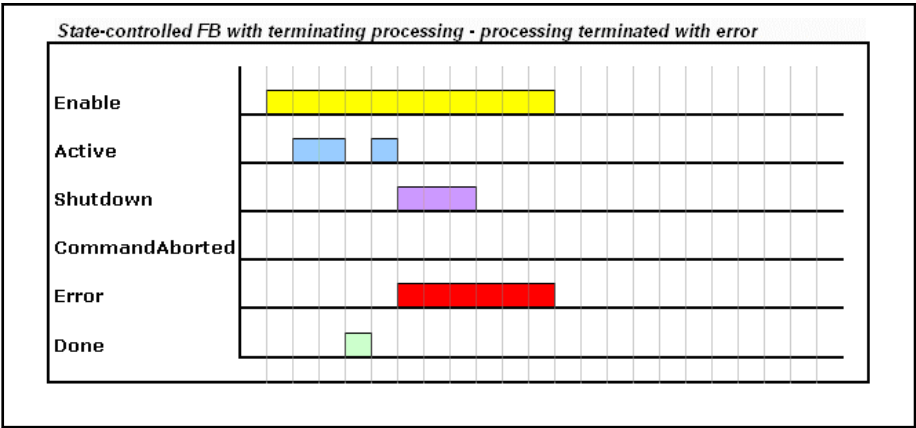


Fig. 2-5: I/O of state-controlled function blocks, final processing with "Shutdown". Processing completed with error

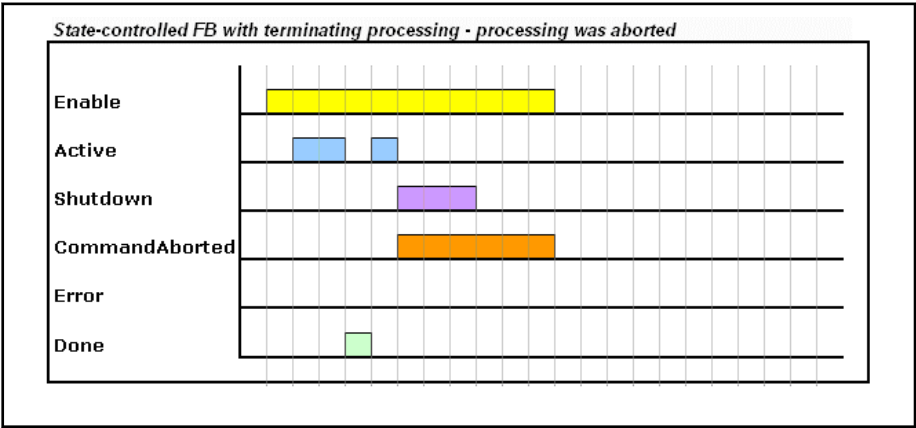


Fig. 2-6: I/O of state-controlled function blocks, final processing with "Shutdown". Processing was interrupted

2.5 Inputs and outputs of edge-controlled function blocks, final processing

Control/ processing	I/O	Name	Description
Edge-con- trolled/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 order and is then in a final state. Data outputs are valid now. "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 processing its actual task and is then in an intermediate state. Possible preprocessing are not marked 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 has been interrupted and is now 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. 2-4: 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).

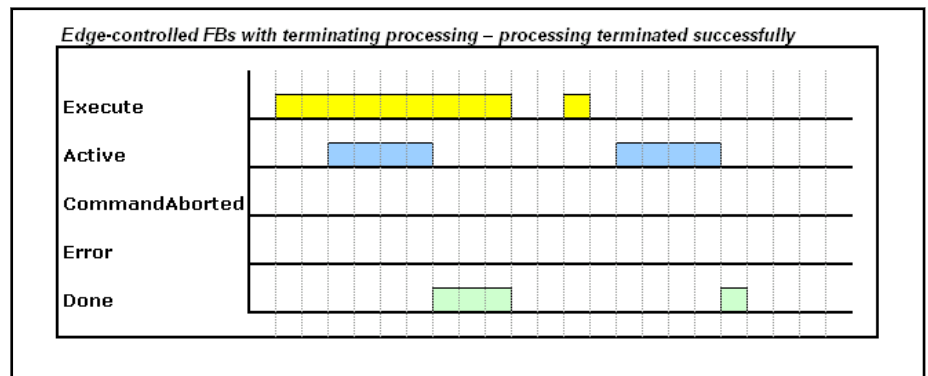


Fig. 2-7: I/O of edge-controlled function blocks, final processing without "Shutdown". Processing successfully completed

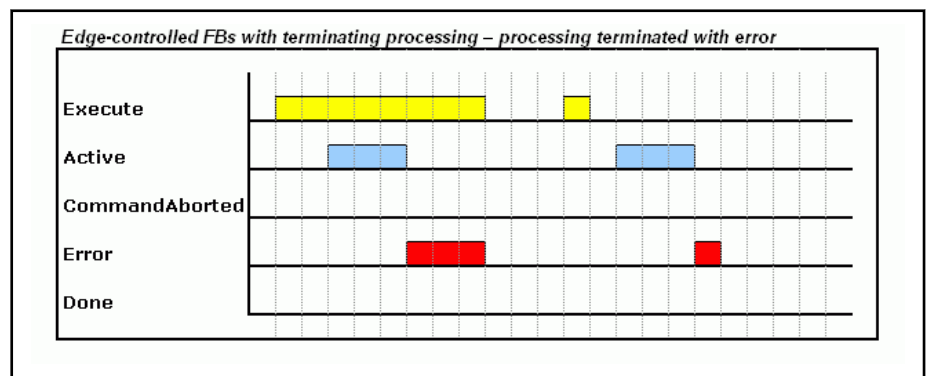


Fig. 2-8: I/O of edge-controlled function blocks, final processing without "Shutdown". Processing completed with error

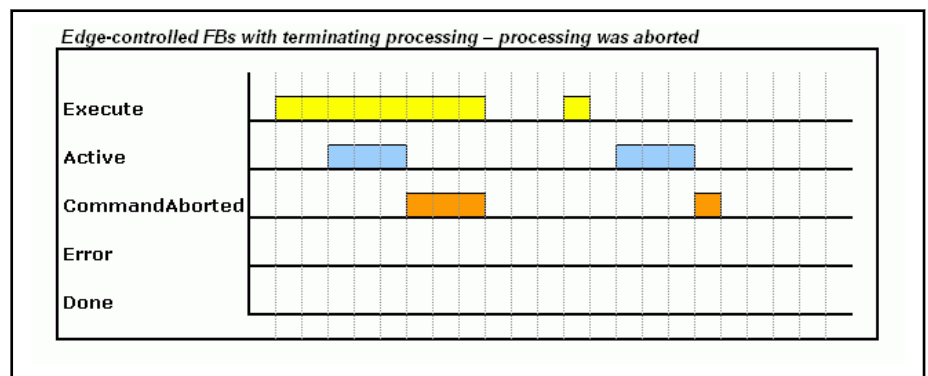


Fig. 2-9: I/O of edge-controlled function blocks, final processing without "Shutdown". Processing interrupted

Standard interfaces at function blocks

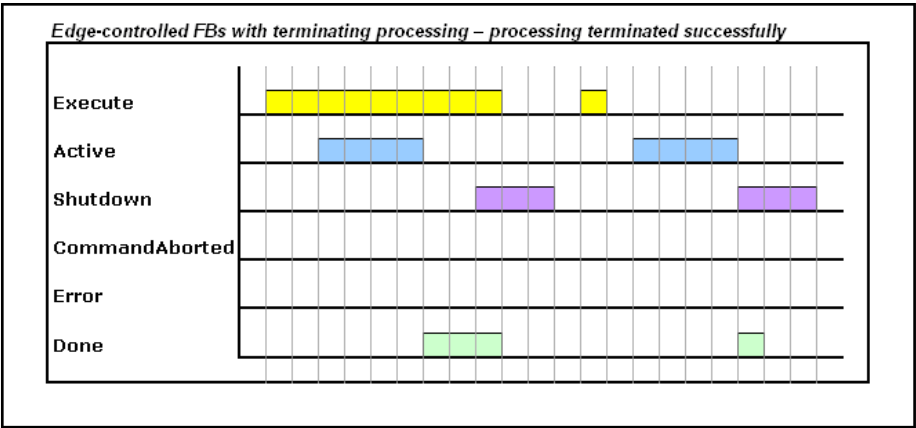


Fig. 2-10: I/O of edge-controlled function blocks, final processing with "Shutdown". Processing successfully completed

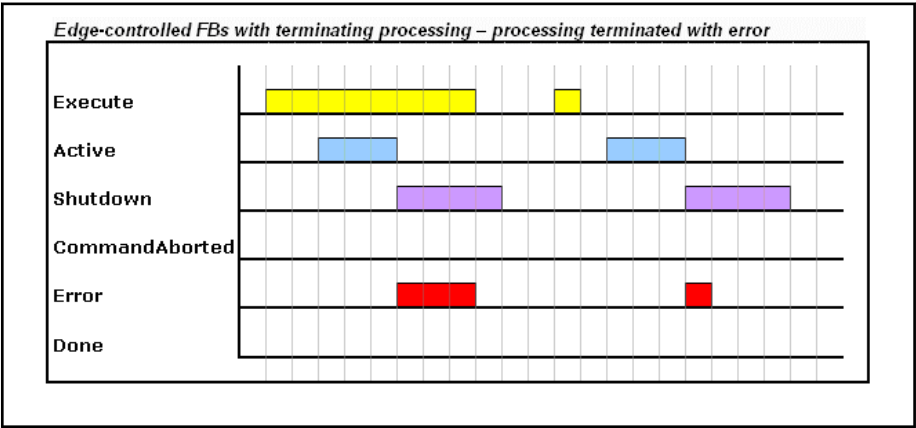


Fig. 2-11: I/O of edge-controlled function blocks, final processing with "Shutdown". Processing completed with error

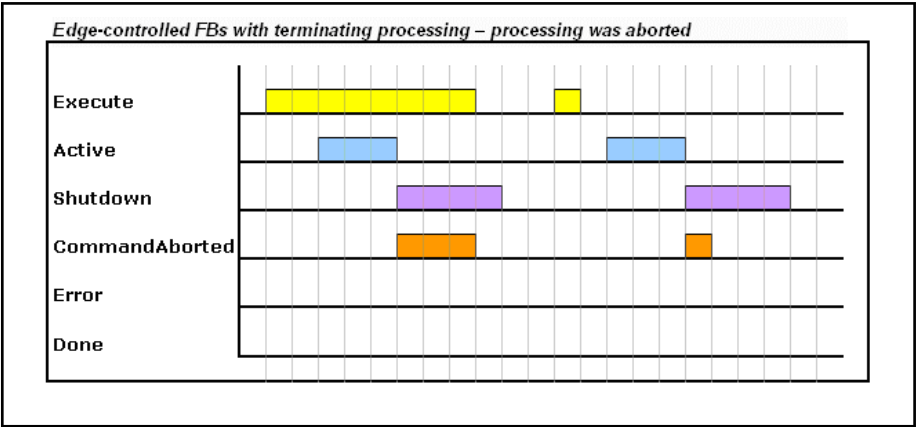


Fig. 2-12: I/O of edge-controlled function blocks, final processing with "Shutdown". Processing interrupted

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...." Is TRUE, the function block reached its target, but it is still in intervention to "keep" the achieved. Thus, it is thus in a permanent final state. Data outputs are valid now. 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 in intervention permanently, "Active" remains TRUE until the function block is switched off via the control input "Enable" or until it is completed by "Error" or "CommandAborted". As long as "Active" is TRUE, "Error" or "CommandAborted" have to 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 has been interrupted and is now 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 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 has been interrupted due to an error and is now in a final state. "Error" remains TRUE until the control input "Enable" is disabled. An additional reset input is 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. 2-5: 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).

Standard interfaces at function blocks

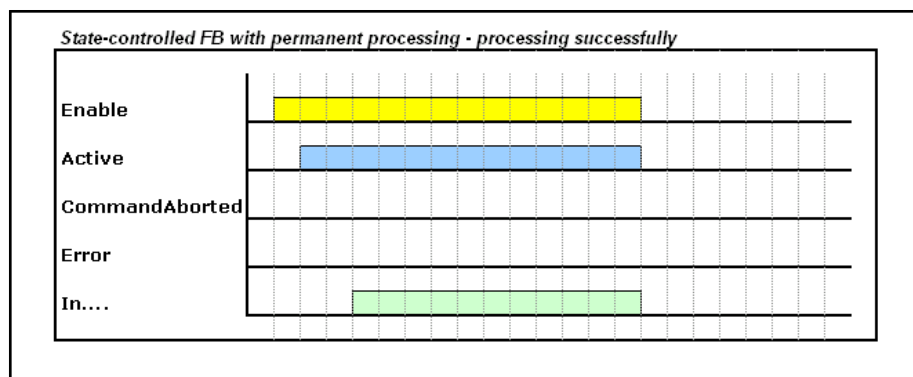


Fig. 2-13: I/O of state-controlled function blocks, permanent processing without "Shutdown". Processing successful

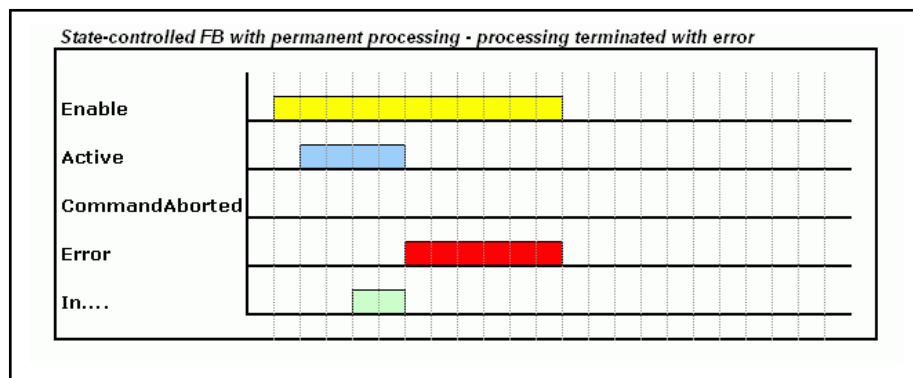


Fig. 2-14: I/O of state-controlled function blocks, permanent processing without "Shutdown". Processing aborted with errors

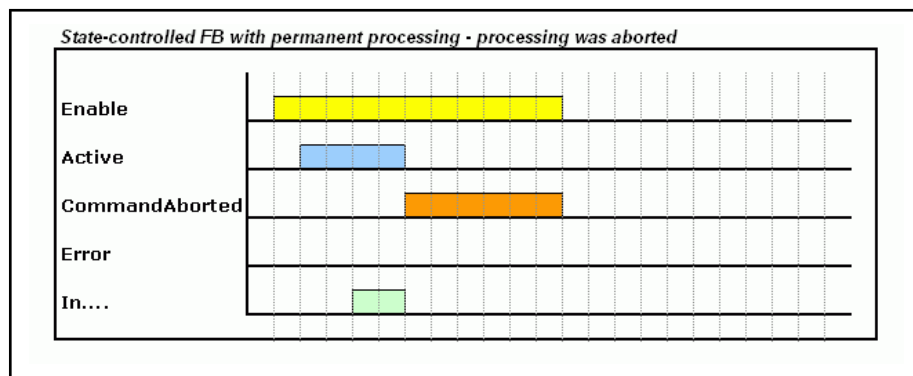


Fig. 2-15: I/O of state-controlled function blocks, permanent processing without "Shutdown". Processing interrupted

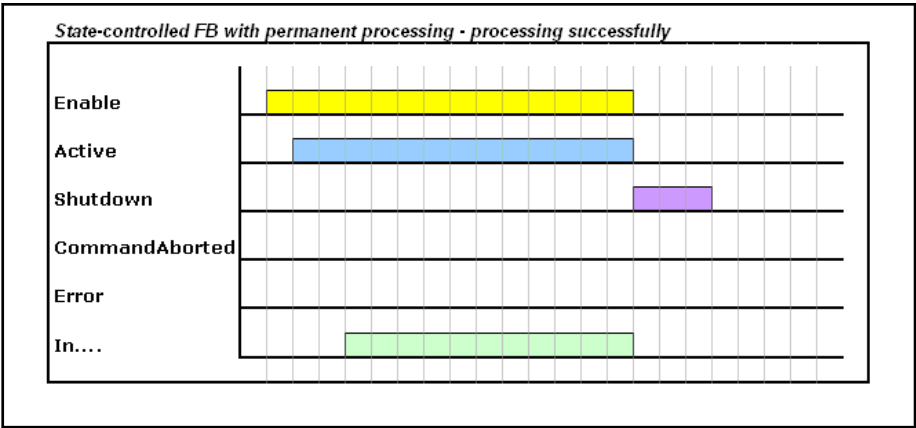


Fig. 2-16: I/O of state-controlled function blocks, permanent processing with "Shutdown". Processing successful

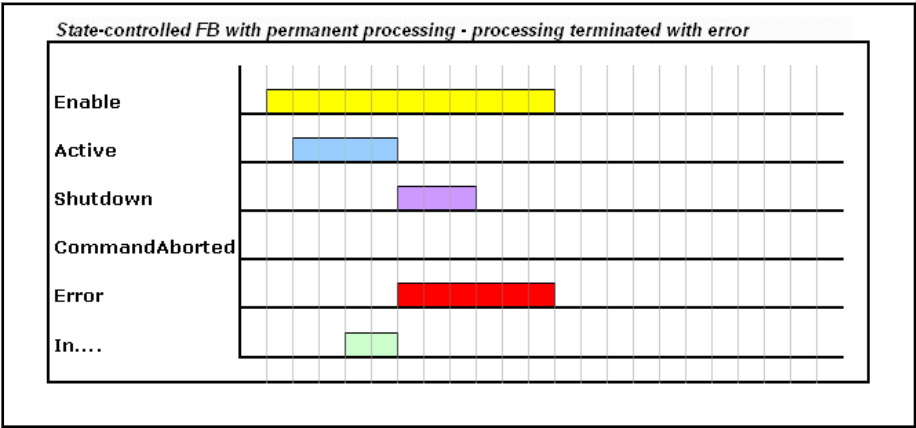


Fig. 2-17: I/O of state-controlled function blocks, permanent processing with "Shutdown". Processing aborted with errors

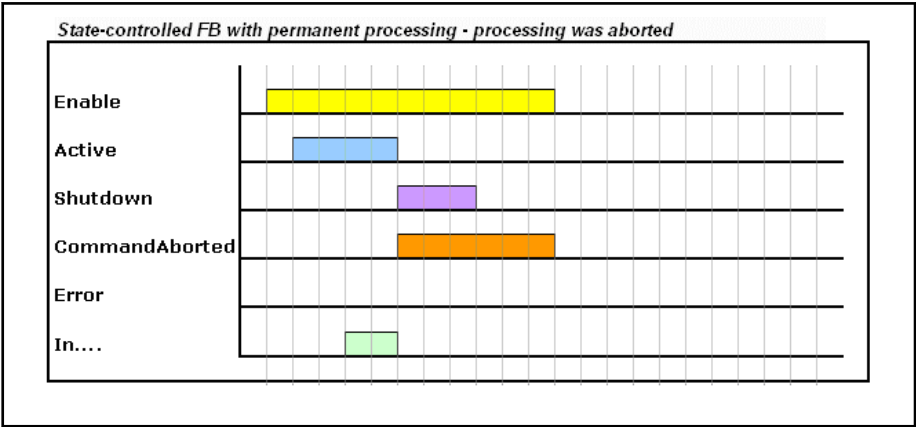


Fig. 2-18: I/O of state-controlled function blocks, permanent processing with "Shutdown". Processing interrupted

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...." is TRUE, the function block reached its goal but is still in intervention to "retain" the achieved and it is thus in a permanent final state. Data outputs are valid now. Since the function block remains in intervention, "Active" also has to remain TRUE. This even applies when "In...." decreases again. "Error" and "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 pre-processing 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 has been interrupted and is now 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. 2-6: 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).

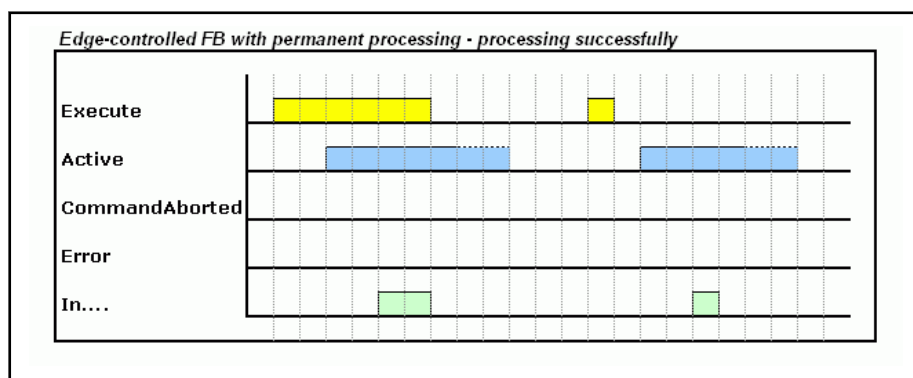


Fig. 2-19: I/O of edge-controlled function blocks, permanent processing without "Shutdown". Processing successful

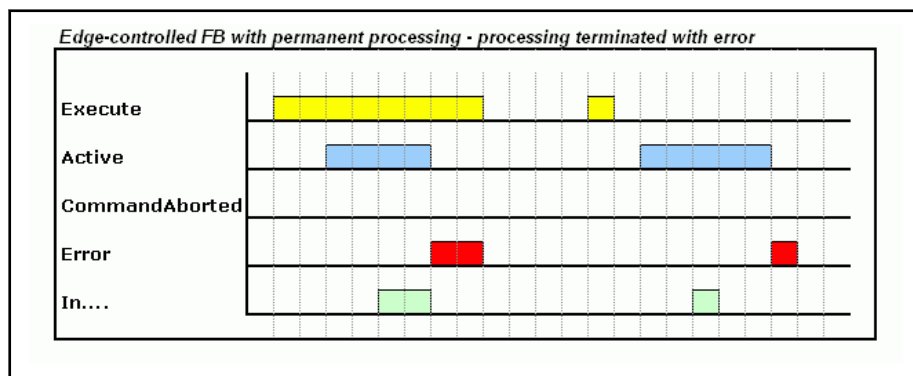


Fig. 2-20: I/O of edge-controlled function blocks, permanent processing without "Shutdown". Processing aborted with errors

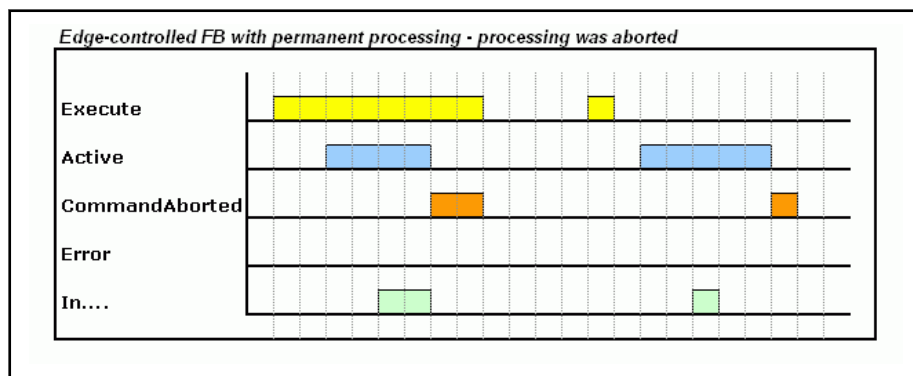


Fig. 2-21: I/O of edge-controlled function blocks, permanent processing without "Shutdown". Processing interrupted

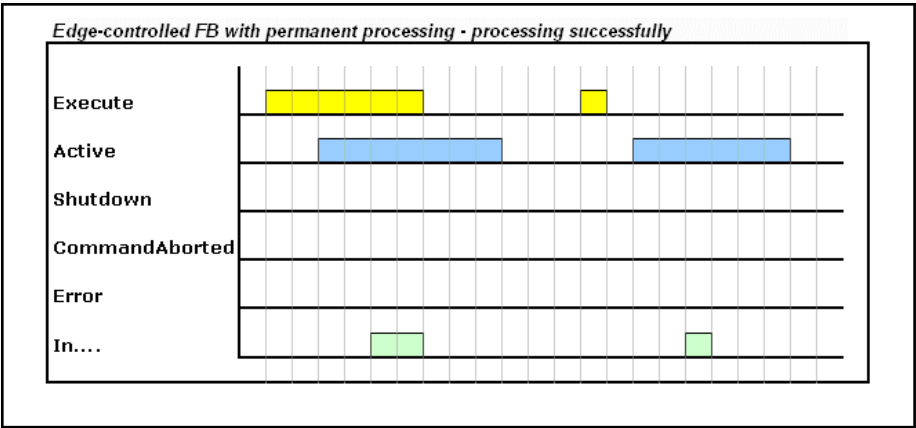


Fig. 2-22: I/O of edge-controlled function blocks, permanent processing with "Shutdown". Processing successful

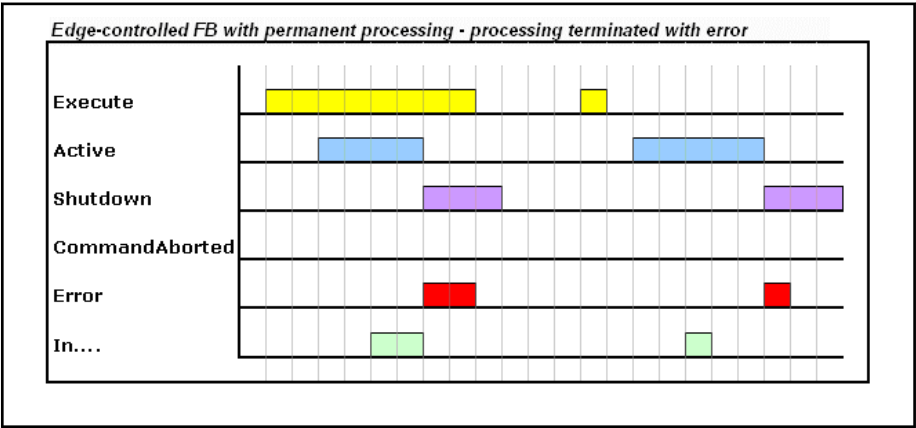


Fig. 2-23: I/O of edge-controlled function blocks, permanent processing with "Shutdown". Processing aborted with errors

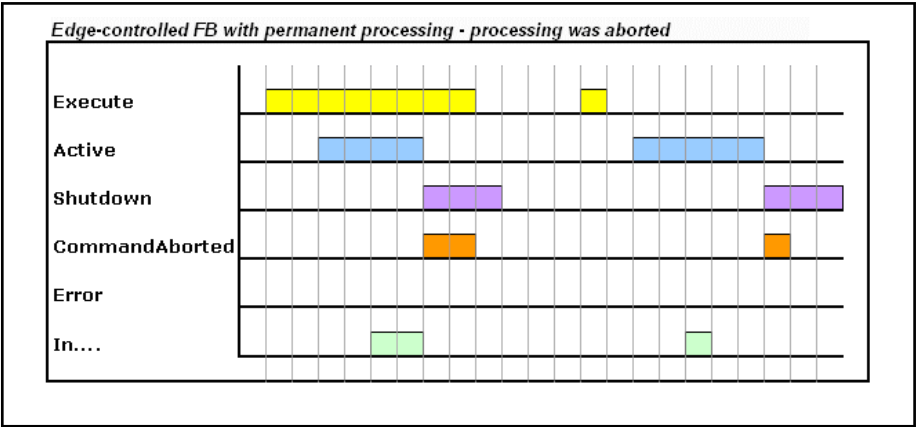


Fig. 2-24: I/O of edge-controlled function blocks, permanent processing with "Shutdown". Processing interrupted

3 RIL_CommonTypes.library

3.1 Introduction and overview



The basic function package PLC is sufficient for the library of the IndraLogic XLC/IndraMotion MLC 14VRS system.

The RIL_CommonTypes library contains the data types which describe the input and output variables of the Bosch Rexroth function blocks.

Library structure:

Global_Variables

Global_Variables

- [IL_NULL](#), page 26, NIL pointer.

SercosAttribute

- The Global Variable List contains the Sercos attributes, including a comment on their use.

Data types

Data types of logical addresses

- [CONTROLS](#), page 27, (ENUM)
- [KIN_OBJECTS](#), page 30, (ENUM)
- [OBJECTS](#), page 28, (ENUM)
- [TP_OBJECTS](#), page 31, (ENUM)

PLCopen data types

- [AXIS_GROUP_REF](#), page 29, (STRUCT),
- [AXIS_REF](#), page 27, (STRUCT)
- [MC_CAM_ID](#), page 32, (ENUM),
- [MC_START_MODE](#), page 57, (ENUM)
- [MC_STEP_MODE](#), [MotionProfile extended to FlexProfile](#) , page 32, (ENUM)
- [MC_SYNC_DIRECTION](#), page 59, (ENUM)
- [TOUCHPROBE_REF](#), page 31, (STRUCT)

Data types of POU diagnostics

- [ERROR_STRUCT](#), page 61, (STRUCT)

Includes

- [ERROR_CODE](#), page 60, (ENUM)
- [ERROR_TABLE](#), page 62, (ENUM)

Time data types

- [EXT_SYS_TIME_DATE](#), page 63, (STRUCT)
- [SYS_TIME64](#), page 63, (STRUCT)
- [SYS_TIME_DATE](#), page 64, (STRUCT)

TouchProbe

- [MB_DIFF_EVAL](#), page 65, (ENUM)
- [MB_EXPECT_WINDOW_MODE](#), page 66, (ENUM)
- [MB_INTERPOLATION_MODE](#), page 66, (ENUM)
- [MB_POSITION_EVAL](#), page 67, (ENUM)
- [MB_PROBE_DATA](#), page 68, (ENUM)
- [MB_PROBE_INPUT](#), page 67, (ENUM)
- [MB_PROBE_MODULE](#), page 67, (ENUM)
- [MB_PROBE_CONFIG](#), page 69, (ENUM)

It was declared to define a uniform access to different device types for different control systems.

3.2 Global_Variables

3.2.1 IL_NULL

Brief description A NIL pointer was defined as POINTER TO BYTE.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/Global_Variables

Tab. 3-1: *IL_NULL* reference table

Type definition

```
VAR_GLOBAL CONSTANT
  IL_NULL: POINTER TO BYTE := 0;
END_VAR
```

3.3 Data types to reference objects of Motion function blocks

3.3.1 Referencing objects, introduction and overview

To reference objects (controls, axes, kinematics/axis groups, touch probes, etc.), one REF structure with allocated enumeration types is provided for each object type.

Information is transferred to the respective function block via **VAR_IN_OUT CONSTANT** for 14VRS and higher.

The REF structure contains two 16 bit address elements:

- Control address (control address element (element of [CONTROLS](#), page 27) and
- Object address, element of enumeration type



Information was transferred up to 13VRS via **VAR_IN_OUT**.

In 14VRS and higher, information is transferred via **VAR_IN_OUT CONSTANT**.

REF structure	Enumeration type
AXIS_REF , page 27, axes, currently still programmable limit switches	OBJECTS , page 28

AXIS_GROUP_REF , page 29, axis groups/kinematics	KIN_OBJECTS , page 30
TOUCHPROBE_REF , page 31, touch probe	TP_OBJECTS , page 31

Tab. 3-2: REF structures and their enumeration types



The elements of the enumeration types used for the actual individual case depend on

- System
- Control type/drive type
- Firmware version

3.3.2 CONTROLS

Brief description

The enumeration contains all controls addressed via Motion function blocks in the link:



In case of a physical address, there are two options to go to the "local control": number "0", locally, and via its own address, e.g. CTRL_3.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of logical addresses

Tab. 3-3: CONTROLS reference table

Type definition

```
// Enumeration of valid logical control addresses
TYPE CONTROLS :
(
    LOCAL_CNTRL := 0,    // Local control (default)
    CNTRL_1 := 1,        // Control 1
    CNTRL_2 := 2,        // Control 2
    // ...
    CNTRL_99 := 99       // Control 99
);
END_TYPE
```

Functional description

This list of connected controls contains all controls that can be addressed via Motion function blocks in the link:

- Number 0 is enabled for the local control (control itself),
- The number range 1... 99 is reserved for all connected controls.

3.3.3 AXIS_REF

Brief description

The structure contains information on the desired control and on the object controlled by this control (axis, programmable limit switch, oscilloscope,...).

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of PLCopen

Tab. 3-4: AXIS_REF reference table

Type definition

```

(* Structure of PLCopen element AXIS_REF. The AXIS_REF is a structure that contains *)
(* information on the corresponding Object. *)
(* It is used as a VAR_IN_OUT CONSTANT in all Motion Control Function Blocks. *)
TYPE AXIS_REF :
STRUCT
  CntrlNo: CONTROLS := LOCAL_CNTRL; (* control reference number *)
                                     (* Steuerungsnummer, Init: lokale Steuerung *)
  AxisNo: OBJECTS := NO_OBJECT; (* object reference number *)
                                   (* Achsreferenznummer, Init: kein Objekt *)
END_STRUCT
END_TYPE

```

Functional description

Information is transferred to the respective PLCopen function block via **VAR_IN_OUT CONSTANT** for 14VRS and higher.

The AXIS_REF structure contains two 16 bit address elements:

- Control address (element of [CONTROLS](#), page 27)
- Object address (element of [OBJECTS](#), page 28)

The elements can be addressed via logical or physical address. In case of a logical address, the element points to a parameter structure with a physical address (e.g. Ethernet address of the control, Sercos address of the drive or similar).



Information was transferred up to 13VRS via **VAR_IN_OUT**.

In 14VRS and higher, information is transferred via **VAR_IN_OUT CONSTANT**.

3.3.4 OBJECTS

Brief description

This enumeration of available axes contains all the objects of a link addressed via "AXIS_REF" using Motion function blocks.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of logical addresses

Tab. 3-5: *OBJECTS reference table*

Type definition

```

// Enumeration of valid logical object addresses
TYPE OBJECTS :
(
  // not specified object
  NO_OBJECT := 0, // no object specified
  // drives 0001..0999
  AXIS_1 := 1, // drive 1
  // ...
  AXIS_99 := 99, // drive 99
  AXIS_100 := 100, // drive 100
  // ...
  AXIS_512 := 512, // drive 512

  RMA_1 := 1001, (* real master axis 1 *)
  (* RMA_2 := 1002, (* real master axis 2 *)

  VMA_1 := 2001, (* virtual master axis 1 *)
  VMA_2 := 2002, (* virtual master axis 2 *)
  VMA_INT := 2003, (* internal virtual master axis *)
  VMA_EXT := 2004, (* external virtual master axis *)
  MA_LINK_1 := 2005, (* link axis 1 *)
  MA_LINK_2 := 2006, (* link axis 2 *)

  // ELS Group Local Master 3001..3999
  GLMA_1 := 3001, // ELS Group Local Master Axis 1
  // ...

```

```

    GLMA_8 := 3008,           // ELS Group Local Master Axis 8
// ELS Groups 4001..4008
    ELSG_1 := 4001,           // ELS Group 1
    // ...
    ELSG_8 := 4008,           // ELS Group 8
// ELS System Masters 5001..5006
    ELSM_1 := 5001,           // ELS System Master 1
    // ...
    ELSM_6 := 5006,           // ELS System Master 6
// FieldBus Drives 6001..6099
    FieldBusAxis_1 := 6001,    //Fieldbus Drive 1
    // ...
    FieldBusAxis_99 := 6099,    //Fieldbus Drive 99
// Program Limit Switch 8001..8999
    PLS_1 := 8001,             //Program Limit Switch unit 1
    // ...
    PLS_4 := 8004,             //Program Limit Switch unit 4
// Oscilloscope 9001..9999
    OSCI_CHANNEL_1 :=9001,      // Oscilloscope Channel 1
    // ...
    OSCI_CHANNEL_128 :=9128     // Oscilloscope Channel 128
// Sercos address 10001..10512
    SERCOS_ADDRESS_1 :=10001,    // Sercos address 1
    // ...
    SERCOS_ADDRESS_512 :=10512,  // Sercos address 512
// Sercos topology 11001..11512
    SERCOS_TOPOLOGY_1 :=11001,   // Sercos topology 1
    // ...
    SERCOS_TOPOLOGY_512 :=11512, // Sercos topology 512
// Logical drive 12001..12512 with native operation data *)
    AXIS_NATIVE_1 := 12001,      // Logical drive 1 with native operation data
    // ...
    AXIS_NATIVE_512 := 12512     // Logical drive 512 with native operation data
);
END_TYPE

```



Number "00", "NO_OBJECT" (no object specified) is used for de-activated axes.



Name and numbers are assigned when the respective axis/the object is created in the IndraWorks Suite (IndraLogic, Global-MlcVariables).

3.3.5 AXIS_GROUP_REF

Brief description The structure contains information on the desired control and the corresponding axis group.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of PLCopen

Tab. 3-6: *AXIS_GROUP_REF* reference table

Type definition

```

// Structure of PLCopen element AXIS_GROUP_REF.
// The AXIS_GROUP_REF is a structure that Contains information of the axis-group.
// It is used as a VAR_IN_OUT CONSTANT in Motion Control Function Blocks.
TYPE AXIS_GROUP_REF:
STRUCT
    CntrlNo: CONTROLS := LOCAL_CNTRL; // control reference number
                                           // Steuerungsnummer, Init: lokale Steuerung

```

```

GroupNo: KIN_OBJECTS := NO_KINEMATICS; // object reference number
// Achsgruppenreferenznummer, Init: NO_KINEMATICS
END_STRUCT
END_TYPE

```

Functional description

Information is transferred to the respective function block via **VAR_IN_OUT CONSTANT** for 14VRS and higher.

The **AXIS_GROUP_REF** structure contains two 16 bit address elements:

- Control address (element of [CONTROLS](#), page 27)
- Object address (element of [KIN_OBJECTS](#), page 30)

The elements can be addressed via logical or physical address. In case of a logical address, the element points to a parameter structure with a physical address (e.g. Ethernet address of the control).



Information was transferred up to 13VRS via **VAR_IN_OUT**.

In 14VRS and higher, information is transferred via **VAR_IN_OUT CONSTANT**.

3.3.6 KIN_OBJECTS

Brief description

The enumeration of available kinematics (axis groups) contains all objects of a Motion link accessed by Coordinated Motion function blocks, also refer to ([AXIS_GROUP_REF](#), page 29).

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of logical addresses

Tab. 3-7: *KIN_OBJECTS* reference table

Type definition

```

TYPE KIN_OBJECTS :
(
  (* not specified object
    NO_KINEMATICS :=0,
  (* axis groups 1..16 *)
    KINEMATICS_1 := 01,
    KINEMATICS_2 := 02,
    KINEMATICS_3 := 03,
    KINEMATICS_4 := 04,
    KINEMATICS_5 := 05,
    KINEMATICS_6 := 06,
    KINEMATICS_7 := 07,
    KINEMATICS_8 := 08,
    KINEMATICS_9 := 09,
    KINEMATICS_10 := 10,
    KINEMATICS_11 := 11,
    KINEMATICS_12 := 12,
    KINEMATICS_13 := 13,
    KINEMATICS_14 := 14,
    KINEMATICS_15 := 15,
    KINEMATICS_16 := 16
  );
END_TYPE

```



Name and numbers are assigned when the object is created in the IndraWorks Suite (IndraLogic, GlobalMlcVariables)



The number of implementable kinematics depends on the control type and the firmware version.

3.3.7 TOUCHPROBE_REF

Brief description The structure contains information on the desired control and on the object controlled by this control (touch probe).

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of PLCopen

Tab. 3-8: TOUCHPROBE_REF reference table

Type definition

```
(* Structure of PLCopen element TOUCHPROBE_REF. The TOUCHPROBE_REF is a structure that *)
(* contains information on the corresponding object. *)

TYPE TOUCHPROBE_REF :
STRUCT
  CntrlNo:      CONTROLS      := LOCAL_CNTRL;  (* control reference number *)
              (* Steuerungsnummer, Init: lokale Steuerung *)
  TouchProbeNo: TP_OBJECTS := NO_TOUCHPROBE  (* object reference number *)
              (* Messtastereferenznummer, Init: kein Objekt *)
END_STRUCT
END_TYPE
```

Functional description Information is transferred to the respective PLCopen function block via **VAR_IN_OUT CONSTANT** for 14VRS and higher.

The TOUCHPROBE_REF structure contains two 16 bit address elements:

- Control address (element of [CONTROLS](#), page 27)
- Object address (element of [TP_OBJECTS](#), page 31).



Information was transferred up to 13VRS via **VAR_IN_OUT**.

In 14VRS and higher, information is transferred via **VAR_IN_OUT CONSTANT**.

3.3.8 TP_OBJECTS

Brief description The enumeration of available touch probes contains all objects of a Motion link accessed by touch probe function blocks, also refer to ([TOUCHPROBE_REF](#), page 31).

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of logical addresses

Tab. 3-9: TP_OBJECTS reference table

Type definition

```
(* Enumeration of valid logical object addresses *)
TYPE TP_OBJECTS :
(
  (* not specified object *)
  NO_TOUCHPROBE := 0,      (* no object specified *)

  (* Touch Probe 1..100, realized on controller *)
  TOUCHPROBE_1  := 01,    (* Touch Probe 1 *)
  (* ... *)
  TOUCHPROBE_100 := 100   (* Touch Probe 100 *)
)
```

```

(* Touch Probe 1001..1002, realized on drive      *)
  TOUCHPROBE_DRIVE_1 :=1001  (* Touch Probe Drive 1 *)
  TOUCHPROBE_DRIVE_2 :=1002  (* Touch Probe Drive 2 *)
);
END_TYPE

```



Names and numbers are assigned when the respective touch probe is manually created in the IndraWorks Suite (IndraLogic, GlobalMlcVariables)



The number of implementable touch probes depends on the control type and the firmware version.

3.4 Data types to support cams, MotionProfiles and FlexProfiles

3.4.1 Data type to switch between cams, general information

Depending on the potential of the control electronics, the following cam tables can be stored:

- MPDx - 4 tables with 1,024 points each (table 1...4),
- MPBx - 4 tables with 1,024 points each (table 1...4),
- MPHx - 4 tables with 1,024 points each (table 1...4), 4 tables with 128 points each (table 5...8).

Cam table switching is possible at runtime. Select and switch via the function block MC_CamIn (CamTableID: [MC_CAM_ID](#), [page 32](#)) or via the parameter control "A-0-2740, ELS, CamIn, cam table preselection".

3.4.2 MC_CAM_ID

Brief description

The enumeration type supports the selection of the cam table ([Data types to support cams, MotionProfiles and FlexProfiles](#), [page 32](#)).

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of PLCopen

Tab. 3-10: MC_CAM_ID reference table

Type definition

```

TYPE MC_CAM_ID :
(
  CAM_TABLE_1  := 1,  (* Kurventabelle 1 *)
  CAM_TABLE_2  := 2,  (* Kurventabelle 2 *)
  (* ... *)
  CAM_TABLE_255 := 255 (* Kurventabelle 255 *)
);
END_TYPE

```

Functional Description

Cam table switching is possible at runtime. Select and switch via the function block MC_CamIn, ML_PLCopen.library or via the parameter control "A-0-2740, ELS, CamIn, cam table preselection".

3.4.3 MC_STEP_MODE, MotionProfile extended to FlexProfile

Brief description

The enumeration data type "MC_STEP_MODE" describes the dependence of the slave axis position from the master axis during a motion step (MotionProfile extended to FlexProfile).

The dependence can be given as functional dependence or as cam table.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of PLCopen

Tab. 3-11: MC_STEP_MODE reference table

Type definition

```

TYPE MC_STEP_MODE :
(
  REST_IN_REST_INCLINEDSINE      := 16#0000, (* MotionProfile, FlexProfile *)
  REST_IN_REST_POLY5             := 16#0100, (* MotionProfile, FlexProfile *)
  REST_IN_VELOCITY_POLY5         := 16#0200, (* MotionProfile, FlexProfile *)
  REST_IN_VELOCITY_POLY7         := 16#0300, (* FlexProfile *)
  VELOCITY_IN_REST_POLY5         := 16#0400, (* MotionProfile, FlexProfile *)
  VELOCITY_IN_REST_POLY7         := 16#0500, (* FlexProfile *)
  CONSTANT_VELOCITY              := 16#0600, (* MotionProfile, FlexProfile *)
  VELOCITY_IN_VELOCITY_POLY5     := 16#0700, (* MotionProfile, FlexProfile *)

  REST_IN_REST_POLY5_VLIM        := 16#1200, (* MotionProfile, FlexProfile *)
  REST_IN_REST_PARABOLA         := 16#1300, (* FlexProfile *)
  REST_IN_REST_POLY8            := 16#1400, (* FlexProfile *)

  REST_IN_REST_LINEAR            := 16#0800, (* FlexProfile *)
  REST_IN_REST_POLY7            := 16#0900, (* FlexProfile *)
  REST_IN_REST_SINE             := 16#0A00, (* FlexProfile *)
  REST_IN_REST_GUTMANSINE       := 16#0B00, (* FlexProfile *)
  REST_IN_REST_SINEACC          := 16#0C00, (* FlexProfile *)
  REST_IN_REST_SINETORQUE       := 16#0D00, (* FlexProfile *)
  REST_IN_REST_MOD_TRAPEZE      := 16#0E00, (* FlexProfile *)
  REST_IN_REST_MOD_SINE         := 16#0F00, (* FlexProfile *)

  VELOCITY_IN_VELOCITY_POLY7     := 16#1000, (* FlexProfile *)
  VELOCITY_IN_VELOCITY_MOD_SINE  := 16#1100, (* FlexProfile *)

  MOTION_IN_MOTION_POLY5        := 16#2000, (* FlexProfile, *)
  MOTION_IN_MOTION_POLY7        := 16#2100, (* FlexProfile, *)
  MOTION_IN_MOTION_POLY7        (* auch FlexStep / also Flex motion step *)

  COMMON_POLY5                  := 16#4000, (* FlexProfile *)
  COMMON_POLY7                  := 16#4100, (* FlexProfile *)
  COMMON_POLY2                  := 16#4200, (* FlexProfile *)
  COMMON_POLY3                  := 16#4300, (* FlexProfile *)
  COMMON_POLY4                  := 16#4400, (* FlexProfile *)
  COMMON_POLY8                  := 16#4500, (* FlexProfile *)

  (* Erweiterte Bewegungsgesetze / Extended motion laws *)
  X_VELOCITY_IN_VELOCITY_TRAPEZE_ALIM := 16#5000, (* FlexProfile *)
  X_MOTION_IN_MOTION_POLY5_VLIM      := 16#6000, (* FlexProfile *)
  X_MOTION_IN_MOTION_POLY5_SLIM      := 16#6100, (* FlexProfile, *)
  X_MOTION_IN_MOTION_POLY5_SLIM      (* Auch FlexStep / also Flex motion step *)
  X_FIT_VEL_TRAPEZE_ALIM              := 16#7000, (* FlexProfile *)
  X_FIT_SINE_TRAPEZE_ALIM             := 16#7100, (* FlexProfile *)
  X_FIT_ACC_TRAPEZE_ULIM              := 16#7200, (* FlexProfile *)

  X_MOTION_IN_MOTION_ACAM             := 16#8000, (* FlexProfile, *)
  X_MOTION_IN_MOTION_VCAM             := 16#8100, (* FlexProfile, *)
  X_MOTION_IN_MOTION_VCAM2_A          := 16#8200, (* FlexProfile, *)
  X_MOTION_IN_MOTION_VCAM2_B          := 16#8300, (* FlexProfile, *)

  (* Punktetabellen MotionProfile / Cam tables *)
  CAMTABLE_1                        := 1,          (* C-0-2001 *)
  CAMTABLE_2                        := 2,          (* C-0-2002 *)
  CAMTABLE_3                        := 3,          (* C-0-2003 *)
  CAMTABLE_98                       := 98,         (* C-0-2098 *)
  CAMTABLE_99                       := 99,         (* C-0-2099 *)
);
END_TYPE

```

Functional description



The following functional description contains details on Motion-Profiles and FlexProfiles.

Different standard profiles or a cam table can be defined as step mode.

The following modes are available:

- R-R: Rest-in-rest, polynomial 5th order,
- R-R: Rest-in-rest, inclined sinusoid,
- R-V: Rest-in-velocity, polynomial 5th order,
- V-R: Velocity-in-rest, polynomial 5th order,
- G: Constant velocity
- V-V: velocity-in-velocity, polynomial 5th order,
- Cam table.

R-R: Rest-in-rest

For the profile "Rest-in-rest", ten profiles are available. It can be selected between the standardized motion laws "Polynomial 5th order" and "Inclined sine curve" ...

These profiles are defined by the distance and the master axis range. The final velocity of the step is determined by the following step.

The profile for the "**polynomial 5th order**" is determined by the following function:

$$f(z) = 6z^5 - 15z^4 + 10z^3$$

Fig. 3-1: Rest-in-rest, polynomial 5th order

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	0	0
2. Order (acceleration)	0	0
3. Order (jerk)	Automatically	Automatically

Tab. 3-12: Boundary values for the motion law - polynomial 5th order

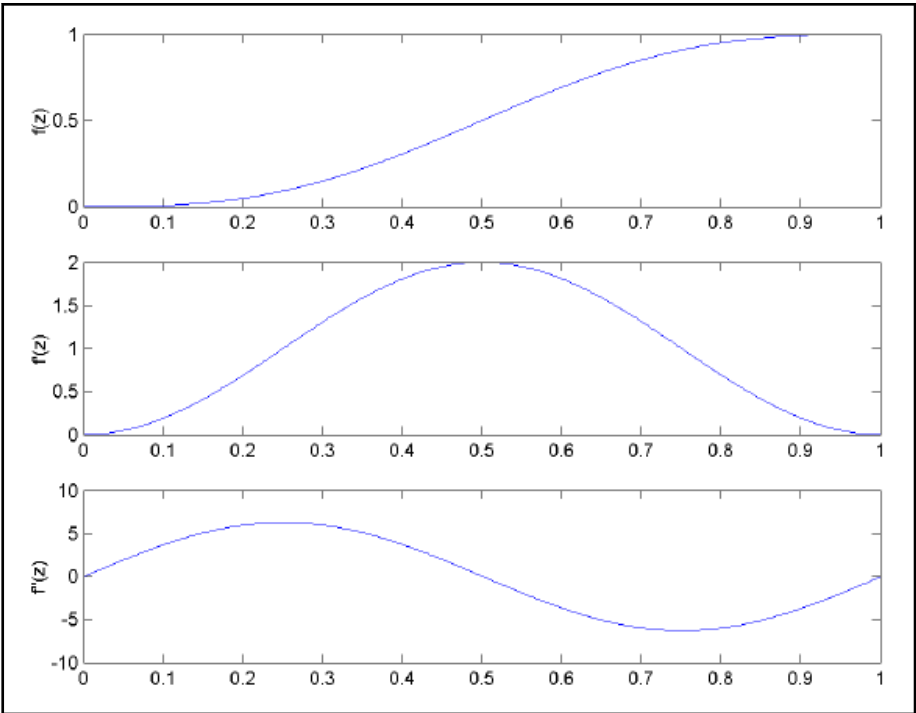


Fig. 3-2: Rest-in-rest, polynomial 5th order

For the "polynomial 7th order", the following formula applies:

$$f(z) = -20z^7 + 70z^6 - 84z^5 + 35z^4$$

Fig. 3-3: Rest-in-rest, polynomial 7th order

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	0	0
2. Order (acceleration)	0	0
3. Order (jerk)	0	0

Tab. 3-13: Marginal values for the motion law - Polynomial 7th order

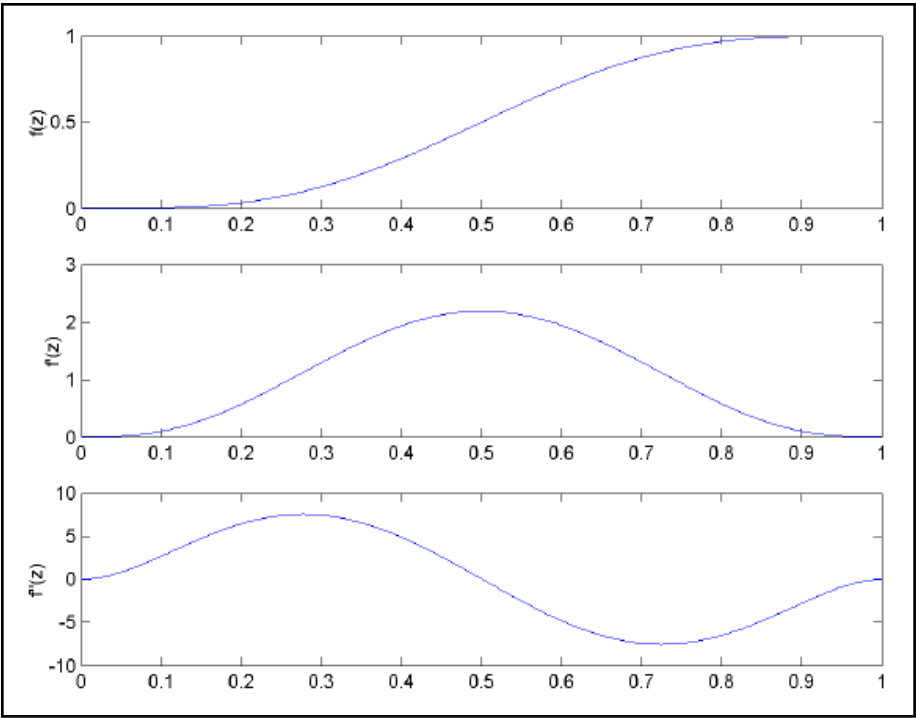


Fig. 3-4: Rest-in-rest, polynomial 7th order

The profile for the **"inclined sine curve"** is determined by the following function:

$$f(z) = z - \frac{1}{2\pi} \sin(2\pi z)$$

Fig. 3-5: Rest-in-rest, inclined sine curve

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	0	0
2. Order (acceleration)	0	0
3. Order (jerk)	Automatically	Automatically

Tab. 3-14: Marginal values for the motion law - Inclined sine curve

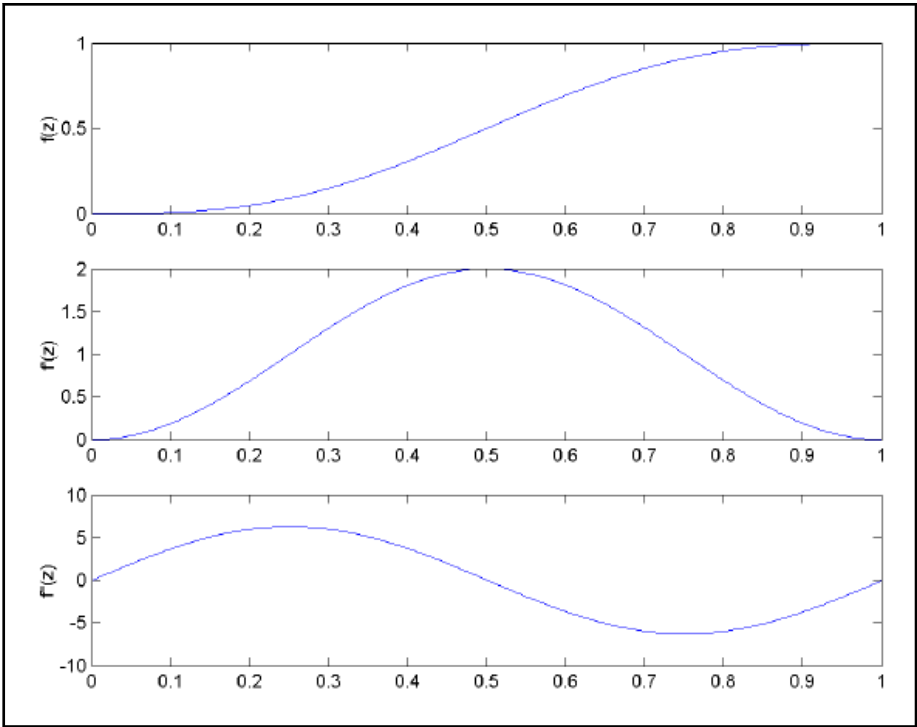


Fig. 3-6: Rest-in-rest, Inclined sine curve (BESTEHORN)

The profile for the "Sine curve (simple sinoid)" is determined by the following function:

$$f(z) = \frac{1}{2} \left[1 - \cos(\pi z) \right]$$

Fig. 3-7: Rest-in-rest, sine curve

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	0	0
2. Order (acceleration)	Automatically	Automatically
3. Order (jerk)	Automatically	Automatically

Tab. 3-15: Marginal values for the motion law "Sine curve"

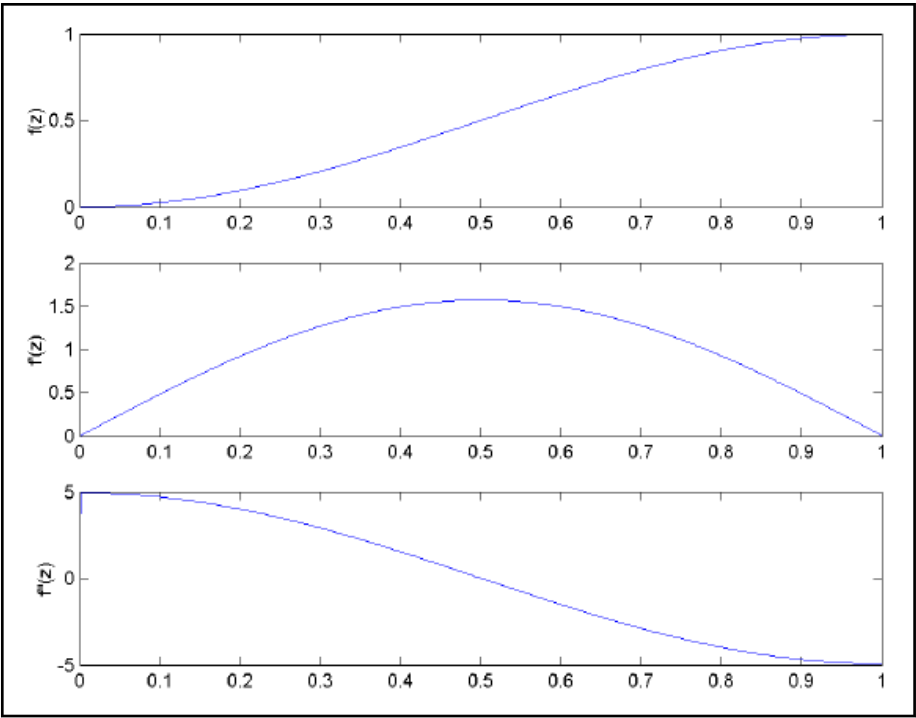


Fig. 3-8: Rest-in-rest, Sine curve (simple sinoide)
The profile for the "Linear rest" is determined by the following function:

$$f(z) = 0$$

Fig. 3-9: Linear rest
The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	0	0
2. Order (acceleration)	0	0
3. Order (jerk)	0	0

Tab. 3-16: Marginal values for motion laws - Linear rest

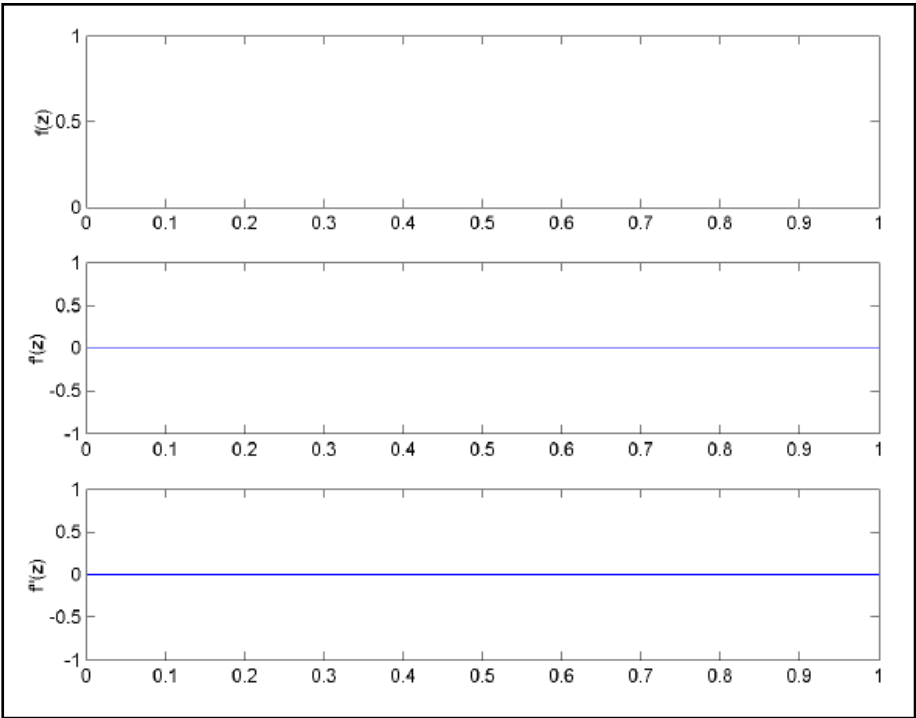


Fig. 3-10: Linear rest

The profile for the "Gutman sinoide" is determined by the following function:

$$f(z) = z - \frac{15}{32\pi} \sin(2\pi z) - \frac{1}{96\pi} \sin(6\pi z)$$

Fig. 3-11: Gutman sinusoid

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	0	0
2. Order (acceleration)	0	0
3. Order (jerk)	Automatically	Automatically

Tab. 3-17: Marginal values for the motion law - Gutman sinoide

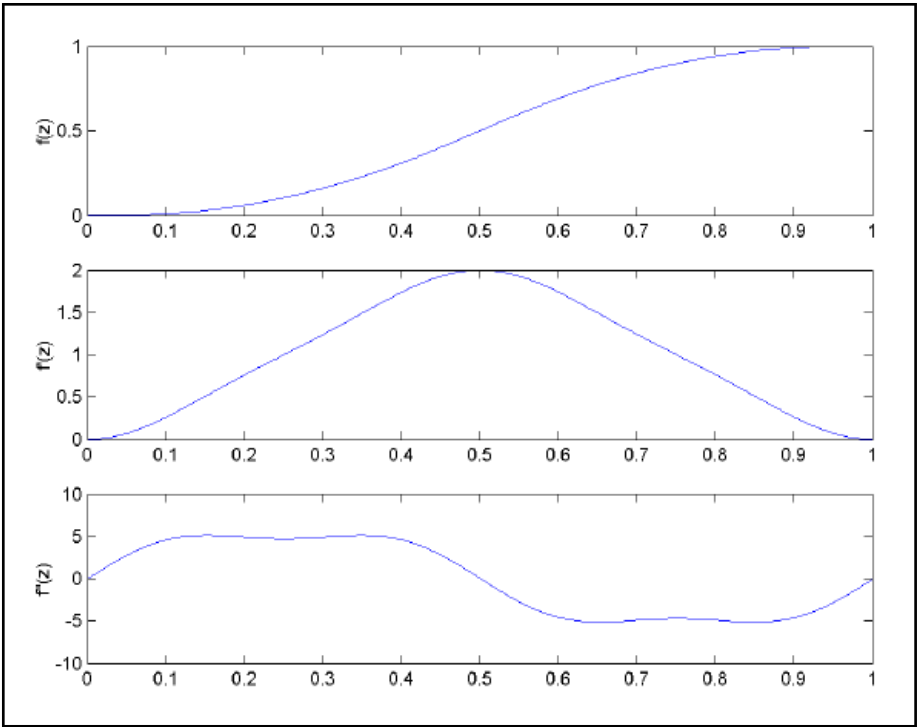


Fig. 3-12: Rest-in-rest, Gutman sinoide

The profile for the **"inclined sine curve optimum for acceleration"** is determined by the following function:

$$f(z) = z - \frac{1}{2\pi} \sin(2\pi z)$$
$$g(z) = z - \frac{0.41}{2\pi} \sin(2\pi z)$$

Fig. 3-13: Acceleration-optimum inclined sinusoid

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	0	0
2. Order (acceleration)	0	0
3. Order (jerk)	Automatically	Automatically

Tab. 3-18: Marginal values for the motion law - Inclined sine curve optimum for acceleration

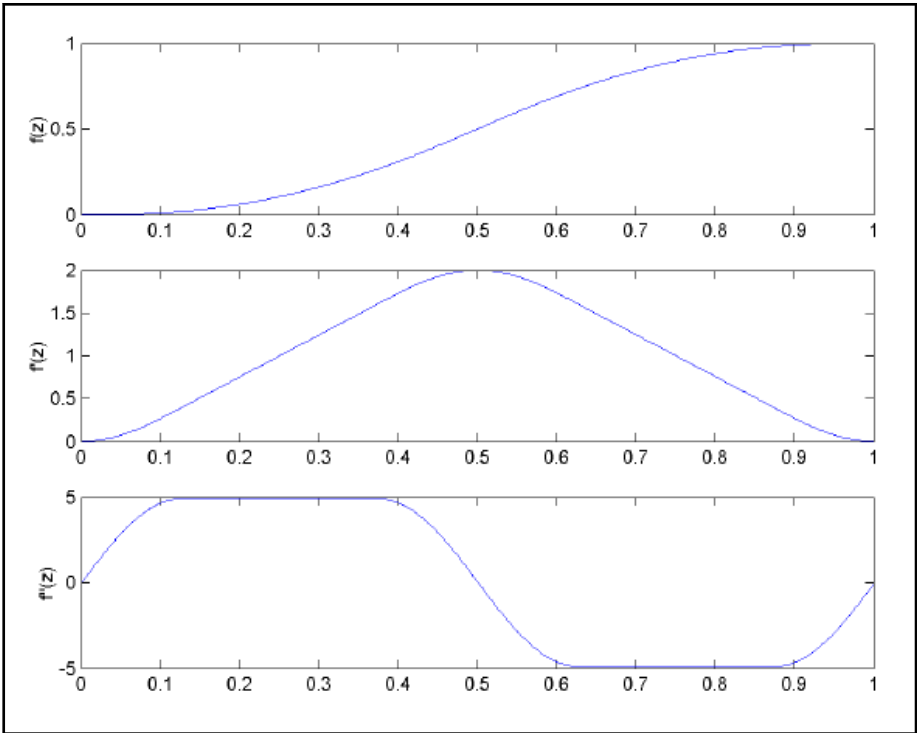


Fig. 3-14: Rest-in-rest, acceleration-optimum inclined sine curve

The profile for the "inclined sine curve optimum for the torque" is determined by the following function:

$$f(z) = z - \frac{1}{2\pi} \sin(2\pi z)$$
$$g(z) = z - \frac{0.134}{2\pi} \sin(2\pi z)$$

Fig. 3-15: Torque-optimum inclined sinusoid

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	0	0
2. Order (acceleration)	0	0
3. Order (jerk)	Automatically	Automatically

Tab. 3-19: Marginal values for the motion law - Inclined sine curve optimum for the torque

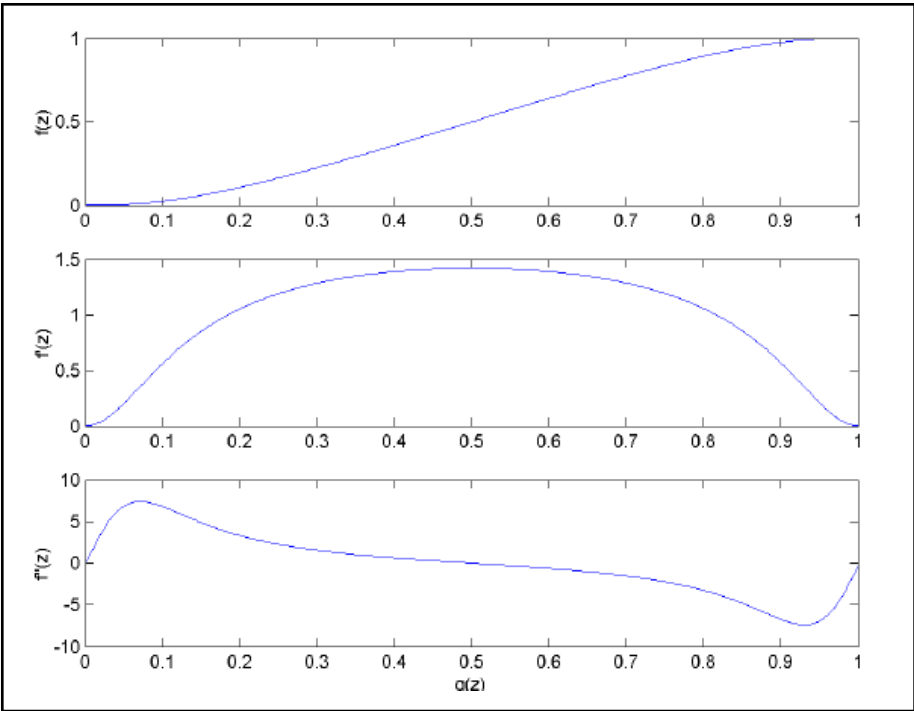


Fig. 3-16: Rest-in-rest, torque-optimum inclined sine curve

The profile for the **"Modified sinoide"** is determined by the following function:

$$\begin{aligned}
 0 \leq z \leq \frac{1}{8} : f(z) &= \frac{1}{4 + \pi} \left[\pi z - \frac{1}{4} \sin(4\pi z) \right] \\
 \frac{1}{8} \leq z \leq \frac{7}{8} : f(z) &= \frac{1}{4 + \pi} \left[2 + \pi z - \frac{4}{9} \sin\left(\frac{4\pi}{3} z + \frac{\pi}{3}\right) \right] \\
 \frac{7}{8} \leq z \leq 1 : f(z) &= \frac{1}{4 + \pi} \left[4 + \pi z - \frac{1}{4} \sin(4\pi z) \right]
 \end{aligned}$$

Fig. 3-17: Modified sinusoid

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	0	0
2. Order (acceleration)	0	0
3. Order (jerk)	Automatically	Automatically

Tab. 3-20: Marginal values for the motion law - Modified sinoide

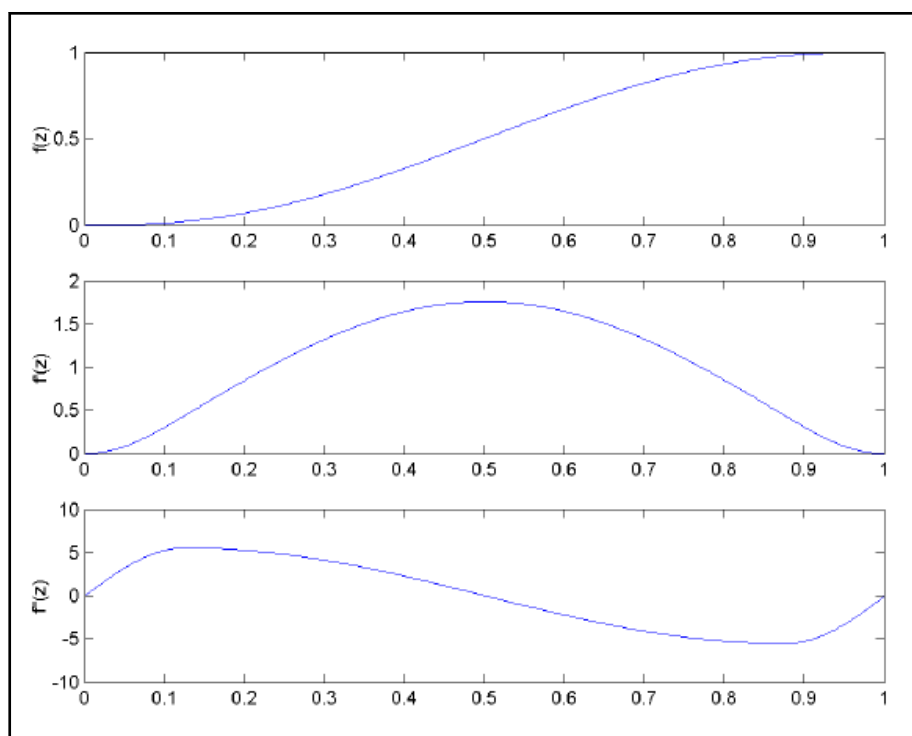


Fig. 3-18: Rest-in-rest, modified sine

The profile for the **"Modified acceleration trapezoid"** is determined by the following function:

$$\begin{aligned}
 0 \leq z \leq \frac{1}{8}: f(z) &= \frac{1}{2+\pi} \left[2z - \frac{1}{2\pi} \sin(4\pi z) \right] \\
 \frac{1}{8} \leq z \leq \frac{3}{8}: f(z) &= \frac{1}{2+\pi} \left[-\frac{1}{2\pi} + 2z + 4\pi \left(z + \frac{1}{8} \right)^2 \right] \\
 \frac{3}{8} \leq z \leq \frac{1}{2}: f(z) &= \frac{1}{2+\pi} \left[-\frac{\pi}{2} + 2(1+\pi)z - \frac{1}{2\pi} \sin(4\pi z - \pi) \right] \\
 \frac{1}{2} \leq z \leq 1: f(z) &= 1 - f(1-z)
 \end{aligned}$$

Fig. 3-19: Modified acceleration trapezoid

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	0	0
2. Order (acceleration)	0	0
3. Order (jerk)	Automatically	Automatically

Tab. 3-21: Marginal values for the motion law - Modified acceleration trapezoid

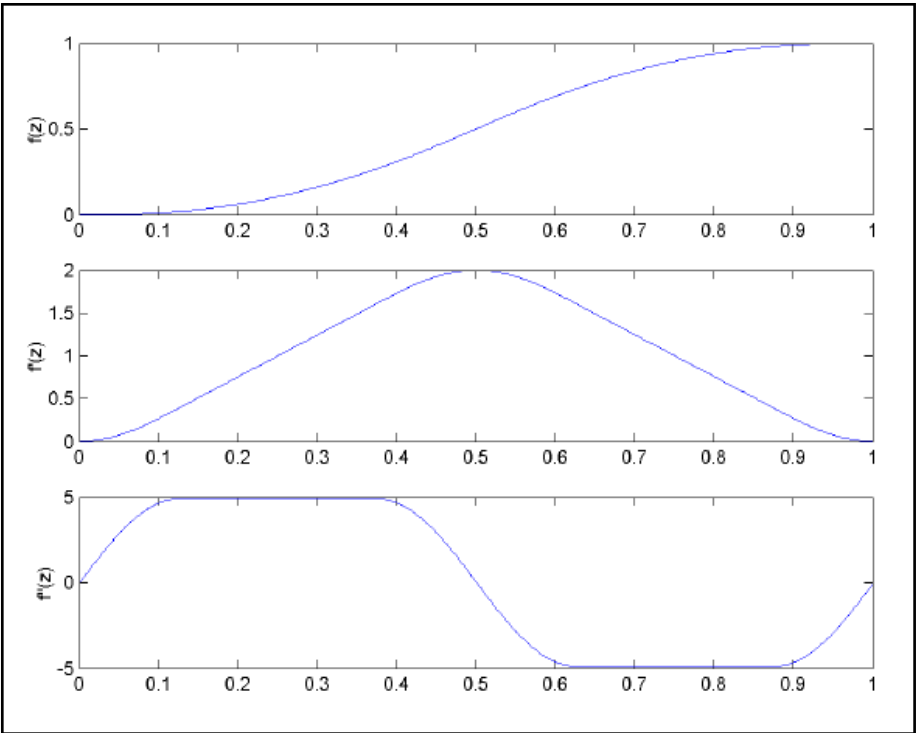


Fig. 3-20: Rest-in-rest, modified acceleration trapezoid

R-V: Rest-in-velocity

This profile is used for the transition from standstill to a certain velocity.
The profile is calculated with a "Polynomial 5th order":

$$f(z) = A_5 z^5 + A_4 z^4 + A_3 z^3$$

Fig. 3-21: Polynomial 5th order

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	0	Selectable
2. Order (acceleration)	0	Selectable
3. Order (jerk)	Automatically	Automatically

Tab. 3-22: Marginal values for the motion law - Polynomial 5th order

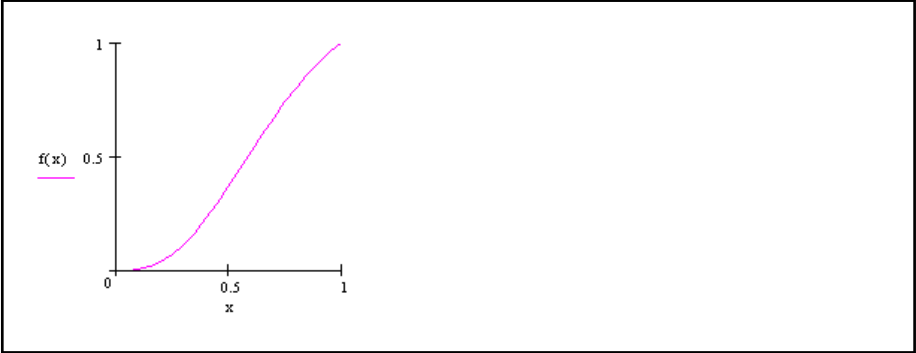


Fig. 3-22: Rest-in-velocity, polynomial 5th order

This profile is used for the transition from standstill to a certain velocity.
The profile is calculated with a "Polynomial 7th order":

$$f(z) = A_7 z^7 + A_6 z^6 + A_5 z^5 + A_4 z^4 + A_3 z^3$$

Fig. 3-23: Polynomial 7th order

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	0	Selectable
2. Order (acceleration)	0	Selectable
3. Order (jerk)	Automatically	Selectable

Tab. 3-23: Marginal values for the motion law - Polynomial 7th order

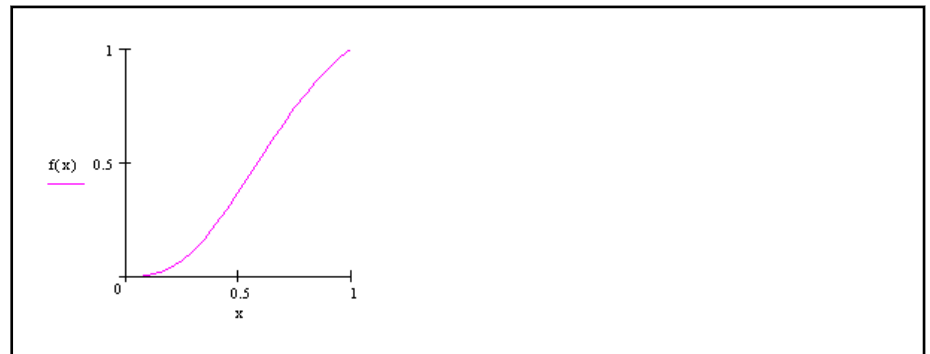


Fig. 3-24: Rest-in-velocity, polynomial 7th order

V-R: Velocity-in-rest

The profile is used for the transition from a defined velocity to the standstill.

The distance traveled in this MotionProfile is defined by the distance for this motion step.

The profile is calculated with a "Polynomial 5th order":

$$f(z) = A_5 z^5 + A_4 z^4 + A_3 z^3 + A_1 z$$

Fig. 3-25: Velocity-in-rest, polynomial 5th order

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	Selectable	0
2. Order (acceleration)	Selectable	0
3. Order (jerk)	Automatically	0

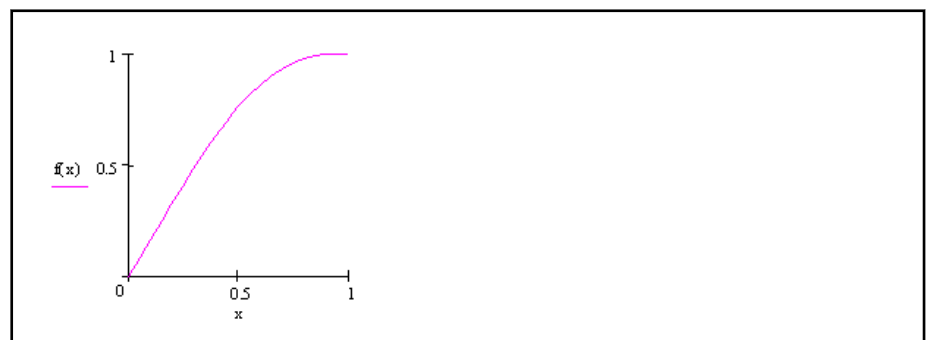


Fig. 3-26: Velocity-in-rest, polynomial 5th order

The profile is used for the transition from a defined velocity to the standstill.

The profile is calculated with a **"Polynomial 7th order"**:

$$f(z) = A_7z^7 + A_6z^6 + A_5z^5 + A_4z^4 + A_3z^3 + A_2z^2 + A_1z$$

Fig. 3-27: Rest-in-velocity, polynomial 7th order

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	Selectable	0
2. Order (acceleration)	Selectable	0
3. Order (jerk)	Selectable	0

Tab. 3-24: Marginal values for the motion law - Polynomial 7th order

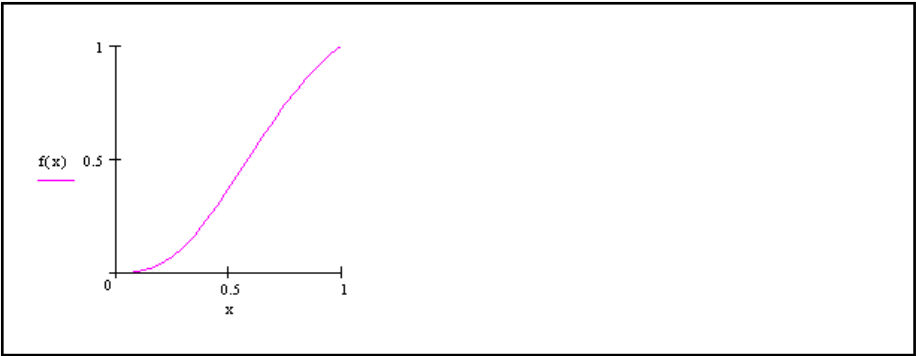


Fig. 3-28: Rest-in-velocity, polynomial 7th order

G: constant velocity

The profile of this motion step is a **"straight line"**. The velocity in this step is constant. The distance traveled is defined by the distance for this motion step.

The formula for the profile is standardized as follows:

$f(z) = z.$

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	Automatically	Automatically
2. Order (acceleration)	Automatically	Automatically
3. Order (jerk)	Automatically	Automatically

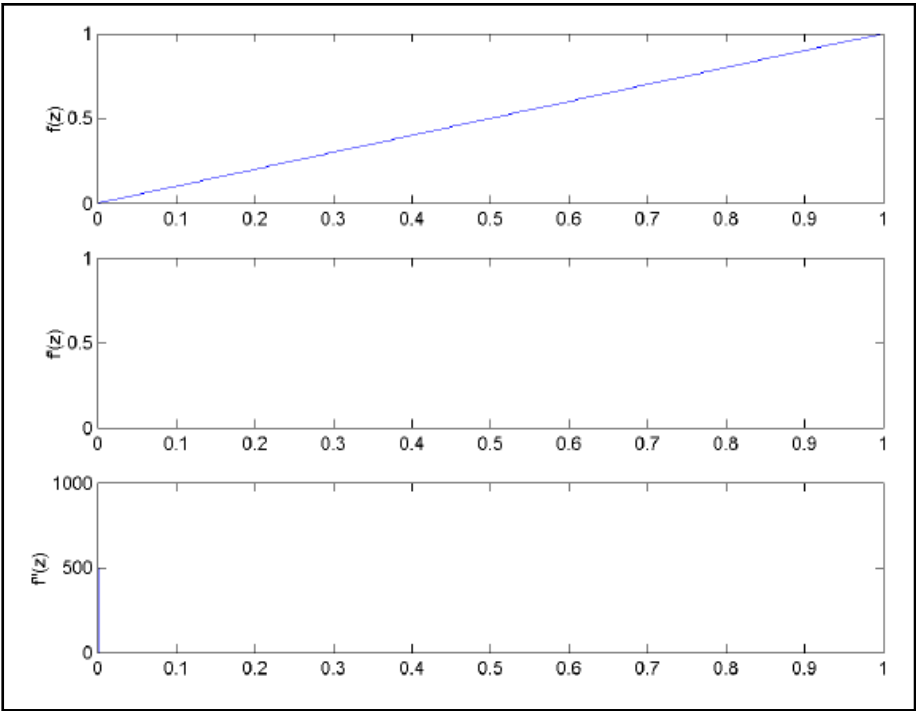


Fig. 3-29: Constant velocity

The velocity of the slave axis is determined from the values specified for this motion step:

- Master axis range
- Distance
- Master axis velocity.

It is not necessary to enter a value for the velocity of the slave axis.

V-V: Velocity-in-velocity

The profile for the "polynomial 5th order" is determined by the following function:

$$f(z) = A_5z^5 + A_4z^4 + A_3z^3 + A_2z^2 + A_1z$$

Fig. 3-30: Polynomial 5th order

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	Selectable	Selectable
2. Order (acceleration)	Selectable	Selectable
3. Order (jerk)	Automatically	Automatically

Tab. 3-25: Boundary values for the motion law - polynomial 5th order

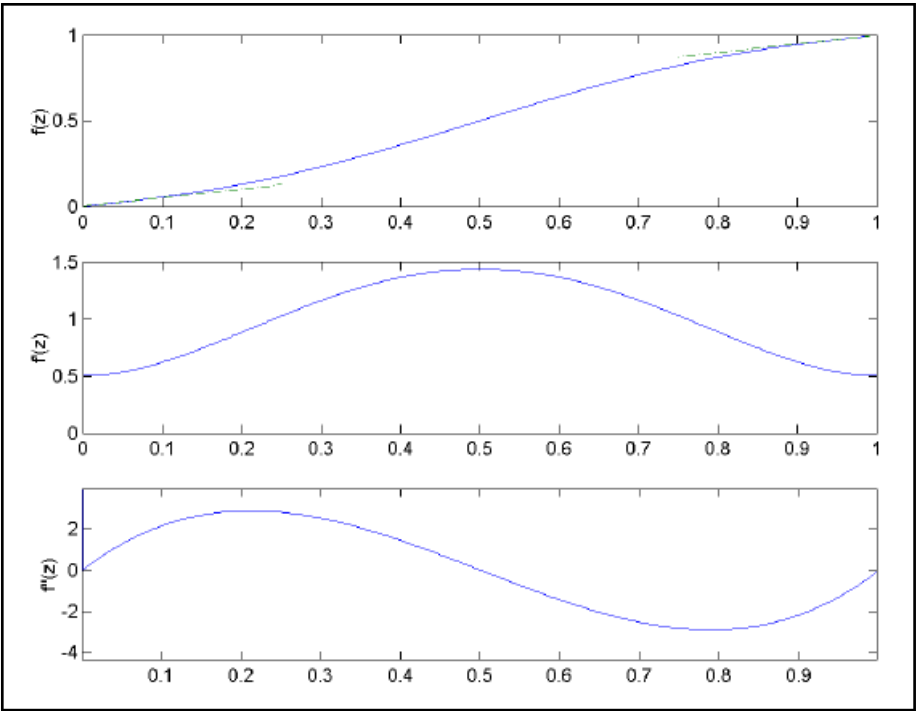


Fig. 3-31: Velocity-in-velocity, polynomial 5th order

The profile for the **"Polynomial 7th order"** is determined by the following function:

$$f(z) = A_7z^7 + A_6z^6 + A_5z^5 + A_4z^4 + A_3z^3 + A_2z^2 + A_1z$$

Fig. 3-32: Polynomial 7th order

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	Selectable	Selectable
2. Order (acceleration)	Selectable	Selectable
3. Order (jerk)	Selectable	Selectable

Tab. 3-26: Marginal values for the motion law - Polynomial 7th order

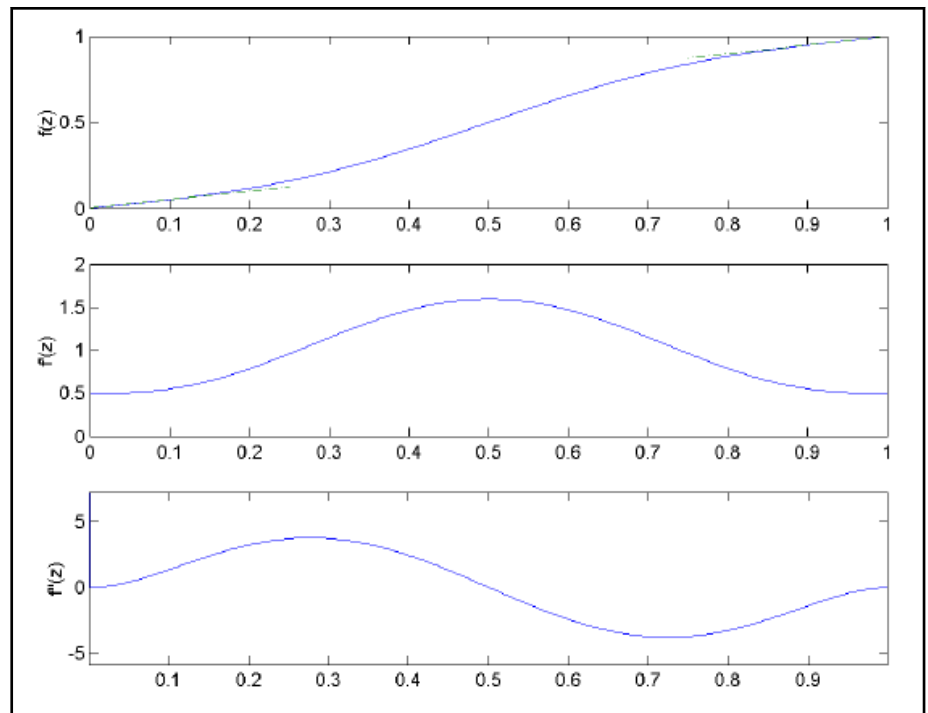


Fig. 3-33: Velocity-in-velocity, polynomial 7th order

The profile for the **"Modified sine curve"** is determined by the following function:

$$\begin{aligned}
 0 \leq z \leq \frac{\lambda}{8}: f'(z) &= C_a \frac{\lambda}{2\pi} \left(z - \frac{\lambda}{2\pi} \sin \left(2\pi \frac{z}{\lambda} \right) \right) + f'(0)z \\
 \frac{\lambda}{8} \leq z \leq \lambda: f'(z) &= C_a \frac{\lambda}{2\pi} \left(z - \frac{9\lambda}{2\pi} \sin \left(2\pi \frac{z + \frac{\lambda}{2}}{3\lambda} \right) + \frac{4\lambda}{\pi} \right) + f'(0)z \\
 \lambda \leq z \leq 1 - \frac{1-\lambda}{4}: f'(z) &= C_b \frac{1-\lambda}{2\pi} \left(z - \frac{9(1-\lambda)}{2\pi} \sin \left(2\pi \frac{z + \frac{3-5\lambda}{3(1-\lambda)}}{3\lambda} \right) + \frac{4\lambda}{\pi} \right) \\
 &\quad + f'(1) + (f'(0) - f'(1)) \left(\frac{3}{4} - \frac{1}{\pi} \right) + \lambda \\
 1 - \frac{1-\lambda}{4} \leq z \leq 1: f'(z) &= -C_b \frac{1-\lambda}{2\pi} \left(z - \frac{1-\lambda}{2\pi} \sin \left(2\pi \frac{z + \lambda}{1-\lambda} \right) - 1 \right) \\
 &\quad + 1 + f'(1) + (z-1)
 \end{aligned}$$

Fig. 3-34: Modified sine line

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	Selectable	Selectable
2. Order (acceleration)	0	0
3. Order (jerk)	0	0

Tab. 3-27: Marginal values for the motion law - Modified sine curve

Extended motion laws

The extended motion laws are special motion laws. They differ from the known motion laws of the VDI guidelines by their new degrees of freedom. While a standard motion law is unambiguously defined by the specification of its left and right marginal values, the motion laws presented allow further different settings, e.g. a required maximum velocity. In this case, the master axis range of the motion step does not have to be indicated. Instead, this results from the parameterization of the motion law and the specification of characteristic values. Thus, for example, acceleration-limited motion allows the distance, starting and end velocity to be specified, but not the setting of a master axis range. The latter, and thus also the duration of the motion step, results from the maximum acceleration and maximum velocity values, which can also be specified.

	Distance	Master axis section	$v_{\max}$	$a_{\max}$	$j_{\max}$
Overshooting-free Polynomial 5th order	Selectable	Selectable	Selectable	Automatically	Automatically
Velocity-limited Polynomial 5th order	Selectable	Selectable	Selectable	Automatically	Automatically
Acceleration-limited Motion	Selectable	Automatically	Selectable	Selectable	Automatically
Velocity-limited Sinoide profile	Selectable	Automatically	Selectable	Selectable	Automatically
Jerk-limited motion	Selectable	Automatically	Selectable	Selectable	Selectable

Tab. 3-28: Extended motion laws

The profile for the **"Velocity-limited polynomial 5th order"** can be as follows:

The polynomial 5th order known from the VDI does not allow for a control of the maximum travel velocity. The motion law is subdivided into three sections to solve this problem. The middle section corresponds to a straight line with the required velocity. This is surrounded by two polynomials 5th order. These ensure that there is a smooth transition from the starting and end velocity to the travel velocity. The intersecting points between the first and second and the second and third sections have to be determined. The points are selected in such a way that no turning point is set for the two polynomials

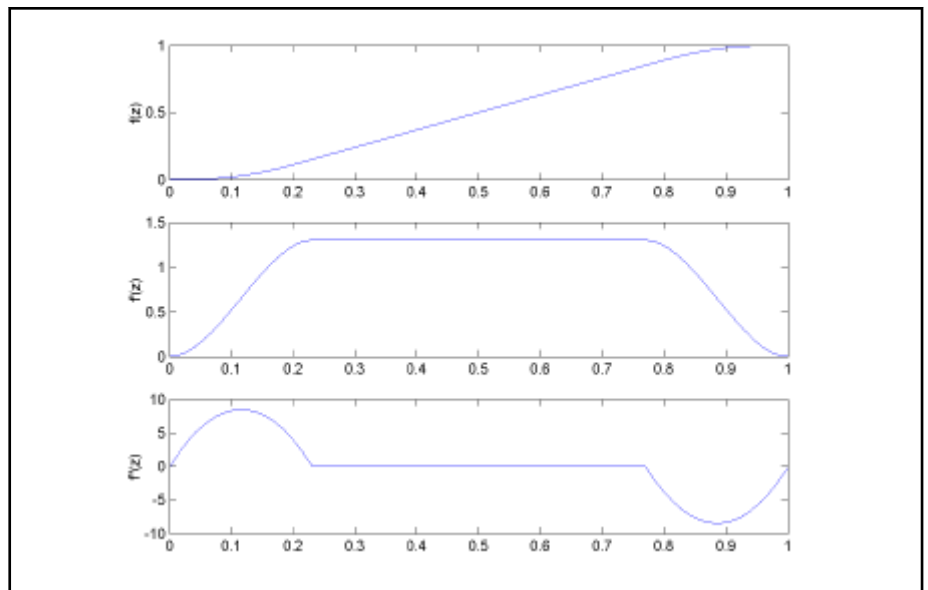


Fig. 3-35: Velocity-limited polynomial 5th order

The profile for **"No overshoot 5th order polynomial"** can be described as follows:

The motion law shown detects and prevents the possible overshoot of the polynomial. The kind of modification depends on the combination of the standardized boundary velocity values. Four cases result. For combinations in which the standardized starting velocity " v_0 " and the standardized end velocity " v_1 " have different signs, this motion law cannot be used. This means that the motion path has a bend at a limit point if it is not able to leave the standardized range of $0 \leq z \leq 1$ and $0 \leq f(z) \leq 1$.

- Case 1:
 $V_0 \leq 1$ and $V_1 \leq 1$

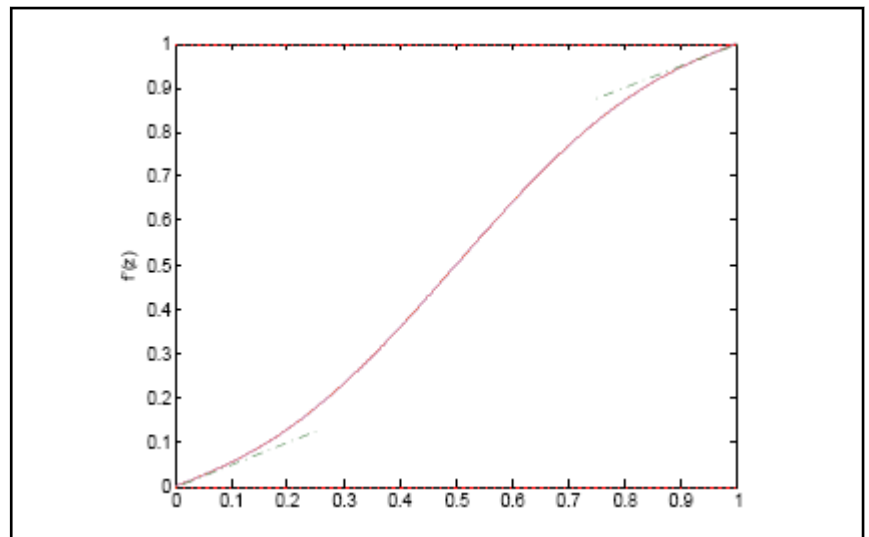


Fig. 3-36: Case 1

- Case 2:
 $V_0 \leq 1$ and $V_1 > 1$

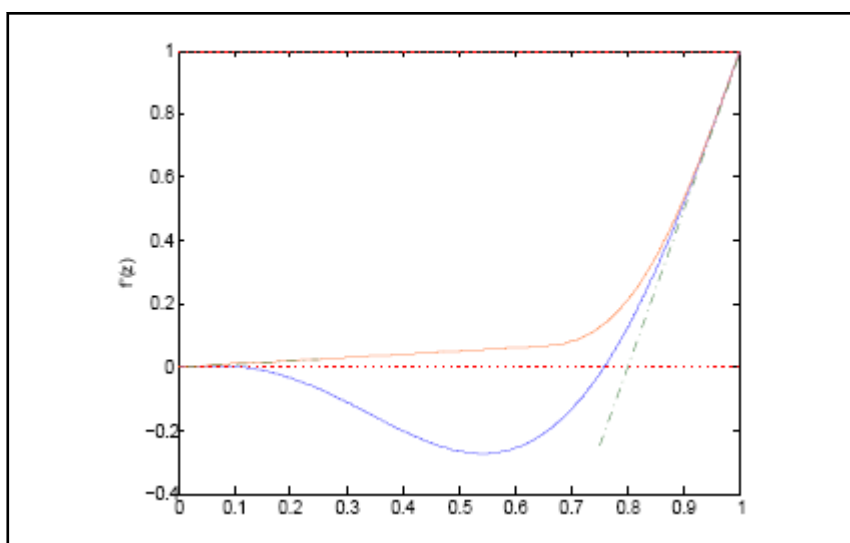


Fig. 3-37: Case 2

- Case 3:
 $V_0 > 1$ and $V_1 \leq 1$

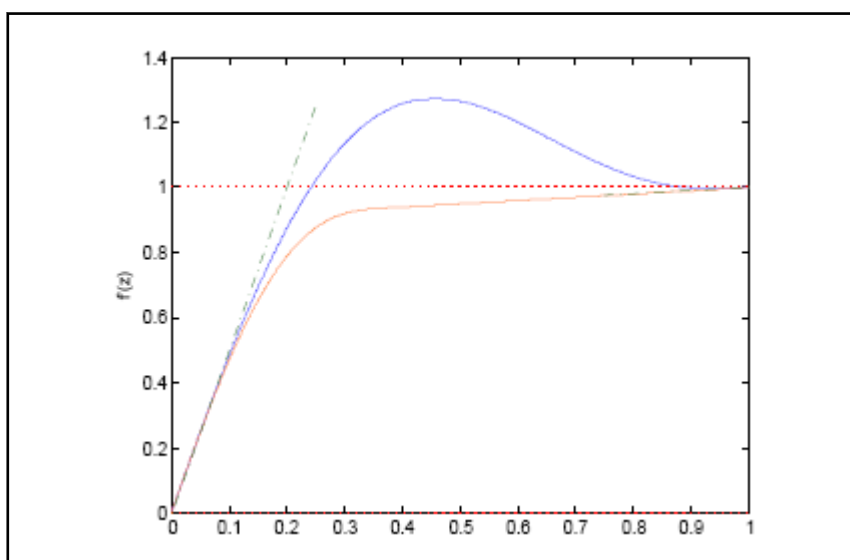


Fig. 3-38: Case 3

- Case 4:
 $V_0 > 1$ and $V_1 > 1$

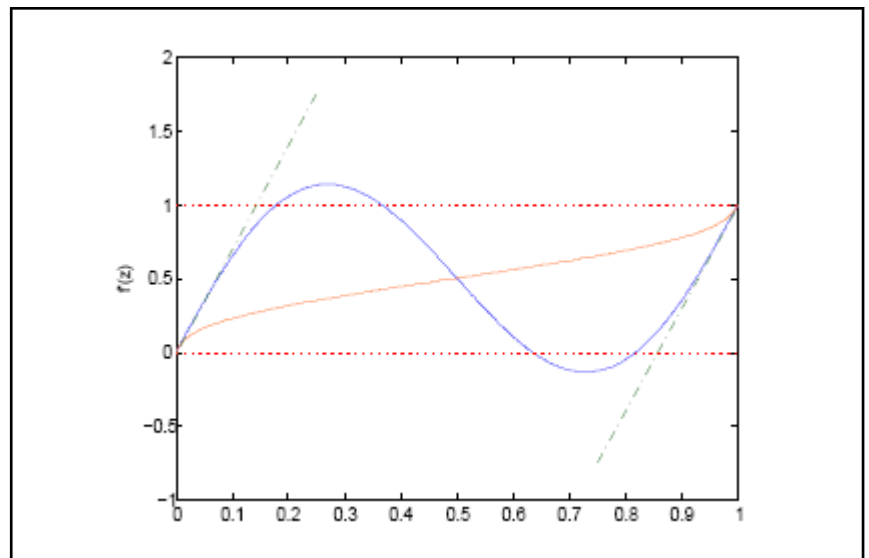


Fig. 3-39: Case 4

The profile for the **"Acceleration-limited motion"** can be described as follows:

The trapezoid profile, often also known as a ramp profile, can be seen as the simplest form of path interpolation. If a way "s" should be traveled, the course of the path can be divided into three phases. In a first phase, in which it is accelerated to a predetermined travel velocity with a predetermined acceleration value, a second phase of constant velocity and finally a deceleration phase in which it is decelerated with a specified delay. The function of the path interpolation is to calculate the starting and end points in time t_1 , t_2 and t_3 and the individual travel times T_1 , T_2 and T_3 from these specified acceleration values a_1 and a_3 as well as the desired velocity v_m and the distance to be traveled s . Furthermore, a required starting velocity v_0 and an end velocity v_3 are considered in the variant presented.

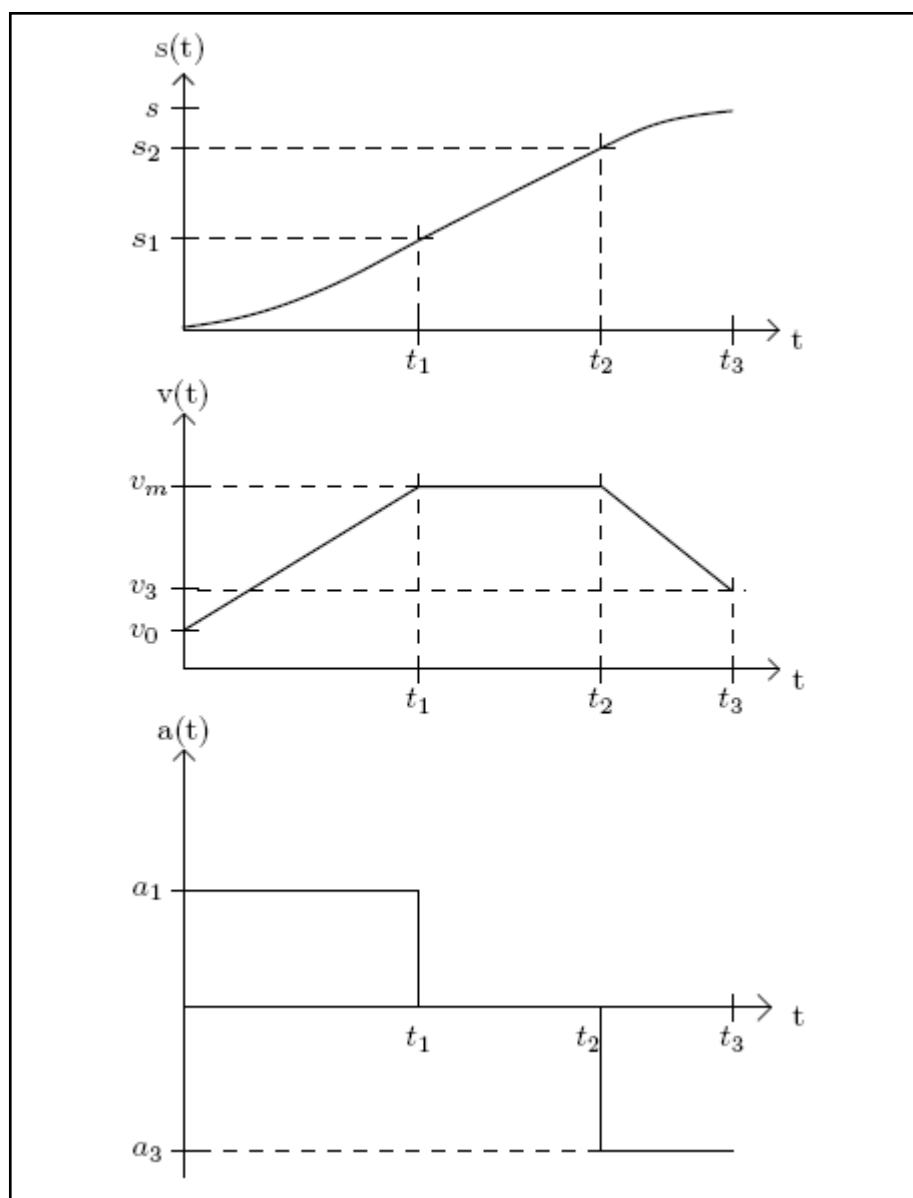


Fig. 3-40: Acceleration-limited motion

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	Selectable	Selectable
2. Order (acceleration)	Automatically	Automatically
3. Order (jerk)	Automatically	Automatically

Tab. 3-29: Marginal values for the motion law - Acceleration-limited motion

The profile for the **"Acceleration-limited motion with a sinoide profile"** can be described as follows:

With the sinoide profile, the starting and end velocities can be unequal to 0 and have different acceleration factors a_1 and a_3 for the acceleration and deceleration phase. Furthermore, special cases in which the starting velocity v_0 is already higher than the desired travel velocity v_m are considered. The same applies for the end velocity v_3 .

The possible marginal values for the motion law are:

Marginal value	z=0	z=1
1. Order (velocity)	Selectable	Selectable
2. Order (acceleration)	Automatically	Automatically
3. Order (jerk)	Automatically	Automatically

Tab. 3-30: Marginal values for the motion law - Acceleration-limited motion with sine profile

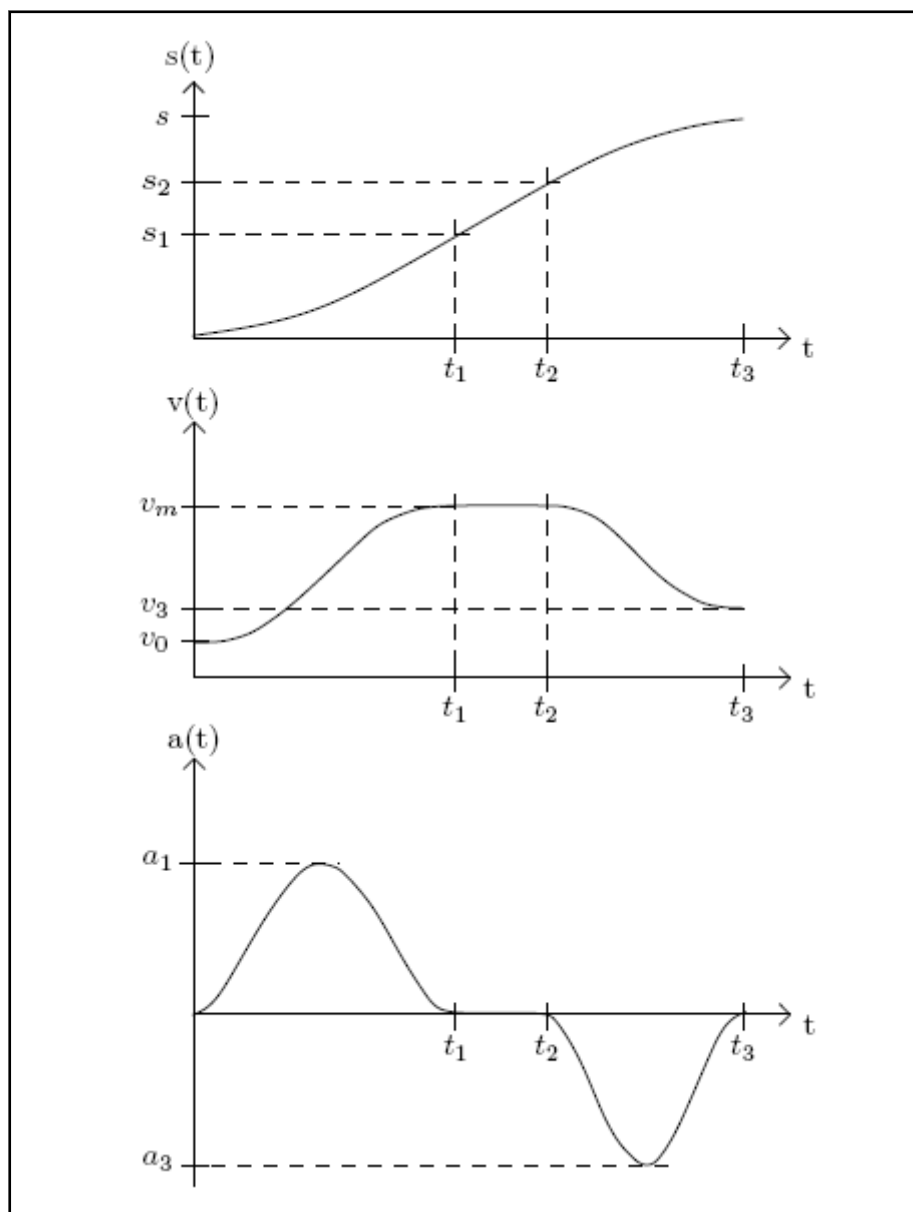


Fig. 3-41: Acceleration-limited motion with sine profile

The profile for the "**Jerk-limited motion**" can be described as follows:

Maintaining acceleration with jerk limitation is applicable to many demanding motion tasks. This applies, for example, to the transportation of fluids and to machines easily susceptible to vibration.

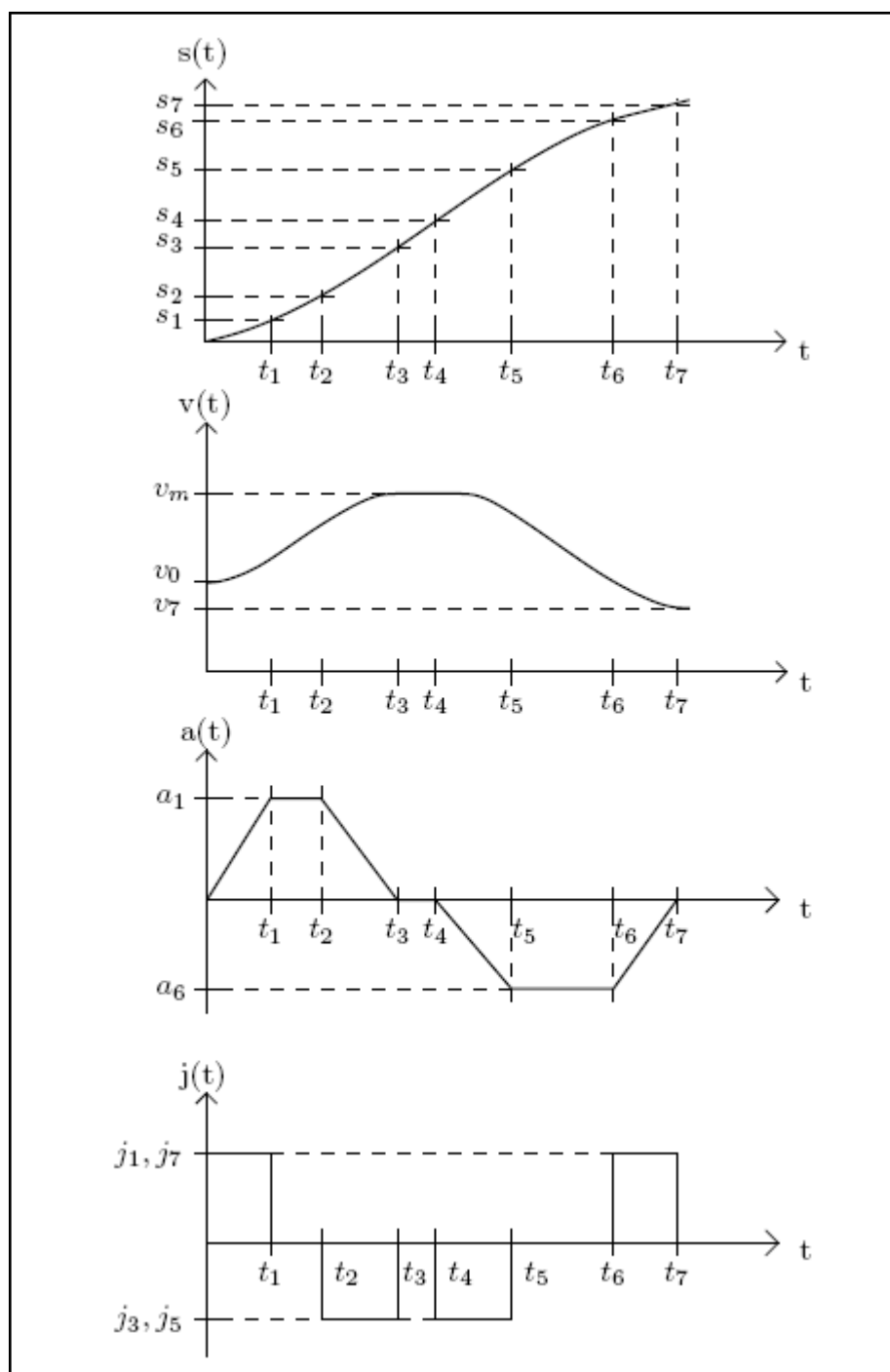


Fig. 3-42: Jerk-limited motion

3.5 Data types to synchronize in the position-controlled synchronous operation modes

3.5.1 Synchronization, introduction and overview

The functions described below are effective in the Phase Synchronization and Electronic Cam operation modes. In general, the slave axis is synchronized after the synchronous operation mode has been activated. This synchronization comprises two steps:

1. First of all, the slave drive accelerates or decelerates to the synchronous velocity with the "synchronization acceleration".
2. In the second step, the position difference is decreased. Therefore, a "synchronization velocity" is superimposed to the synchronous velocity.

Regarding the position adjustment in the second step, it is distinguished between relative and absolute synchronization ([MC_START_MODE](#), [page 57](#)):

- In case of a relative synchronization, the synchronization is completed once the synchronous velocity is reached.
- The absolute angle reference is only determined with the absolute synchronization once the synchronous velocity is reached (synchronization direction, [MC_SYNC_DIRECTION](#), [page 59](#)).

3.5.2 MC_START_MODE

Brief description This parameter defines the type of automatic dynamic synchronization in case of phase synchronization.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of PLCopen

Tab. 3-31: MC_START_MODE reference table

Type definition

```

TYPE MC_START_MODE :
(
  ABSOLUTE      := 0, (* synchronization with absolute angle reference, *)
                  (* only for first design *)
  RELATIVE      := 1, (* synchronization to synchronous velocity, *)
                  (* only for first design *)
  ABSOLUTE_RAMPIN := 2, (* synchronization with absolute angle reference, *)
                  (* for every design *)
  RELATIVE_RAMPIN := 3 (* synchronization to synchronous velocity, *)
                  (* for every design *)
);
END_TYPE

```



For the MB_GearInPos command, only the start modes ABSOLUTE and RELATIVE apply.

Absolute synchronization The slave axis position in the zero-crossing of the master axis position is unambiguously defined and does not change with every call of MB_MotionProfile/ MC_CamIn.

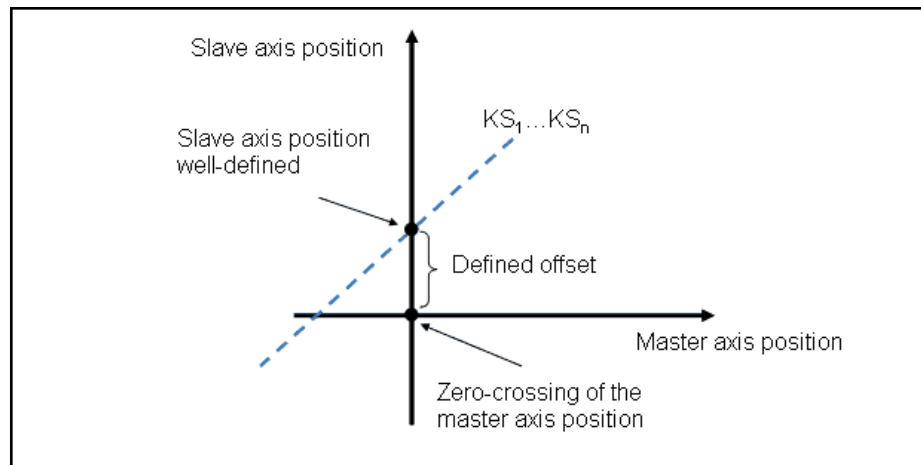


Fig. 3-43: Absolute synchronization

Relative synchronization

The slave axis position in the zero-crossing of the master axis position is arbitrary and changes with every call of MB_MotionProfile/ MC_CamIn.

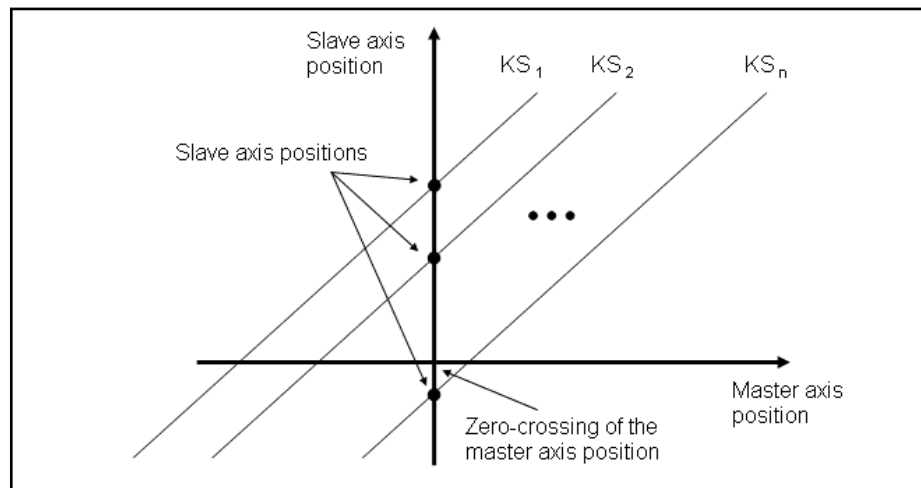


Fig. 3-44: Relative synchronization

"Hard" Synchronization

- StartMode= 0: Absolute synchronization, "hard changeover" in case of a change of profile,
- StartMode= 1: Relative synchronization, "hard changeover" in case of a change of profile

Switching after crossing the master axis switching position (A-0-2900, MotionProfile, master axis switching position) by the master axis.

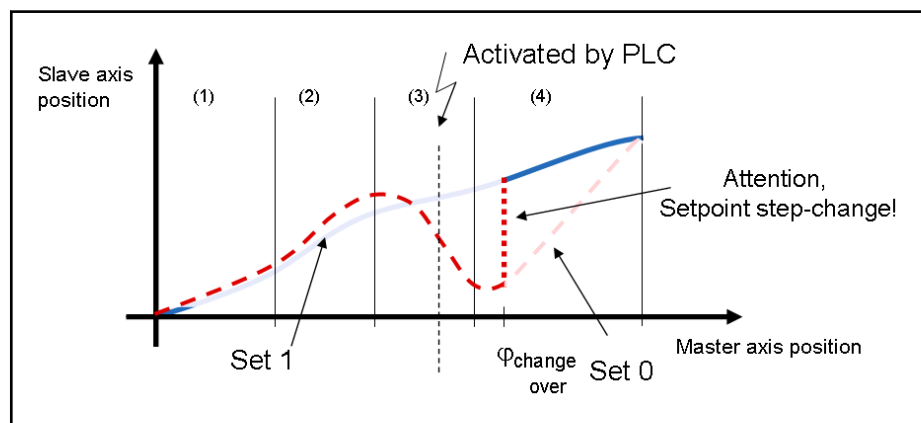


Fig. 3-45: Hard changeover

- "Soft" Synchronization**
- StartMode= 2: Absolute synchronization, "soft changeover" in case of a change of profile,
 - StartMode= 3: Relative synchronization, "soft changeover" in case of a change of profile.

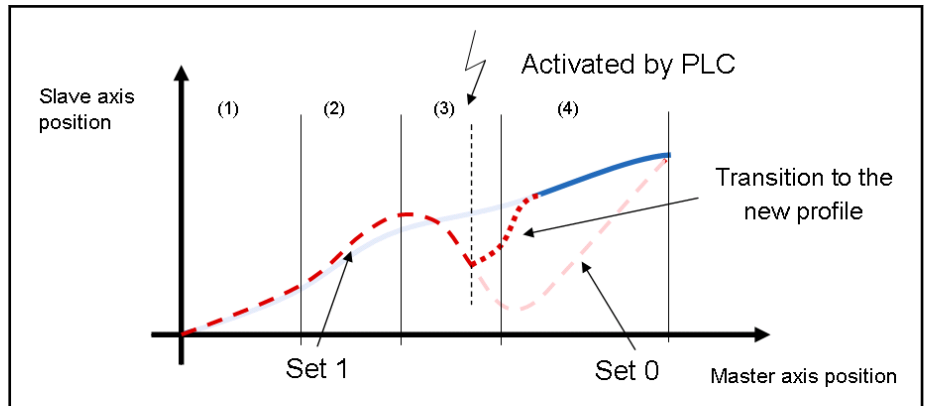


Fig. 3-46: Soft changeover

It is switched immediately after starting the function block and not waited for crossing the master axis switching position (A-0-2900, MotionProfile, master axis switching position) by the master axis.



The "soft synchronization" requires the calculation on the control.

3.5.3 MC_SYNC_DIRECTION

Brief description For modulo axes, the motion of the axis can be in positive or negative direction.

The direction of synchronization defines the direction of the axis motion.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of PLCopen

Tab. 3-32: MC_SYNC_DIRECTION reference table

Type definition

```

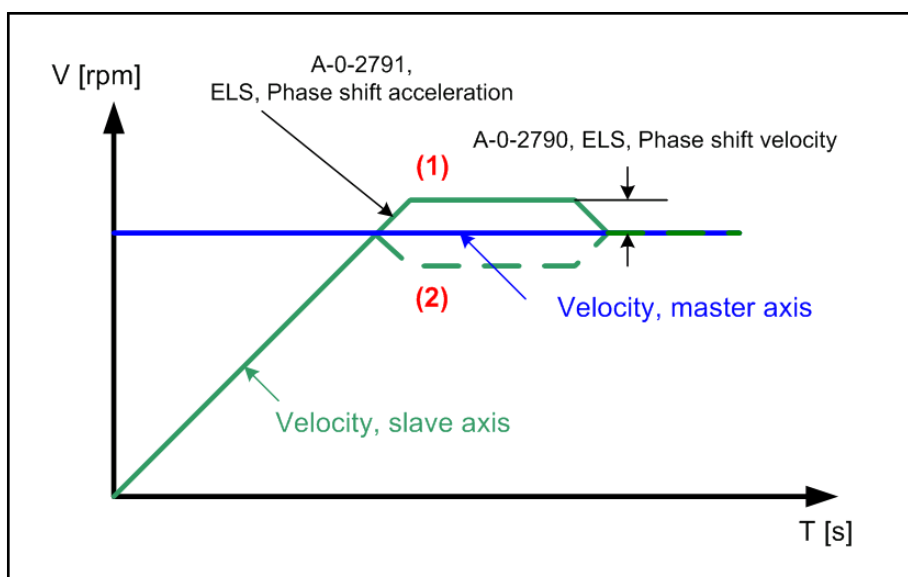
TYPE MC_SYNC_DIRECTION :
(
  SHORTEST_WAY    := 0,  (* kürzester Weg /shortest distance *)
  CATCH_UP        := 1,  (* immer vorwärts / always forward *)
  SLOW_DOWN       := 2
);
END_TYPE

```



If the shortest distance to the absolute synchronization is less than the "Synchronization window" for modulo format, the axis travels the shortest distance and ignores the specified synchronization direction.

Example MLC:



- ① The slave axis follows the master axis
 ② The master axis follows the slave axis

Fig. 3-47: Transition from velocity-synchronous to position-synchronous

In the figure, the slave axis first accelerates to the same velocity as the master axis. Subsequently, the angle difference between the master and the slave axis has to be reduced depending on which one runs in front of the other.

During synchronization, both parameters "A-0-2790, ELS, synchronization velocity" and "A-0-2791, ELS, synchronization acceleration" are applied.

The process is completed when both run velocity-synchronous **and** position-synchronous.

3.6 Data types to diagnose errors in the Motion function blocks

3.6.1 ERROR_CODE

Brief description This enumeration contains all error types characterizing an error at the function block.

The error code provides quick information on the cause of the error.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of POU diagnostics

Tab. 3-33: ERROR_CODE reference table

Type definition

```
(* Enumeration of valid types of diagnosis/ Aufzählung gültiger Fehlerarten *)
TYPE ERROR_CODE :
(
  NONE_ERROR           := 16#0000,  (* No error code available *)
  INPUT_INVALID_ERROR  := 16#0001,  (* kein Fehlercode verfügbar, FB fehlerfrei *)
  COMMUNICATION_ERROR  := 16#0002,  (* Invalid input *)
  RESOURCE_ERROR       := 16#0003,  (* Fehler auf Grund ungültiger Eingabewerte *)
  ACCESS_ERROR         := 16#0004,  (* Error while communication *)
                                (* Fehler beim Aufbau einer Kommunikation *)
                                (* bzw.während der Kommunikation *)
                                (* Resource error *)
                                (* Ressource (z.B.Parameter) ist nicht vorhanden *)
                                (* Access error *)
)
```

```

(* Ressource ist zwar da, aber man kann nicht *)
(* auf sie zugreifen. *)
(* Beispiel: Man möchte Parameter in Phase 4 be- *)
(* schreiben, der nur in Phase 2 beschreibbar ist*)
STATE_MACHINE_ERROR := 16#0005, (* Invalid state State - Machine *)
(* Der aktuelle Zustand der State-Machine *)
(* verhindert die weitere Abarbeitung *)
INPUT_RANGE_ERROR := 16#0006, (* Error because Input out of range *)
(* Während der Verarbeitung wird festgestellt, *)
(* dass durch eine oder mehrere Eingangsgrößen *)
(* undefinierte bzw. zu große oder zu kleine *)
(* Sollwerte generiert werden. *)
CALCULATION_ERROR := 16#0007, (* Error while calculation *)
(* Ein Berechnungsfehler ist aufgetreten, *)
(* z.B. Division durch 0 *)
DEVICE_ERROR := 16#0008, (* Device Error occurred *)
(* Übergeordneter Gerätefehler ist aufgetreten, *)
(* z.B IndraDrive: *)
(* Kurvenscheibenbetriebsart F2005, *)
(* Ungültige Kurvenscheibe *)
OTHER_ERROR := 16#7FFE, (* Undefined Error *)
(* allgemeiner (undefinierter) Fehler *)
SYSTEM_ERROR := 16#7FFF (* System error occurred *)
(* z.B. Betriebssystemfehler *)
);
END_TYPE

```

3.6.2 ERROR_STRUCT

Brief description The error structure provides any detailed information on an actual error.

This information can be evaluated and read directly at the corresponding instance of a PLCopen function block. The components must be interpreted as follows:

1. Determining table number([ERROR_TABLE](#), page 62). This identifies the origin of the error.
2. Use this finding to evaluate "Additional1". The content is the diagnostic number which can be found in the particular block descriptions (also refer to Troubleshooting Guide, MLC). Check whether "Additional2" = 0; if yes, diagnostic is completed; if no, proceed.
3. Evaluation of "Additional2" (not yet used for MLC_Table).

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of POU diagnostics, includes

Tab. 3-34: *ERROR_STRUCT* reference table

Type definition

```

(* Structure for description of diagnosis of function blocks. *)
(* The ERROR_STRUCT is a structure that contains information on the *)
(* corresponding diagnose, which is market by the PLCopen output 'Error' *)
TYPE ERROR_STRUCT :
STRUCT
  Table      : ERROR_TABLE; (* Additional diagnosis table *)
(* links to list used for identification *)
(* of the error of "Additional1" *)
  Additional1 : DWORD; (* Additional diagnosis number1 *)
(* Contains error number according to the *)
(* specified table. *)
  Additional2 : DWORD; (* Additional diagnosis number2 *)
(* Can contain additions to "Additional1". *)
END_STRUCT
END_TYPE

```



In **Additional1**, the messages, warnings and error messages described in "MLC, Diagnostics - Troubleshooting Guide" are displayed.

3.6.3 ERROR_TABLE

Brief description The ERROR_TABLE provides the first element of the [ERROR_STRUCT](#), [page 61](#). Using the number, select the table to be searched for the "Additional1" error code.

Example:

This should for example be "MLC_TABLE", 16#0030 for MLC applications.

The error numbers under "Additional1" contained in function block description get their information from this table.

Function blocks can use the table SERCOS_TABLE, 16#0010, if errors occur during parameter access.

Function blocks can use the tables F_RELATED_TABLE, 16#0170 or INDRV_TABLE, 16#0070 if errors occur when processing the command.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of POU diagnostics, includes

Tab. 3-35: ERROR_TABLE reference table

Type definition

```
(* Enumeration of valid tables in additional information *)
TYPE ERROR_TABLE:
(
    NO_TABLE_USED      := 16#0000, (* reserved *)
    SERCOS_TABLE       := 16#0010, (* Sercos Error: Addtitionall = Sercos Code *)
    MLD_TABLE          := 16#0020, (* Drive-based Motion - Logic *)
    MLC_TABLE          := 16#0030, (* Controller - based Motion - Logic *)
    MTX_TABLE          := 16#0040, (* CNC *)
    MLP_TABLE          := 16#0050, (* PC - based Motion Logic *)
    PLC_TABLE          := 16#0060, (* PLC *)
    INDRV_TABLE        := 16#0070, (* IndraDrive *)
    DIAX_TABLE         := 16#0080, (* DIAX *)
    ECO_TABLE          := 16#0090, (* EcoDrive *)
    PB_DP_TABLE        := 16#0130, (* Profibus *)
    DEVICENET_TABLE    := 16#0140, (* DeviceNet *)
    ETHERNET_TABLE     := 16#0150, (* EtherNet *)
    ETHERNET_IP_TABLE  := 16#0151, (* EtherNet/IP ODVA *)
    INTERBUS_TABLE     := 16#0160, (* Interbus *)
    F_RELATED_TABLE    := 16#0170, (* function related *)
    CAN_TABLE          := 16#0180, (* CAN bus *)
    INLINEIO_TABLE     := 16#0190, (* Inline IOs *)
    PROFINETIO_TABLE   := 16#0200, (* ProfinetIO, ProfinetIO Device *)

    USER1_TABLE        := 16#1000, (* free User Table *)
    ...
    USER10_TABLE       := 16#1009 (* free User Table *)
);
END_TYPE
```


3.7 Data types to report date and absolute time specifications for Motion function blocks

3.7.1 Reporting date and absolute time specification, introduction and overview

For the indication of the date and other absolute time designations, there are three structures:

- [SYS_TIME64](#), [page 63](#), system time in microseconds since 01.01.1970, 00:00 o'clock.
- [EXT_SYS_TIME_DATE](#), [page 63](#), system time in microseconds since 01.01.1970, 00:00 o'clock, additional data year, month, day, hours, minutes, seconds, milliseconds and week day are calculated from the system time.
- [EXT_SYS_TIME_DATE](#), [page 63](#), month, day, hours, minutes, seconds, milliseconds and week day, calculated on the basis of the system time in microseconds since 01.01.1970, 00:00 o'clock.

3.7.2 SYS_TIME64

Brief description Structure for the indication of the system time in microseconds since 01.01.1970, 00:00 o'clock.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of times

Tab. 3-36: *SYS_TIME64 reference table*

Type definition

```
(* Structure of system time - Microseconds since 1970 *)
TYPE SYS_TIME64 :
STRUCT
  LowValue:    DWORD;    (* Low value [us]      *)
  HighValue:   DWORD;    (* High value [us]   *)
END_STRUCT
END_TYPE
```

3.7.3 EXT_SYS_TIME_DATE

Brief description Structure to return the system time in microseconds since 01.01.1970, 00:00 o'clock, additional data year, month, day, hours, minutes, seconds, milliseconds and week day are calculated from the system time.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of times

Tab. 3-37: *EXT_SYS_TIME_DATE reference table*

Type definition

```
(* Extended structure of system time *)
TYPE EXT_SYS_TIME_DATE :
STRUCT
  LowValue: DWORD;    (* Low value [us]      *)
  HighValue: DWORD;   (* High value [us]   *)
END_STRUCT
```

```

Year : UINT;          (* Year          *)
Month : UINT;         (* Month         *)
Day : UINT;           (* Day           *)
Hour : UINT;          (* Hour          *)
Minute : UINT;        (* Minute        *)
Second : UINT;        (* Second        *)
Milliseconds : UINT;  (* Milliseconds  *)
DayOfWeek: UINT;      (* Day of week   *)
END_STRUCT
END_TYPE

```

3.7.4 SYS_TIME_DATE

Brief description Structure for the indication of the information year, month, day, hours, minutes, seconds, milliseconds and week day which are calculated from a system time in microseconds since 01.01.1970, 00:00 o'clock.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/data types of times

Tab. 3-38: *SYS_TIME_DATE* reference

Type definition

```

(* Structure of system time *)
TYPE SYS_TIME_DATE :
STRUCT
  Year : UINT;          (* Year          *)
  Month : UINT;         (* Month         *)
  Day : UINT;           (* Day           *)
  Hour : UINT;          (* Hour          *)
  Minute : UINT;        (* Minute        *)
  Second : UINT;        (* Second        *)
  Milliseconds : UINT;  (* Milliseconds  *)
  DayOfWeek: UINT;      (* Day of week   *)
END_STRUCT
END_TYPE

```

3.8 Data types to support touch probe functions

3.8.1 Data types to support touch probe functions, general information

The section contains data types supporting the touch probe functions during the processing in the drive and in the control as well:

- [MB_DIFF_EVAL](#), page 65, setting of the methods for differential measuring
- [MB_EXPECT_WINDOW_MODE](#), page 66, configuration and activation of the expectation window
- [MB_INTERPOLATION_MODE](#), page 66, activation of the interpolation
- [MB_POSITION_EVAL](#), page 67, selection and activation of the trigger signal
- [MB_PROBE_MODULE](#), page 67, selection of the trigger source for the touch probe
- [MB_PROBE_INPUT](#), page 67, selection of the desired input on the selected module
- [MB_PROBE_DATA](#), page 68, selection of the reference signal source of the touch probe

The data belonging to a touch probe function is summarized in the following structure:

- [MB_PROBE_CONFIG](#), page 69

3.8.2 MB_DIFF_EVAL

Brief description The enumeration type "MB_DIFF_EVAL" supports the setting of the methods for difference measuring.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/TouchProbe

Tab. 3-39: MB_DIFF_EVAL reference table

Type definition

```

TYPE MB_DIFF_EVAL :
(
  DIFF_DISABLED      := 0, (* diff.measurement is disabled *)
  DIFF_STANDARD_MODE := 1, (* diff. last two measured values*)
  DIFF_NEG_TO_NEG_EDGE:= 2, (* neg. edge to neg. edge diff. *)
  DIFF_POS_TO_NEG_EDGE:= 3, (* pos. edge to neg. edge diff. *)
  DIFF_NEG_TO_POS_EDGE:= 4, (* neg. edge to pos. edge diff. *)
  DIFF_POS_TO_POS_EDGE:= 5 (* pos. edge to pos. edge diff. *)
);
END_TYPE

```

Functional description The following settings are supported:

- 0 - Difference measurement is switched off, difference signals for ML_TouchProbe and/or ML_TouchProbe_Continuous are not calculated.
- 1 - Difference between the last two measured values, depending on the activated edge.
Example: Both edges are activated, the difference pos/neg edge, then neg/pos edge, pos/neg edge is registered.
- In the following figure, the extended methods 2..5 of the difference measurement are displayed:

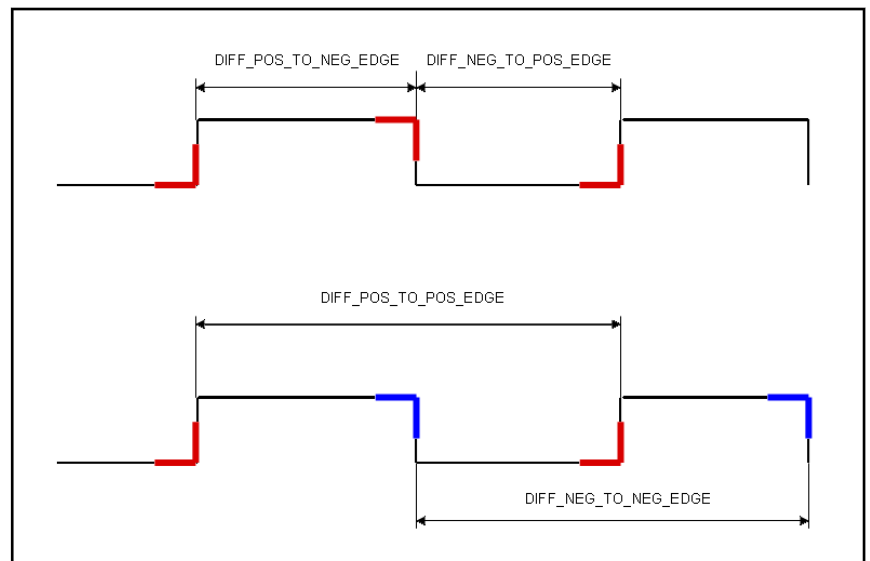


Fig. 3-48: Difference measurement methods

3.8.3 MB_EXPECT_WINDOW_MODE

Brief description The enumeration type "MB_EXPECT_WINDOW_MODE" supports the configuration and activation of the expectation window.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/TouchProbe

Tab. 3-40: MB_EXPECT_WINDOW_MODE reference table

Type definition

```

TYPE MB_EXPECT_WINDOW_MODE :
(
  EXPECT_WINDOW_DISABLED:= 0, (* expectation window disabled *)
  EXPECT_WINDOW_ABS      := 1, (* expectation window absolute *)
  EXPECT_WINDOW_REL      := 2, (* expectation window relative *)
                                (* to "ExpectWindowSetPoint" *)
);
END_TYPE

```

Functional description The user can decide separately for the positive and the negative edge of the touch probe value for an expectation window.

If the user changes the decision for the use of an expectation window, he can define the limits thereof in two different ways:

1. Expectation window disabled:
The respective edge of the touch probe signal is used unfiltered.
2. Expectation window defined absolute:
The start position and the end position of the expectation window have to be indicated as absolute values. The center point value is meaningless in this case.
3. Expectation window defined relative:
The center point value of the expectation window has to be indicated as an absolute value. Starting and end position indicate the distance to the center point value.

3.8.4 MB_INTERPOLATION_MODE

Brief description The enumeration type "MB_INTERPOLATION_MODE" supports the activation of the interpolation.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/TouchProbe

Tab. 3-41: MB_INTERPOLATION_MODE reference table

Type definition

```

TYPE MB_INTERPOLATION_MODE:
(
  NO_INTERPOLATION      := 0, (* no interpolation *)
  LINEAR_INTERPOLATION  := 1, (* linear interpolation *)
  CUBIC_3_INTERPOLATION := 2, (* cubic interpolation with 3 setpoints *)
  CUBIC_4_INTERPOLATION := 3, (* cubic interpolation with 4 setpoints *)
);

```

```
);
END_TYPE
```

3.8.5 MB_POSITION_EVAL

Brief description The enumeration type "MB_POSITION_EVAL" supports the selection and activation of the trigger signal.

Use these enumeration values to select whether the positive, negative or both edges are triggered.



The default value "Zero" blocks triggering!

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/TouchProbe

Tab. 3-42: MB_POSITION_EVAL reference table

Type definition

```
TYPE MB_POSITION_EVAL :
(
  POSITION_EVAL_DISABLED:= 0, (* disabled *)
  POSITION_POS_EDGE      := 1, (* positive edge is evaluated *)
  POSITION_NEG_EDGE      := 2, (* negative edge is evaluated *)
  POSITION_POS_NEG_EDGE := 3 (* pos.and neg. edge are evaluated *)
);
END_TYPE
```

3.8.6 MB_PROBE_MODULE

Brief description The enumeration type "MB_PROBE_MODULE" supports the selection of the trigger source for the touch probe.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/TouchProbe

Tab. 3-43: MB_PROBE_MODULE reference table

Type definition

```
TYPE MB_PROBE_MODULE :
(
  ONBOARD_MODULE      := 0, (* Onboard module MLC, Inputs 1...8 *)
  FUNCTION_MODULE_1:= 1, (* Function module 1 MLC, Inputs 1..16 (CFL-01.1-E2, Fast IO) *)
  FUNCTION_MODULE_2:= 2, (* Function module 2 MLC, Inputs 1..16 (CFL-01.1-E2, Fast IO) *)
  FUNCTION_MODULE_3:= 3, (* Function module 3 MLC, Inputs 1..16 (CFL-01.1-E2, Fast IO) *)
  FUNCTION_MODULE_4:= 4, (* Function module 4 MLC, Inputs 1..16 (CFL-01.1-E2, Fast IO) *)
);
END_TYPE
```

3.8.7 MB_PROBE_INPUT

Brief description The enumeration type "MB_PROBE_INPUT" supports the selection of the desired input on the selected module.



For the OnBoard module, the inputs 1..8 are available.

The Fast I/O modules are provided with the inputs 1..16 as trigger source.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/TouchProbe

Tab. 3-44: MB_PROBE_INPUT reference table

Type definition

```

TYPE MB_PROBE_INPUT :
(
  D_INPUT_1  := 1,  (* digital input 1 *)
  D_INPUT_2  := 2,  (* digital input 2 *)
  D_INPUT_3  := 3,  (* digital input 3 *)
  D_INPUT_4  := 4,  (* digital input 4 *)
  D_INPUT_5  := 5,  (* digital input 5 *)
  D_INPUT_6  := 6,  (* digital input 6 *)
  D_INPUT_7  := 7,  (* digital input 7 *)
  D_INPUT_8  := 8,  (* digital input 8 *)
  D_INPUT_9  := 9,  (* digital input 9 *)
  D_INPUT_10 := 10, (* digital input 10 *)
  D_INPUT_11 := 11, (* digital input 11 *)
  D_INPUT_12 := 12, (* digital input 12 *)
  D_INPUT_13 := 13, (* digital input 13 *)
  D_INPUT_14 := 14, (* digital input 14 *)
  D_INPUT_15 := 15, (* digital input 15 *)
  D_INPUT_16 := 16, (* digital input 16 *)
);
END_TYPE

```

3.8.8 MB_PROBE_DATA

Brief description

The enumeration type "MB_PROBE_DATA" supports the selection of the reference signal source of the touch probe for the touch probes of the IndraLogic XLC/IndraMotion MLC system (A-parameters) and for the touch probes of the drive (S-/P-parameter).



Up to XLC/MLC 13V04, only the parameter "A-0-0100, Actual position value" was supported by the touch probes of the control.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/TouchProbe

Tab. 3-45: MB_PROBE_DATA reference table

Type definition

```

TYPE MB_PROBE_DATA :
(
  TP_ACT_POS_FEEDBACK1 := 0, // S-0-0051 / A-0-0100, Position, feedback 1 value
  TP_ACT_POS_FEEDBACK2 := 1, // S-0-0053 / -----, Position, feedback 2 value
  TP_ACT_POSIN_ACT_CYCLE := 2, // P-0-0753 / -----, Position actual value in actual value cycle
  TP_SYNC_CMD_POS := 3, // P-0-0778 / A-0-2653, Synchronous position command value
  TP_PROBE_TIME := 4, // ----- / -----, time measurement
  TP_ACT_MEASURING_ENC_POS := 5, // P-0-0052 / -----, Actual position value of measuring encoder
  TP_RES_MASTER_AXIS_POS := 6, // P-0-0775 / A-0-2650, Resulting master axis position
  TP_EFF_MASTER_AXIS_POS := 7, // P-0-0776 / A-0-2651, Effective master axis position
  TP_CAM_SHAFT_PROF_ACCESS_ANGLE := 8, // P-0-0227 / A-0-2656, Cam shaft profile access angle
  TP_ACT_MASTER_AXIS_POS := 9, // not yet supported!!!
  // P-0-0053 / A-0-0300, Master axis, actual position value
  TP_ELS_CMD_POS := 10, // ----- / A-0-2655, ELS, position command value
  TP_USER_DATA_CONTAINER_A := 11, // ----- / A-0-0521, Container of actual user-defined value A
);

```

```

TP_USER_DATA_CONTAINER_B := 12, // ----- / A-0-0523, Container of actual user-defined value B
TP_USER_DATA_CONTAINER_C := 13, // ----- / A-0-0525, Container of actual user-defined value C
TP_USER_DATA_CONTAINER_D := 14, // ----- / A-0-0527, Container of actual user-defined value D
TP_USER_DATA_CONTAINER_E := 15, // ----- / A-0-0529, Container of actual user-defined value E
TP_USER_DATA_CONTAINER_F := 16, // ----- / A-0-0531, Container of actual user-defined value F
);
END_TYPE

```

3.8.9 MB_PROBE_CONFIG

Brief description The structure "MB_PROBE_CONFIG" is used for registering the data of a touch probe function.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	RIL_CommonTypes/TouchProbe

Tab. 3-46: MB_PROBE_CONFIG reference table

Type definition

```

TYPE MB_PROBE_CONFIG :
STRUCT
  ModuloValueDrive      :REAL;      (* Modulo value *)
  ProcessingFormat      :BOOL;      (* Weighting position, abs./modulo *)
  ProbePositionEval     :MB_POSITION_EVAL; (* Enabling trigger signal, def.: Off *)
                                (* pos., neg., both edges *)
  DifferenzMeasurment   :MB_DIFF_EVAL; (* difference measurement method, def.: Off *)
  ExpectWindowPosMode   :MB_EXPECT_WINDOW_MODE; (* Expectation window mode pos. edge, *)
                                (* def.: Off, absolute / relative to the setpoint *)
  ExpectWindowNegMode   :MB_EXPECT_WINDOW_MODE; (* Expectation window mode neg. edge, *)
                                (* def.: Off, absolute / relative to the setpoint *)
  ExpectWindowPosStart  :REAL;      (* Start pos. expectation window pos. edge *)
  ExpectWindowPosEnd    :REAL;      (* End pos. expectation window pos. edge *)
  ExpectWindowNegStart  :REAL;      (* Start pos. expectation window neg. edge *)
  ExpectWindowNegEnd    :REAL;      (* End pos. expectation window neg. edge *)
END_STRUCT
END_TYPE

```

"MB_PROBE_CONFIG" uses the following data types

- [MB_POSITION_EVAL, page 67](#)
- [MB_DIFF_EVAL, page 65](#)
- [MB_EXPECT_WINDOW_MODE, page 66](#)

3.8.10 MB_PROBE_CONFIG, special features of Sercos-based axes

Functional description: The following table reflects the connection to the parameter system of the control:

Structure element	Data type	Parameters
ModuloValueDrive	REAL	A-0-0045, Modulo value
ProcessingFormat	BOOL	A-0-0059.7, Scaling factor for velocity data, bit 7 absolute/modulo
ProbePositionEval	MB_POSITION_EVAL, page 67	Activation positive/negative edge, M-0-0007, Control word
Differential measuring	MB_DIFF_EVAL, page 65	Differential measuring, M-0-0007, Control word
ExpectWindowPosMode	MB_EXPECT_WINDOW_MODE, page 66	Activation and configuration of the expectation window of the positive edge M-0-0007, Control word

Structure element	Data type	Parameters
ExpectWindowNegMode	MB_EXPECT_WINDOW_MODE , page 66	Activation and configuration of the expectation window of the negative edge M-0-0007, Control word
ExpectWindowPosStart	REAL	M-0-0202, Starting position of the expectation window, positive edge
ExpectWindowPosEnd	REAL	M-0-0203, End position of the expectation window, positive edge
ExpectWindowNegStart	REAL	M-0-0204, Starting position of the expectation window, negative edge
ExpectWindowNegEnd	REAL	M-0-0205, End position of the expectation window, negative edge

Tab. 3-47: Parameter assignment touch probe function

4 ML_Base.library

4.1 Introduction and overview



The basic function package PLC is sufficient for the library of the IndraLogic XLC/IndraMotion MLC 14VRS system.

The "ML_Base" library contains data types, functions and function blocks required for the implementation of XLC/MLC systems which, in contrast to the RIL_CommonTypes, are only used by the following systems.

- IndraLogic XLC based on the IndraControl Lx5 (L25, L45, L65, L75) with and without Sercos interface
- IndraMotion MLC based on the IndraControl Lx5 (L25, L45, L65, L75)
- IndraMotion MLC based on IndraControl XM2
- IndraMotion MLC based on IndraControl VPx

The data types were extended by functions, function blocks and methods to access the control hardware.

The ML_Base library is supplemented by a hardware-related ML_Base_Constant library with GVL_BaseConstant (VAR_GLOBAL CONSTANT).

- GVL_BaseConstant_xlc_Lx5
- GVL_BaseConstant_mlc_Lx5
- GVL_BaseConstant_xm2x
- GVL_BaseConstant_vpx



Elements other than the elements described may not be used due to the side effects, including malfunctions in the motion sequence of the axes!

Library structure:

Data types

- **Command types (internal use)**
- **Common Types**
 - [MB_IDN on page 75](#), defines the parameter length (system-dependent)
 - Others: Internal use
- **Flex Structs**
 - [MB_CHANGESTEP_MODE on page 76](#), (ENUM), valid change options within a FlexProfile
 - [MB_EVENT_ACTION on page 77](#), actions to be triggered by an event
 - [MB_EVENT_TRIGGER on page 78](#), (ENUM), describes the trigger which should cause a desired action in case of an event
 - [MB_EXECUTION_MODE on page 78](#), (ENUM), one-time or cyclic processing of the FlexProfile
 - [MB_FLEX_EVENT on page 79](#), (STRUCT), configuration of conditions which can be included in a FlexProfile at a specified step

- [MB_MASTER_TYPE on page 79](#), (ENUM), dependence of the motion step, current master axis (MASTER_AXIS0) or time (MASTER_TIME)
- [MB_PROFILE_START_POINT on page 79](#), (ENUM), describes how the origin of the profile has to be interpreted in case of a profile start or profile change. It can be related to the current position or to the zero point for the master and slave axis
- [MB_PROFILE_STEP on page 81](#), (STRUCT), configuration of a motion step in a FlexProfile
- [MB_STEP_TYPE on page 81](#), (ENUM), characteristic of the motion step, fix (STEP_FIX) step or FlexStep
- [MB_SYNC_TYPE on page 82](#), (ENUM), synchronization direction of the axis and type of switching
- **Read/write parameter types**
 - [MB_SERCOS_DATA_STATUS on page 82](#), (ENUM), status representation of a sercos-parameter access
 - Others: Internal use
- **Shared memory types**
 - [ML_AXISDATA_SM on page 110](#), (STRUCT), axis data via shared memory
 - [ML_CONTROLDATA_SM on page 111](#), (STRUCT), control data via shared memory
 - [ML_PLSDATA_SM on page 112](#), (STRUCT), programmable limit switch via shared memory
 - [ML_PROBEDATA_SM on page 113](#), (STRUCT), touch probe data via shared memory

Shared memory types/types

- [ML_LINK on page 113](#), validity bit of the control in the MLC link
- [ML_SERCOS on page 114](#), contains the diagnostic bits of the Sercos interface of the control
- [ML_SYNC on page 114](#), contains four input bits of the control which are exchanged in the MLC link
- [SM_EXT_TYPES on page 115](#), (STRUCT), for the type-correct interpretation of formatted data
- [SM_TYPES on page 114](#), (STRUCT), for the type-correct interpretation of formatted data

Error/diagnostic memory

- [Error/diagnostic memory - supporting user diagnostics and logbook access on page 116](#)
- [E_LOG_SEVERITY on page 121](#), diagnostic system categories of the IndraMotion MLC: Error "F", Warning "E", Message "A"
- [E_LOG_CLASS on page 120](#), classes (error reactions) of the IndraMotion MLC diagnostic system: 0...8 (9)
- [E_LOG_FACILITY on page 121](#), diagnostic groups of the diagnostic system of the IndraMotion MLC: User 16#003B0000
- [ML_ReadDiagLog on page 116](#), retrieves entries from the diagnostic logbook
- [ML_SetAxisDiagEntry on page 117](#), causes an entry in the control logbook and affects the referenced axis (drive) via the "Axis" input

- [ML_SetDiagEntry on page 119](#), causes an entry in the control logbook

Execute

- [ML_TaskExecuteFile on page 122](#), executes a file
- [ML_TaskExecuteGetActive on page 122](#), returns the list of all currently executed files
- [ML_TaskExecuteGetName on page 123](#), returns the name of the executed file
- [ML_TaskExecuteGetStatus on page 123](#), returns the status of an executed file
- [ML_TaskExecuteKill on page 124](#), stops the execution of a file

Global Variable List

- [GVL_Base on page 125](#)
- [GVL_BaseConstant on page 126](#), system-dependent constants of the ML_Base_Constant_xxx libraries.

- **Global Variable Index**

Declaration files for global constants; C2C control link

- [GVL_A_GlobalVarIdx on page 127](#)

- **Parameter ID**

Declaration files for Global Constants

- [GVL_A_Param_ID on page 130](#)
 - [GVL_C_Param_ID on page 130](#)
 - [GVL_M_Param_ID on page 130](#)
 - [GVL_N_Param_ID on page 131](#)
 - [GVL_O_Param_ID on page 131](#)
 - [GVL_P_Param_ID on page 132](#)
 - [GVL_S_Param_ID on page 132](#)

Interface

- [Interfaces, functions and function blocks on page 133](#), (access to Sercos parameters)
 - [IL_SercosAttribute on page 133](#)
 - [IL_SercosElementsToParaNo on page 135](#)
 - [IL_SercosStringToParaNo on page 137](#)

IP settings

- [IP settings - IP configuration on page 138](#)
 - [ML_GetGateway on page 138](#)
 - [ML_GetIpAddress on page 139](#)
 - [ML_GetMacAddress on page 141](#)
 - [ML_GetSubnetMask on page 142](#)
 - [ML_SetGateway on page 143](#)
 - [ML_SetIpAddress on page 144](#)
 - [ML_GetSubnetMask on page 142](#)

Project information

- Library information and access functions, these library components are automatically created for the respective library and do not contain specific parts

SystemTime

- [SystemTime](#) - system time on page 146
 - [IL_SetDateTime](#) on page 146
 - [ML_GetSystemTimeUTC](#) on page 147
 - [ML_SetSystemTimeUTC](#) on page 149
 - [ML_DAYSOFWEEK](#) on page 150

Target

- [Target](#) - Target system on page 151

Functions and function blocks

- [ML_GetCpuLoad](#) on page 151
- [ML_GetDiagnosisLog](#) on page 151
- [ML_GetFirmwareString](#) on page 152
- [ML_GetHardwareInfoCtrl](#) on page 153
- [ML_GetInfoApplication](#) on page 153
- [ML_GetMaxTemperature](#) on page 154
- [ML_GetNewestDiagnosisIndex](#) on page 155
- [ML_GetOhcFan](#) on page 155
- [ML_GetScalingType](#) on page 156
- [ML_GetSystemSpecialPath](#) on page 156
- [ML_GetTargetInfo](#) on page 157
- [ML_GetTemperature](#) on page 157
- [ML_GetWarningCpuLoad](#) on page 158
- [ML_SetWarningCpuLoad](#) on page 158

Data types

- [APP_INFO](#) on page 159
- [E_SCALING_TYPE](#) on page 159
- [E_SYSTEM_SPECIAL_PATH](#) on page 159
- [INFO_APPLICATIONS](#) on page 160
- [ML_DIAGNOSIS_LOG](#) on page 160
- [TARGET_INFO](#) on page 161

Transmission

- [Transmission](#) on page 161

Functions and function blocks

- [ML_SystemGetTransmissionInfo](#) on page 161
- [ML_SystemGetTransmissionStatus](#) on page 162
- [ML_SystemStartTransmission](#) on page 162
- [ML_SystemTftpGetRetransmitCount](#) on page 163
- [ML_SystemTftpGetTimeout](#) on page 163
- [ML_SystemTftpSetRetransmitCount](#) on page 164
- [ML_SystemTftpSetTimeout](#) on page 164

Data types

- [ML_TransmissionInfo](#) on page 164
- [ML_TransmissionStatus](#) on page 165

The ML_Base.library was declared to define a uniform access to the different IndraMotion device types.

The library RIL_CommonTypes.library contains more (system-independent) type definitions.

4.2 Data types

4.2.1 Data types/common types - general types

MB_IDN

Brief description The data type allows to define system-specific parameter lengths. The parameter of IndraLogic XLC/IndraMotion MLC has a width of 64 bit. S- and P-parameters are automatically filled with up to 64 bits.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/CommonTypes

Tab. 4-1: Library assignment
Type definition

```
TYPE
  MB_IDN : LINT ;
END_TYPE
```

Offsets der Parameter

Parameter	Decimal	Hexadecimal
S-0-0000	0	16#0
P-0-0000	32768	16#8000
A-0-0000	4294967296	16#1 0000 0000
C-0-0000	8589934592	16#2 0000 0000
K-0-0000	184683593728	16#2B 0000 0000
M-0-0000	193273528320	16#2D 0000 0000
N-0-0000	197568495616	16#2E 0000 0000
O-0-0000	201863462912	16#2F 0000 0000

Tab. 4-2: Parameter offsets

Program example for MB_IDN in drive and control

As the MB_IDN data type in the drive is of type DINT, the data type has to be converted when filling ident number lists in the drive!

Program example

```
PROGRAM PlcProg
VAR
  aValueWriteP_0_0300:ARRAY[0..10]OF DINT:=[
    LINT_TO_DINT(FP_S_0_0401),
    LINT_TO_DINT(FP_S_0_0402),
    LINT_TO_DINT(FP_P_0_0223),
    LINT_TO_DINT(FP_P_0_0222),
    LINT_TO_DINT(FP_P_0_0222),
    0,0,0,0,0,0];
  fbWriteListParameter:MB_WriteListParameter;
  Exe_WLP:          BOOL;
END_VAR
-----
fbWriteListParameter(
  Execute:= Exe_WLP,
```

```

ParameterNumber:= FP_P_0_0300,
NoOfBytes:= UINT:= SIZEOF(aValueWriteP_0_0300),
ValueAdr:=ADR(aValueWriteP_0_0300),
Axis:= Drive1,
Done=>,
Active=>,
Error=> ,
ErrorID=>,
ErrorIdent=> );

```

4.2.2 Data Types/ Flex Structs

Data types to support Motion and FlexProfile functions, general information

The section contains data types supporting the Motion and FlexProfile functions in the processing in the control:

- [MB_CHANGESTEP_MODE](#), page 76, admissible modification possibilities within a FlexProfile,
- [MB_EVENT_ACTION](#), page 77, actions triggered by an event,
- [MB_EVENT_TRIGGER](#), page 78, describes the trigger which should cause a desired action to be carried out in case of an event.
- [MB_EXECUTION_MODE](#), page 78, one-time or cyclic processing of the FlexProfile,
- [MB_FLEX_EVENT](#), page 79, configuration of conditions which might be included in a FlexProfile at a predetermined step,
- [MB_MASTER_TYPE](#), page 79, dependence of the motion step, current master axis (MASTER_AXIS0) or time (MASTER_TIME),
- [MB_PROFILE_START_POINT](#), page 79, describes how the origin of the profile has to be interpreted in case of a profile start or profile change. It can be related to the current position or to the zero point for the master and slave axis,
- [MB_PROFILE_STEP](#), page 81, configuration of a motion step in a FlexProfile.
- [MB_STEP_TYPE](#), page 81, characteristics of the motion step, fixed (STEP_FIX) step or Flex step.
- [MB_SYNC_TYPE](#), page 82, synchronization direction of the axis and type of changeover.

MB_CHANGESTEP_MODE

Brief description

The enumeration data type MB_CHANGESTEP_MODE describes the admissible modification possibilities in a FlexProfile.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Flex Structs

Tab. 4-3: Library assignment

Type definition

```

TYPE MB_CHANGESTEP_MODE :
(
    CHANGESTEP_EDIT           := 0, (*edit motion step*)
    CHANGESTEP_REMOVE        := 1, (*remove motion step*)
    CHANGESTEP_APPEND        := 2, (*append motion step*)
    CHANGESTEP_INSERT_BEFORE := 3, (*insert motion step before*)
    CHANGESTEP_INSERT_AFTER  := 4, (*insert motion step after*)
);
END_TYPE

```

MB_EVENT_ACTION

Brief description

The enumeration data type MB_EVENT_ACTION describes the desired actions which should trigger an event.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Flex Structs

Tab. 4-4: Library assignment

Type definition

```

TYPE MB_EVENT_ACTION :
(
  ACTION_STATUSBIT0_INSTANT      := 16#0000, (* set bit0 immediately *)
  ACTION_STATUSBIT1_INSTANT      := 16#0001, (* set bit1 immediately *)
  ACTION_STATUSBIT2_INSTANT      := 16#0002, (* set bit2 immediately *)
  ACTION_STATUSBIT3_INSTANT      := 16#0003, (* set bit3 immediately *)
  ACTION_STATUSBIT4_INSTANT      := 16#0004, (* set bit4 immediately *)
  ACTION_STATUSBIT5_INSTANT      := 16#0005, (* set bit5 immediately *)
  ACTION_STATUSBIT6_INSTANT      := 16#0006, (* set bit6 immediately *)
  ACTION_STATUSBIT7_INSTANT      := 16#0007, (* set bit7 immediately *)
  ACTION_STATUSBIT8_INSTANT      := 16#0008, (* set bit8 immediately *)
  ACTION_STATUSBIT9_INSTANT      := 16#0009, (* set bit9 immediately *)
  ACTION_STATUSBIT10_INSTANT     := 16#000A, (* set bit10 immediately *)
  ACTION_STATUSBIT11_INSTANT     := 16#000B, (* set bit11 immediately *)
  ACTION_STATUSBIT12_INSTANT     := 16#000C, (* set bit12 immediately *)
  ACTION_STATUSBIT13_INSTANT     := 16#000D, (* set bit13 immediately *)
  ACTION_STATUSBIT14_INSTANT     := 16#000E, (* set bit14 immediately *)
  ACTION_STATUSBIT15_INSTANT     := 16#000F, (* set bit15 immediately *)

  ACTION_STATUSBIT0_AFTERSTEP    := 16#0100, (* set bit0 at the end of the step *)
  ACTION_STATUSBIT1_AFTERSTEP    := 16#0101, (* set bit1 at the end of the step *)
  ACTION_STATUSBIT2_AFTERSTEP    := 16#0102, (* set bit2 at the end of the step *)
  ACTION_STATUSBIT3_AFTERSTEP    := 16#0103, (* set bit3 at the end of the step *)
  ACTION_STATUSBIT4_AFTERSTEP    := 16#0104, (* set bit4 at the end of the step *)
  ACTION_STATUSBIT5_AFTERSTEP    := 16#0105, (* set bit5 at the end of the step *)
  ACTION_STATUSBIT6_AFTERSTEP    := 16#0106, (* set bit6 at the end of the step *)
  ACTION_STATUSBIT7_AFTERSTEP    := 16#0107, (* set bit7 at the end of the step *)
  ACTION_STATUSBIT8_AFTERSTEP    := 16#0108, (* set bit8 at the end of the step *)
  ACTION_STATUSBIT9_AFTERSTEP    := 16#0109, (* set bit9 at the end of the step *)
  ACTION_STATUSBIT10_AFTERSTEP   := 16#010A, (* set bit10 at the end of the step *)
  ACTION_STATUSBIT11_AFTERSTEP   := 16#010B, (* set bit11 at the end of the step *)
  ACTION_STATUSBIT12_AFTERSTEP   := 16#010C, (* set bit12 at the end of the step *)
  ACTION_STATUSBIT13_AFTERSTEP   := 16#010D, (* set bit13 at the end of the step *)
  ACTION_STATUSBIT14_AFTERSTEP   := 16#010E, (* set bit14 at the end of the step *)
  ACTION_STATUSBIT15_AFTERSTEP   := 16#010F, (* set bit15 at the end of the step *)

  ACTION_STATUSBIT0_AFTERPROFILE := 16#0200, (* set bit0 at the end of the profile *)
  ACTION_STATUSBIT1_AFTERPROFILE := 16#0201, (* set bit1 at the end of the profile *)
  ACTION_STATUSBIT2_AFTERPROFILE := 16#0202, (* set bit2 at the end of the profile *)
  ACTION_STATUSBIT3_AFTERPROFILE := 16#0203, (* set bit3 at the end of the profile *)
  ACTION_STATUSBIT4_AFTERPROFILE := 16#0204, (* set bit4 at the end of the profile *)
  ACTION_STATUSBIT5_AFTERPROFILE := 16#0205, (* set bit5 at the end of the profile *)
  ACTION_STATUSBIT6_AFTERPROFILE := 16#0206, (* set bit6 at the end of the profile *)
  ACTION_STATUSBIT7_AFTERPROFILE := 16#0207, (* set bit7 at the end of the profile *)
  ACTION_STATUSBIT8_AFTERPROFILE := 16#0208, (* set bit8 at the end of the profile *)
  ACTION_STATUSBIT9_AFTERPROFILE := 16#0209, (* set bit9 at the end of the profile *)
  ACTION_STATUSBIT10_AFTERPROFILE := 16#020A, (* set bit10 at the end of the profile *)
  ACTION_STATUSBIT11_AFTERPROFILE := 16#020B, (* set bit11 at the end of the profile *)
  ACTION_STATUSBIT12_AFTERPROFILE := 16#020C, (* set bit12 at the end of the profile *)
  ACTION_STATUSBIT13_AFTERPROFILE := 16#020D, (* set bit13 at the end of the profile *)
  ACTION_STATUSBIT14_AFTERPROFILE := 16#020E, (* set bit14 at the end of the profile *)
  ACTION_STATUSBIT15_AFTERPROFILE := 16#020F, (* set bit15 at the end of the profile *)

  ACTION_STATUSBIT0_INSTANT_NEXTSTEP := 16#0300, (* set bit0 immediately and *)
                                           (* immediately connect next step *)
  ACTION_STATUSBIT1_INSTANT_NEXTSTEP := 16#0301, (* set bit1 immediately and *)
                                           (* immediately connect next step *)
  ACTION_STATUSBIT2_INSTANT_NEXTSTEP := 16#0302, (* set bit2 immediately and *)
                                           (* immediately connect next step *)
  ACTION_STATUSBIT3_INSTANT_NEXTSTEP := 16#0303, (* set bit3 immediately and *)
                                           (* immediately connect next step *)
  ACTION_STATUSBIT4_INSTANT_NEXTSTEP := 16#0304, (* set bit4 immediately and *)
                                           (* immediately connect next step *)
  ACTION_STATUSBIT5_INSTANT_NEXTSTEP := 16#0305, (* set bit5 immediately and *)
                                           (* immediately connect next step *)

```


ML_Base.library

```

ACTION_STATUSBIT6_INSTANT_NEXTSTEP := 16#0306, (* set bit6 immediately and *)
(* immediately connect next step *)
ACTION_STATUSBIT7_INSTANT_NEXTSTEP := 16#0307, (* set bit7 immediately and *)
(* immediately connect next step *)
ACTION_STATUSBIT8_INSTANT_NEXTSTEP := 16#0308, (* set bit8 immediately and *)
(* immediately connect next step *)
ACTION_STATUSBIT9_INSTANT_NEXTSTEP := 16#0309, (* set bit9 immediately and *)
(* immediately connect next step *)
ACTION_STATUSBIT10_INSTANT_NEXTSTEP := 16#030A, (* set bit10 immediately and *)
(* immediately connect next step *)
ACTION_STATUSBIT11_INSTANT_NEXTSTEP := 16#030B, (* set bit11 immediately and *)
(* immediately connect next step *)
ACTION_STATUSBIT12_INSTANT_NEXTSTEP := 16#030C, (* set bit12 immediately and *)
(* immediately connect next step *)
ACTION_STATUSBIT13_INSTANT_NEXTSTEP := 16#030D, (* set bit13 immediately and *)
(* immediately connect next step *)
ACTION_STATUSBIT14_INSTANT_NEXTSTEP := 16#030E, (* set bit14 immediately and *)
(* immediately connect next step *)
ACTION_STATUSBIT15_INSTANT_NEXTSTEP := 16#030F, (* set bit15 immediately and *)
(* immediately connect next step *)

(* Which bit is set results in this case automatically from the event no.: *)
ACTION_STATUSBIT_INSTANT := 16#00FF, (* set bit immediately *)
ACTION_STATUSBIT_AFTERSTEP := 16#01FF, (* set bit at the end of the step *)
ACTION_STATUSBIT_AFTERPROFILE := 16#02FF, (* set bit at the end of the profile *)
ACTION_STATUSBIT_INSTANT_NEXTSTEP := 16#03FF (* set bit immediately and*)
(* immediately connect next step *)

```

```

);
END_TYPE

```

MB_EVENT_TRIGGER

Brief description

Enumeration data type MB_EVENT_TRIGGER describes the trigger that is to lead to the execution of a desired action ([MB_EVENT_ACTION on page 77](#)) during an event.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Flex Structs

Tab. 4-5: Library assignment

Type definition

```

TYPE MB_EVENT_TRIGGER :
(
  TRIGGER_CONTROLBIT_ONLY := 0, (*trigger only on PLC signal*)
  TRIGGER_ABS_SLAVE_REACHED := 1, (*trigger on slave axis position reached*)
  TRIGGER_REL_SLAVE_REACHED := 2, (*trigger on slave axis path traveled*)
  TRIGGER_ABS_MASTER_REACHED := 3, (*trigger on master position reached*)
  TRIGGER_REL_MASTER_REACHED := 4, (*trigger on master path traveled*)
  TRIGGER_REL_TIME_MASTER_REACHED := 5 (*trigger on time out*)
);
END_TYPE

```

MB_EXECUTION_MODE

Brief description

The enumeration data type MB_EXECUTION_MODE describes whether the processing of the FlexProfile has to be carried out once or cyclically.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Flex Structs

Tab. 4-6: Library assignment

Type definition

```

TYPE MB_EXECUTION_MODE:
  EXECUTE_CYCLIC := 0, (*cyclic processing*)
  EXECUTE_SINGLE := 1 (*one-time profile processing*)

```



```
);
END_TYPE
```

MB_FLEX_EVENT

Brief description The structure MB_FLEX_EVENT describes the configuration of conditions which might be included in the FlexProfile at a predetermined step.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Flex Structs

Tab. 4-7: Library assignment

Type definition

```
TYPE MB_FLEX_EVENT :
STRUCT
  StepNumber: INT; (*motion step*)
  TriggerMode: MB_EVENT_TRIGGER; (*selection of trigger mode*)
  TriggerValue: LREAL; (*depends on mode; position or traveling path
                        to be reached *)
  ActionMode: MB_EVENT_ACTION; (* selection of the action mode *)
END_STRUCT
END_TYPE
```

MB_FLEX_EVENT used

- [MB_EVENT_TRIGGER on page 78](#)
- [MB_EVENT_ACTION on page 77](#)

MB_MASTER_TYPE

Brief description The enumeration data type MB_MASTER_TYPE describes the dependence of the motion step. This can either be the current master axis (MASTER_AXIS0) or the time (MASTER_TIME).

Motion steps which are coupled to a time master run independently from the master axis and always in the desired period.

If the time is selected as master, the parameter for the master axis range is not interpreted in degrees but as time specification in milliseconds.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Flex Structs

Tab. 4-8: Library assignment

Type definition

```
TYPE MB_MASTER_TYPE:
(
  MASTER_TIME := 0, (*motion step with time master*)
  MASTER_AXIS0 := 1 (*motion step with axis as master*)
);
END_TYPE
```

MB_PROFILE_START_POINT

Brief description The enumeration data type MB_PROFILE_START_POINT describes in which way the origin of the profile has to be interpreted in case of a profile start or change of profile. This can be related to the current position or the zero point of the master axis as well as the slave axis.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Flex Structs

Tab. 4-9: Library assignment

Type definition

```

TYPE MB_PROFILE_START_POINT:
(
  SLAVE_ORIGIN_MASTER_ORIGIN:  = 0, (* Absolute access to the profile *)
  SLAVE_CURRENT_MASTER_CURRENT: = 1, (* Profile starts at the current master
                                       and slave axis position *)
  SLAVE_ORIGIN_MASTER_CURRENT:  = 2, (* Profile starts at the current master axis
                                       position and the absolute slave axis position *)
  SLAVE_CURRENT_MASTER_ORIGIN:  = 3, (* Profile starts at the current slave axis
                                       position and the absolute master axis position *)
  SLAVE_RELATIVE_MASTER_ORIGIN: = 4, (* Profile starts at the current master axis position.
                                       The slave axis position is defined in such a way
                                       that the axis is traversed relatively.*)
);
END_TYPE

```

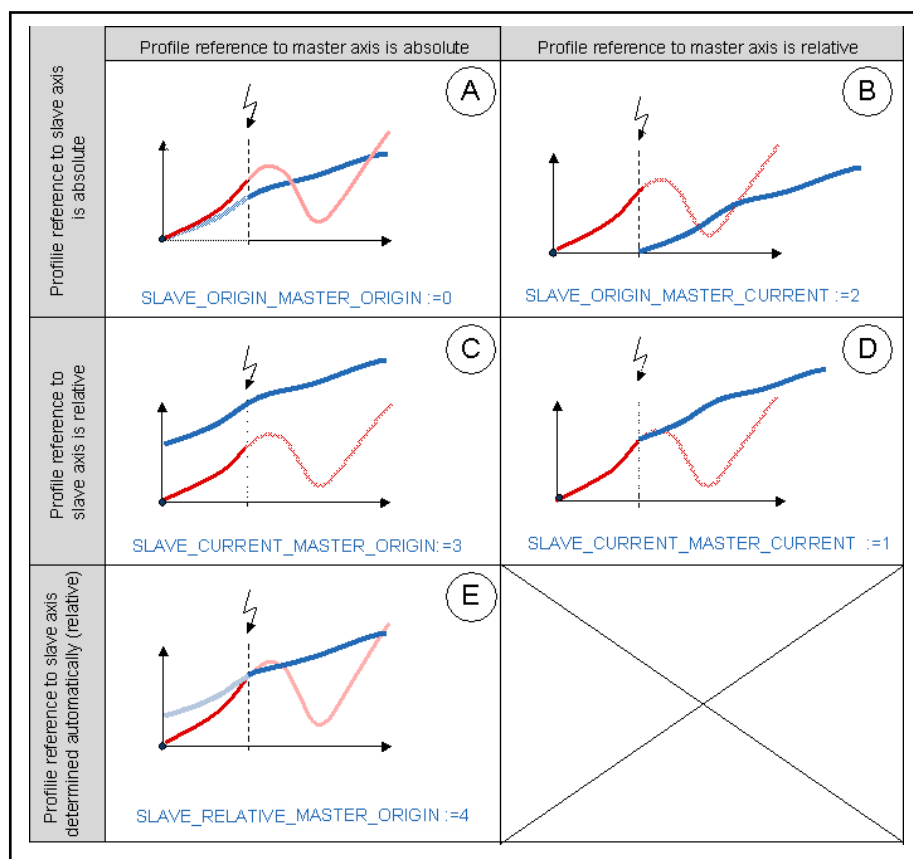


Fig. 4-1: Functional description



Describes the MotionProfile of a fixed position reference between master axis and slave axis; mode A (SLAVE_ORIGIN_MASTER_ORIGIN) is best suited for synchronization. If a slave axis phase offset is not significant and if the slave axis is not supposed to move during synchronization with a master axis at standstill, mode E (SLAVE_RELATIVE_MASTER_ORIGIN) is appropriate here. Subsequently, the profile entry is relative.



For modes B (SLAVE_ORIGIN_MASTER_CURRENT) and D (SLAVE_CURRENT_MASTER_CURRENT), the MotionProfile is always processed from its beginning. This is helpful when a fixed master axis range is not required. In this way feeds or relative motions can be realized from the current axis position.



The position and start of the MotionProfile can be further influenced at the ML_FlexProfile block via the SlaveOffset and MasterOffset inputs. SlaveOffset is used to add a fixed phase offset to the slave axis position (shifting the profile on the Y-axis). With the MasterOffset, the profile entry can be made at a later point in the profile (shifting the profile and the X-axis to the left).

MB_PROFILE_STEP

Brief description

The structure MB_PROFILE_STEP describes the configuration of a motion step in a FlexProfile.

Several motion steps in a row result in a motion set.

These can be written on the axis as a FlexProfile motion set with the function block MB_ChangeFlexProfileSet and finally be started with the Motion function block ML_FlexProfile.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Flex Structs

Tab. 4-10: Library assignment

Type definition

```

TYPE MB_PROFILE_STEP :
STRUCT
  Master:MB_MASTER_TYPE := MASTER_AXIS0; (* master axis source select.:
time or position*)
  Hub:LREAL; (* stroke *)
  Range:LREAL; (* master axis section def. *)
  MotionLaw:MC_STEP_MODE; (* definition of motion law *)
  StepType:MB_STEP_TYPE := STEP_FIX; (* step type: standard(fix. or Flex *)
  StartVel: LREAL := 0.0; (* slave axis start vel. *)
  StartAcc: LREAL := 0.0; (* slave axis start accel. *)
  StartJrk: LREAL := 0.0; (* slave axis start jerk *)
  EndVel: LREAL := 0.0; (* slave axis final vel. *)
  EndAcc: LREAL := 0.0; (* slave axis final accel. *)
  EndJrk: LREAL := 0.0; (* slave axis final jerk *)
  TravelVel:LREAL := 0.0; (* slave axis traveling vel.,
not considered by all motion laws*)
  LimitAcc: LREAL := 0.0; (* slave axis traveling acceleration,
not considered by all motion laws*)
  LimitJrk: LREAL := 0.0; (* slave axis traveling jerk not
considered by all motion laws*)
  TurningPointDisplacement: LREAL := 0.5; (* turning point displ. for symm.
motion laws *)
END_STRUCT
END_TYPE

```

MB_STEP_TYPE

Brief description

The enumeration data type MB_STEP_TYPE describes the characteristic of the motion step. Principally, a difference can be made between fix steps (STEP_FIX) and Flex steps. The Flex steps can furthermore be divided in four subtypes with regard to their master axis and slave axis ranges.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Flex Structs

Tab. 4-11: Library assignment

Type definition

```

TYPE MB_STEP_TYPE:
(
  STEP_FIX           := 0,  (*standard step*)
  STEP_FLEX_REL_REL  := 1,  (*Flex step with rel. distance and rel. master axis section*)
  STEP_FLEX_REL_ABS  := 2,  (*Flex step with rel. distance and abs. master axis section*)
  STEP_FLEX_ABS_REL  := 3,  (*Flex step with abs. slave axis pos. and rel. master axis section*)
  STEP_FLEX_ABS_ABS  := 4   (*Flex step with abs. slave axis pos. and abs. master axis section*)
);
END_TYPE

```

A fixed motion step corresponds to the motion step familiar from other cam solutions, which is specifically defined in its dimensions by its distance and its master axis range. Furthermore, its left and right marginal values for velocity, acceleration and, if necessary, jerk are fixedly determined.

MB_SYNC_TYPE**Brief description**

The enumeration data type MB_SYNC_TYPE defines the synchronization direction of the axis as well as the type of changeover.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Flex Structs

Tab. 4-12: Library assignment

Type definition

```

TYPE MB_SYNC_TYPE:
(
  SYNC_RAMPIN_SHORTEST_WAY := 0,  (*smooth switching shortest way*)
  SYNC_RAMPIN_CATCH_UP     := 1,  (*smooth switching positive direction*)
  SYNC_RAMPIN_SLOW_DOWN    := 2,  (*smooth switching negative direction*)
  SYNC_DIRECT              := 3,  (*hard switching*)
);
END_TYPE

```

4.2.3 Data types/read write parameter - parameter access**MB_SERCOS_DATA_STATUS****Brief description**

The status of a parameter selected by the user is displayed as a clear text and clearly readable with the function block MB_ReadSercosDataStatus, ML_Plcopen.library, with the ENUM type "MB_SERCOS_DATA_STATUS".

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Flex Structs

Tab. 4-13: Library assignment

Type definition

```

TYPE MB_SERCOS_DATA_STATUS:
(* State of a parameter *)
(
  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 impos.*)
MB_SERCOS_DATA_DATUM_FAULT    := 256      (* Data invalid *)
);
END_TYPE

```



The data status "MB_SERCOS_DATA_DATE_FAULT" indicates that the value of the parameter lies beyond its validity limits, i.e. it is smaller than its minimum value or higher than its maximum admissible value.

4.2.4 Data types/shared memory - direct access

Data types to support the data exchange between Motion and PLC

- [Axis data via shared memory and function variable access, page 83](#)
- [Control data via shared memory and function variable access, page 92](#)
- [Programmable limit switch data via shared memory and function variable access, page 97](#)
- [Touch probe data via shared memory and function variable access, page 102](#)
- [Oscilloscope data via function variable access, page 107](#)

More data types

- [ML_LINK, page 113](#), validity bit of the control in the MLC link.
- [ML_SERCOS on page 114](#), contains the diagnostic bits of the Sercos interface of the control
- [ML_SYNC, page 114](#), contains four input bits of the control which are exchanged in the MLC link.
- [SM_EXT_TYPER, page 115](#) (STRUCT), for the type-correct interpretation of formatted data.
- [SM_TYPER, page 114](#) (STRUCT), for the type-correct interpretation of formatted data.

Axis data via shared memory and function variable access

Axis data via shared memory and function variable access, general

The axis data saved in the A-parameters can be accessed by the PLC either via a simplified, blocking instruction "Function variable access" (alias "direct variable") or via the shared memory.

In the Motion mode "Operating mode", axis data in shared memory is consistently and cyclically updated. Due to the time required for updating, the scope of this data is limited. The actual values of the axis data in the shared memory are always updated when the Motion kernel MK1 has completed its process. The command values are always transferred when the Motion kernel MK2 has completed its process.

Axis data via function variable access is updated irrespective of the Motion mode at the access time.

Part of the axis data is available in shared memory as well as via the function variable access. As axis data in shared memory is cyclically exchanged irrespective of its use, further redundant access via function variable access should be avoided due to performance reasons.



The simplified, blocking access to a function variable is an implicit function call and requires more runtime resources than a simple memory access to a local PLC variable.

It is recommended to avoid accessing a function variable several times and to buffer the operating data to a local PLC variable if required.

Via shared memory and function variable access, axis data can be distributed across several Motion cycles (command value generation, actual value provision) depending on the task properties of the POU and are therefore not necessarily consistent among each other.



A function variable is not part of the process image of the PLC task.

The simplified, blocking access to a function variable is an implicit function call which is executed at the moment of calling.

Multiple access to a function variable can cause different results in a PLC program (POU) as the operating data can be modified due to task changes.

Axis data can also be exchanged via a C2C link between several controls.

The operating data synchronized in the C2C link of this parameter can be accessed via a simplified, blocking instruction "function variable access" (alias "direct variable").

Shared memory

Axis data in the shared memory is declared as a global array "AxisData[]" of the data structure [ML_AXISDATA_SM, page 110](#). The definition is part of the global variable list "GVL_Base" in the "ML_Base" library.

The axes are numbered and the axis name of the individual axis is not important for this access type.¹⁾

Axis data in the shared memory:

```
VAR_GLOBAL
...
AxisData AT %MBaaa: ARRAY[MB_MIN_AXIS_NUMBER..MB_MAX_AXIS_NUMBER] OF ML_AXISDATA_SM;
...
END_VAR
```



For the addresses of "AxisData", "ControlData", "PlsData" and "ProbeData", refer to the library manager of the BASELIB= ML_Base / Global Variables List / GVL_Base.

The array limits are located in the system-specific library parts BASELIBCONST= ML_Base_Constant_xxx / Global Variable Lists / GVL_BaseConstant.

Function variable access

Function variables of the axis data are declared as the global array "DV_Axis[]" of the function block "ML_DirectVarAxis". The definition is part of the global variable list "GVL_Base" in the "ML_Base" library, refer to [DV_Axis\[\] - axis data via function variable access, page 87](#).

Function variables of the axis data:

```
VAR_GLOBAL
...
DV_Axis: ARRAY [MB_MIN_AXIS_NUMBER..MB_MAX_AXIS_NUMBER] OF ML_DirectVarAxis;
```

¹⁾ MB_MAX_AXIS_NUMBER, maximum number of admissible axes, currently 64, control type-dependent.

```
...
END_VAR
```



If a function variable access is available for a parameter, it is noted in the corresponding parameter description (head of parameter description, direct access line, example A-0-0000).

Synchronous function variable access in the C2C link

Synchronous function variables in the C2C link of the axis data are declared as global array "GV_LinkAxis[]". The definition is part of the global variable list "GVL_Base" in the "ML_Base" library, refer to [GV_LinkAxis\[\] - axis data via synchronous function variable access in C2C link](#), page 90.

Synchronous function variables in the C2C link of the axis data:

```
VAR_GLOBAL
...
GV_LinkAxis: ARRAY[MB_MIN_GLOBAL_CONTROL_NUMBER..MB_MAX_GLOBAL_CONTROL_NUMBER,
                  MB_MIN_AXIS_NUMBER..MB_MAX_AXIS_NUMBER] OF ML_GlobalVarAxis;
...
END_VAR
```



If the synchronous function variable access is available to a parameter in the C2C link, it is described in the corresponding parameter description (head of parameter description, line C2C access, example A-0-0021).

AxisData[] - axis data in shared memory

Brief description

When a new axis is created in IndraWorks, a device name and a drive address are queried and an axis number is automatically assigned to the axis.

Using the automatically generated axis number, axis data can be accessed in a subscripted way

- via the function variable access („DV_Axis[]“),
- if the control enabled the axis for a C2C link, it can be accessed via function variable access ("GV_LinkAxis[]") and
- shared memory ("AxisData[]")



In the Motion mode "Operating mode", axis data in shared memory is consistently and cyclically updated. Due to the time required for updating, the scope of this data is limited. Axis data as direct variables is updated irrespective of the Motion mode at the point in time when it is accessed.

Via shared memory and function variables (alias "direct variables"), axis data can be distributed across several Motion cycles (command value generation, actual value provision) depending on the task properties of the POU and are therefore not necessarily consistent among each other.

Axis data in the shared memory (flag area) is declared as global array "AxisData[]" of the data structure "ML_AXISDATA_SM". The definition is part of the global variable list "GVL_Base" in the "ML_Base" library.

Definition of the global array "AxisData[]"

```
VAR_GLOBAL
...
AxisData AT %Mbaa: ARRAY[MB_MIN_AXIS_NUMBER..MB_MAX_AXIS_NUMBER] OF ML_AXISDATA_SM;
...
END_VAR
```



For the addresses of "AxisData", "ControlData", "PlsData" and "ProbeData", refer to the library manager of the BASELIB= ML_Base / Global Variables List / GVL_Base.

The array limits are located in the system-specific library parts BASELIBCONST= ML_Base_Constant_xxx / Global Variable Lists / GVL_BaseConstant.

Also refer to

- [ML_AXISDATA_SM, page 110](#)

PLC access to parameter values via Shared Memory

The reading access on the data in the shared memory can be carried out via the following equivalent instructions (ST).

Reading Access on the Data in the Shared Memory

```
MyVar:= AxisData[MyAxisName.AxisNo].FlexProfil_Active;
MyVar:= AxisData[Index].FlexProfil_Active;
MyVar:= AxisData[1].FlexProfil_Active;
```

The writing access on the data in the shared memory can be carried out via the following equivalent instructions (ST).

Writing Access on the Data in the Shared Memory

```
AxisData[MyAxisName.AxisNo].dwFlexProfileEventControlBits_q:= MyVar;
AxisData[Index].dwFlexProfileEventControlBits_q:= MyVar;
AxisData[1].dwFlexProfileEventControlBits_q:= MyVar;
```

The IntelliSense of a variable in the shared memory contains the parameter ident number and if necessary the bit, the access possibilities (RD or RW), the parameter names and the meaning of the bit.

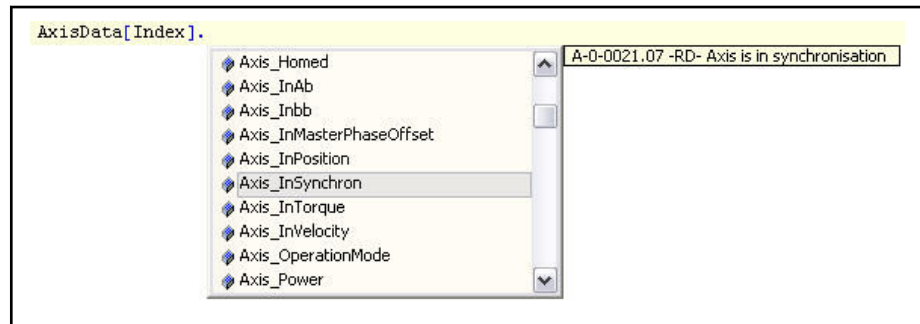


Fig. 4-2: IntelliSense of a variable in the shared memory



Variables of an axis in the shared memory can also be accessed via an user-defined designation by using references.

User-defined Designation by Using References

```
VAR
  MyAxisName: REFERENCE TO ML_AXISDATA_SM:= AxisData[MyAxisName.AxisNo];
END_VAR

MyVar:= MyAxisName.FlexProfil_Active;
MyAxisName.dwFlexProfileEventControlBits_q:= MyVar;
```

Selected ranges of the data structure ML_AXISDATA_SM:

```
dwUserCmdDataA_q :SM_TYPES; (* A-0-0501 -RW- Container user defined commanded data A *)
dwUserCmdDataB_q :SM_TYPES; (* A-0-0503 -RW- Container user defined commanded data B *)
dwUserCmdDataC_q :SM_TYPES; (* A-0-0505 -RW- Container user defined commanded data C *)
dwUserCmdDataD_q :SM_TYPES; (* A-0-0507 -RW- Container user defined commanded data D *)
```



```

dwUserCmdDataE_q :SM_TYPES; (* A-0-0509 -RW- Container user defined commanded data E *)

dwUserActualDataA_i :SM_TYPES; (* A-0-0521 -RD- Container user defined actual data A *)
dwUserActualDataB_i :SM_TYPES; (* A-0-0523 -RD- Container user defined actual data B *)
dwUserActualDataC_i :SM_TYPES; (* A-0-0525 -RD- Container user defined actual data C *)
dwUserActualDataD_i :SM_TYPES; (* A-0-0527 -RD- Container user defined actual data D *)
dwUserActualDataE_i :SM_TYPES; (* A-0-0529 -RD- Container user defined actual data E *)
dwUserActualDataF_i :SM_TYPES; (* A-0-0531 -RD- Container user defined actual data F *)

```

The "dwUserCmdData*_q" and "dwUserActualData*_i" data containers contain information on the "cyclic sercos channel".

Depending on the configuration, the containers may contain values of various data types.



The data types of the A-parameters to be transmitted are in the parameter description (attributes, PLC data type).

The PLC data types are to be determined from the attributes of the respective parameters for S-parameters and P-parameters:

- Decimal places ≠ 0, type "REAL"
- Decimal places = 0 and FLOAT, type "REAL"
- Decimal places = 0 and
 - 4 bytes with sign, "DINT"
 - 2 bytes with sign, "INT"
 - 4 bytes without sign, "UDINT"
 - 2 bytes without sign, "DINT"

However, the point operator can be used to access the respective data type.

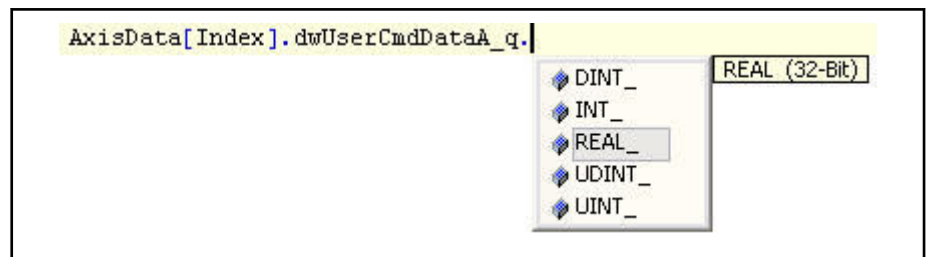


Fig. 4-3: Data type specification via point operator

Also refer to

- [SM_TYPES](#), page 114
- [ML_AXISDATA_SM](#), page 110

DV_Axis[] - axis data via function variable access

Brief description

When a new axis is created in IndraWorks, a device name and a drive address are queried. The axis is automatically assigned to an axis number MB_MIN_AXIS_NUMBER..MB_MAX_AXIS_NUMBER ([GVL_BaseConstant](#), page 126).

Using the automatically generated axis number, axis data can be accessed in a subscripted way

- via the function variable access („DV_Axis[]“),
- if the control enabled the axis for a C2C link, it can be accessed via function variable access ("GV_LinkAxis[]") and

- shared memory ("AxisData[]")



In the Motion mode "Operating mode", axis data in shared memory is consistently and cyclically updated. Due to the time required for updating, the scope of this data is limited.

Axis data via function variable access is updated irrespective of the Motion mode at access time.

Via the function variable access, axis data can be distributed across several Motion cycles (command value generation, actual value provision) depending on the task properties of the POU and are therefore not necessarily consistent among each other.

Axis data via function variable access is declared as global array "DV_Axis[]" of the function block "ML_DirectVarAxis". The definition is part of the global variable list "GVL_Base" in the "ML_Base_xxx_yyy" library.

The axis data (operating data of the parameter) can be accessed via a simplified, blocking instruction "function variable access" (alias "direct variable"). A function variable call is executed as blocking and may require different periods according to its characteristics such as the axis type.

The function block "ML_DirectVarAxis" provides direct variables as methods. The existence of the GET and SET properties for the respective method defines the access option. Executing the write-access SET on a function variable depends for example on the axis type and on the Motion mode and does not generate an error message in a negative case.



The function block "ML_DirectVarAxis" is not intended for direct use in the user program.

Extract from GVL_Base

```
VAR_GLOBAL
...
DV_Axis: ARRAY [MB_MIN_AXIS_NUMBER..MB_MAX_AXIS_NUMBER] OF ML_DirectVarAxis;
...
END_VAR
```

PLC access to parameter values function variable access

The read-only access to a function variable can be executed via the following equivalent instructions (ST) if the respective method is equipped with a GET property (READ).

Read-only access to a function variable

```
MyVar:= DV_Axis[MyAxisName.AxisNo].A_0_0004;
MyVar:= DV_Axis[Index].A_0_0004;
MyVar:= DV_Axis[1].A_0_0004;
```

The write-access to a function variable can be executed via the following equivalent instructions (ST) if the respective method is equipped with a SET property (WRITE).

Write access to a function variable

```
DV_Axis[MyAxisName.AxisNo].A_0_0216:= MyVar;
DV_Axis[Index].A_0_0216:= MyVar;
DV_Axis[1].A_0_0216:= MyVar;
```

The IntelliSense of a function variable contains the parameter ident number, the access possibilities (READ or READ/WRITE), the parameter names as well as the method to determine the current access state.

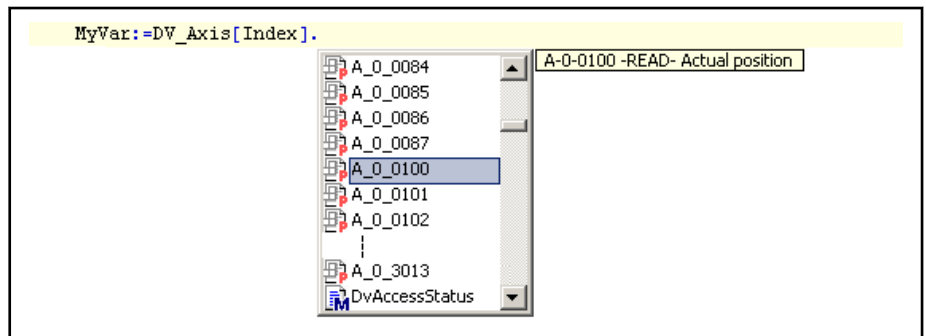


Fig. 4-4: Function variable access using the IntelliSense functionality



Function variables of an axis can also be accessed via an user-defined designation by using references.

Access via REFERENCE_TO

```
VAR
  MyAxisName: REFERENCE_TO ML_DirectVarAxis:= DV_Axis[MyAxisName.AxisNo];
END_VAR

MyVar:= MyAxisName.A_0_0004;
MyAxisName.A_0_0216:= MyVar;
```

Status during function variable access

In addition to properties for reading and writing, the function blocks for function variable access also offer a way to determine the **current access status** of a parameter using the **DvAccessStatus** method.

The corresponding parameter constant is transferred to the method.

The method returns the return value of the last access (== current access status).

The current access status of a function variable is read according to the following statement (ST):

Read: current access status of a function variable

```
A_Status:= DV_Axis[MyAxisName.AxisNo].DvAccessStatus(FP_A_0_0000);
C_Status:= DV_Control[0].DvAccessStatus(FP_C_0_0023);
M_Status:= DV_Probe[MB_MIN_PROBE_NUMBER].DvAccessStatus(FP_M_0_0021);
N_Status:= DV_Pls[Index].DvAccessStatus(FP_N_0_0303);
O_Status:= DV_Osci[Index].DvAccessStatus(FP_O_0_0021);
```

Declaration files for Global Constants

- [GVL_A_Param_ID, page 130](#)
- [GVL_C_Param_ID, page 130](#)
- [GVL_M_Param_ID, page 130](#)
- [GVL_N_Param_ID, page 131](#)
- [GVL_O_Param_ID, page 131](#)

Status (extract, for further information, see SERCOS_TABLE)

- 16# 0000.0000: No error
- 16# F02D1001: IDN parameter not available

- 16# F02Dxxxx: sercos error codes
- 16# F02D7013: command cannot be executed (FW: command not implemented)
- 16# F02D8001: access to parameter currently not possible (BUSY).

GV_LinkAxis[] - axis data via synchronous function variable access in C2C link

Brief description

When a new axis is created in IndraWorks, a device name and a drive address are queried and an axis number is automatically assigned to the axis.

If the control operates in a C2C link, the user control as well as third-party controls can be accessed via "Synchronous function variable access in C2C link" by means of an automatically generated axis number subscripted to axis data ("GV_LinkAxis[]").

Axis data is declared via "Synchronous function variable access in a C2C link" as global array "GV_LinkAxis[]" of the "ML_GlobalVarAxis" function block. The definition is part of the global variable list "GVL_Base" in the "ML_Base" library.

The axis data (operating data of the respective parameter) synchronized in the C2C link of this parameter can be accessed via a simplified, blocking instruction "function variable access" (alias "direct variable"). A function variable call is executed as blocking and can require different time periods according to its characteristics.

The function block "ML_GlobalVarAxis" provides direct variables as methods. The existence of the GET (and SET) properties for the respective method defines the access option. Executing the write access SET to a function variable does not generate an error message in a negative case.



The function block "ML_GlobalVarAxis" is not intended for direct use in the user program.

Extract from GVL_Base

```
VAR_GLOBAL
```

```
...
GV_LinkAxis: ARRAY[MB_MIN_GLOBAL_CONTROL_NUMBER..MB_MAX_GLOBAL_CONTROL_NUMBER,
                  MB_MIN_AXIS_NUMBER..MB_MAX_AXIS_NUMBER] OF ML_GlobalVarAxis;
...
END_VAR
```

PLC access to the parameter values with function variable access synchronously in the C2C link

The read-only access to a function variable synchronously in the C2C link can be executed via the following equivalent instructions (ST) if the respective method is equipped with a GET property (READ).

Program:

```
MyVar:= GV_LinkAxis[CtrlNo, AxisNo].A_0_0021;
MyVar:= GV_LinkAxis[0, 1].A_0_0021;
```

The indexed, blocked access with CtrlNo equal to LOCAL_CTRL accesses the synchronized operating data of the local control located in the C2C link. This access is equivalent to the indexed, blocked access with CtrlNo equal to C-0-0031.

The write access to a function variable synchronous in the C2C link is not supported.

The IntelliSense of a function variable synchronous in the C2C variable contains the parameter ident number, the access possibilities (READ or READ/ WRITE), the parameter names as well as the method to determine the current access state.

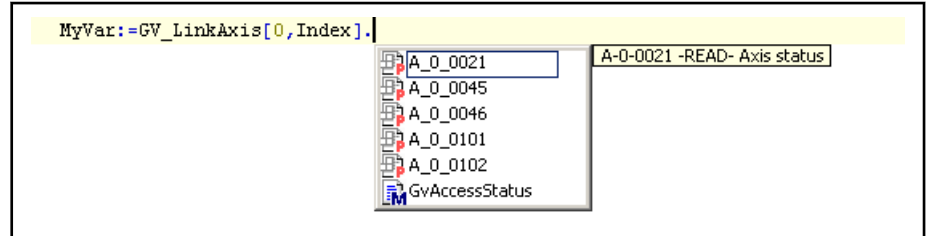


Fig. 4-5: Function variable access synchronous in the C2C link using the IntelliSense functionality



Function variables of an axis can also be accessed via an user-defined designation by using references.

Access via REFERENCE_TO

```
VAR
  MyAxisName: REFERENCE TO ML_GlobalVarAxis:= GV_LinkAxis[CtrlNo, AxisNo];
END_VAR

MyVar:= MyAxisName.A_0_0021;
```

Status during synchronous function variable access in the C2C link

The function blocks for "Synchronous function variable access in C2C link" also offer the option, apart from the properties for reading (and writing) of the operating data, to determine the **current access status** of a parameter using the **GvAccessStatus** method.

The corresponding parameter constant is transferred to the method.

The method returns the return value of the last access (== current access status).

The current access status of a function variable is read according to the following statement (ST):

Read: current access status of synchronous function variable in the C2C link

```
Status:= GV_LinkAxis[CtrlNo, AxisNo].GvAccessStatus(GV_A_0_0021);
Status:= GV_LinkAxis[0, 1].GvAccessStatus(GV_A_0_0021);
```

GV_A_0_xxxx are constants of the Global Variable List/Global Variable Index / [GVL_A_GlobalVarIdx](#), page 127.

Status (extract, for further information, see SERCOS_TABLE)

- 16# 0000.0000: No error
- 16# F02D1001: IDN parameter not available
- 16# F02Dxxxx: SERCOS error codes
- 16# F02D7013: command cannot be executed (FW: command not implemented)
- 16# F02D8001: access to parameter currently not possible (BUSY).

Control data via shared memory and function variable access

Control data via shared memory and function variable access, general

The control data saved in the C-parameters can be accessed by the PLC either via a simplified, blocking instruction "Function variable access" (alias "direct variable") or via the shared memory.

In the Motion mode "Operating mode", control data in shared memory is consistently and cyclically updated. Due to the time required for updating, the scope of this data is limited. The actual values of the control data in shared memory are always updated when the Motion kernel MK1 has completed its process. The command values are always transferred when the Motion kernel MK2 has completed its process.

Control data via function variable access is updated irrespective of the Motion mode at the access time.

Part of the control data is available in shared memory as well as via function variable access. Since control data in the shared memory is cyclically exchanged irrespective of its use, further redundant access via direct variables should be avoided due to performance reasons.



The simplified, blocking access to a function variable is an implicit function call and requires more runtime resources than a simple memory access to a local PLC variable.

It is recommended to avoid accessing a function variable several times and to buffer the operating data to a local PLC variable if required.

Via shared memory and function variable access, control data can be distributed across several Motion cycles (command value generation, actual value provision), depending on the task properties of the POU and are therefore not necessarily consistent among each other.



A function variable is not part of the process image of the PLC task.

The simplified, blocking access to a function variable is an implicit function call which is executed at the moment of calling.

Multiple access to a function variable can cause different results in a PLC program (POU) as the operating data can be modified due to task changes.

Shared memory

Control data in the shared memory is declared as a global array "ControlData[]" of the data structure [ControlData\[\] - control data in shared memory, page 93](#). The definition is part of the global variable list "GVL_Base" in the "ML_Base" library.

The control name is not important for this access type. The data of the local control is always provided.

Control data in the shared memory

```
VAR_GLOBAL
...
ControlData AT %MBbbb:ARRAY
[MB_MIN_CONTROL_NUMBER..MB_MAX_CONTROL_NUMBER]
OF ML_CONTROLDATA_SM;
...
END_VAR
```



For the addresses of "AxisData", "ControlData", "PlsData" and "ProbeData", refer to the library manager of the BASELIB= ML_Base / Global Variables List / GVL_Base.

The array limits are located in the system-specific library parts BASELIBCONST= ML_Base_Constant_xxx / Global Variable Lists / GVL_BaseConstant.

Function variable access

Function variables of the control data are declared as global array "DV_Control[]" of the function block "ML_DirectVarAxis". The definition is part of the global variable list "GVL_Base" in the "ML_Base_MLC/ML_Base_XLC" library, refer to [DV_Control\[\] - control data via function variable access](#), page 94.

Function variables of the control data

```
VAR_GLOBAL
...
DV_Control: ARRAY[MB_MIN_CONTROL_NUMBER..MB_MAX_CONTROL_NUMBER] OF
ML_DirectVarControl;
...
END_VAR
```



If a parameter can use function variable access, it is described in the respective parameter description (parameter attributes, direct access, example C-0-0000).

Brief description

ControlData[] - control data in shared memory

When creating a new project in IndraWorks, control data is automatically provided. By means of a fixedly defined index MB_MIN_CONTROL_NUMBER or MB_MAX_CONTROL_NUMBER, control data can be accessed via

- function variable access ("DV_Control[]") and
- shared memory ("ControlData[]")



In the Motion mode "Operating mode", control data in shared memory is consistently and cyclically updated. Due to the time required for updating, the scope of this data is limited. Control data as direct variables is updated irrespective of the Motion mode at the access time.

Control data in the shared memory and function variable access can be distributed across several Motion cycles (command value generation, actual value provision) depending on the task properties of the POU and are therefore not necessarily consistent among each other.

Control data in the shared memory is declared as a global array "ControlData[]" of the data structure "ML_CONTROLDATA_SM". The definition is part of the global variable list "GVL_Base" in the "ML_Base" library.

Defining the global array "ControlData[]"

```
VAR_GLOBAL
...
ControlData AT %MBbbb:ARRAY
[MB_MIN_CONTROL_NUMBER..MB_MAX_CONTROL_NUMBER]
OF ML_CONTROLDATA_SM;
...
END_VAR
```



For the addresses of "AxisData", "ControlData", "PlsData" and "ProbeData", refer to the library manager of the BASELIB= ML_Base / Global Variables List / GVL_Base.

The array limits are located in the system-specific library parts BASELIBCONST= ML_Base_Constant_xxx / Global Variable Lists / GVL_BaseConstant.

Functional description

The read-only access to data in the shared memory can be carried out via the following equivalent statements (ST).

Read-only access to data

```
MyVar:= ControlData[MB_MIN_CONTROL_NUMBER].LINK_ESTABLISHED;
MyVar:= ControlData[Index].LINK_ESTABLISHED;
MyVar:= ControlData[0].LINK_ESTABLISHED;
```

The write access to data in the shared memory can be carried out via the following equivalent statements (ST).

Write access to data

```
ControlData[Index].LINK_REBUILD_RING:= MyVar;
ControlData[1].LINK_REBUILD_RING:= MyVar;
```

The IntelliSense of a variable in the shared memory contains the parameter ident number and if necessary the bit, access possibilities (RD or RW), parameter names and bit description.

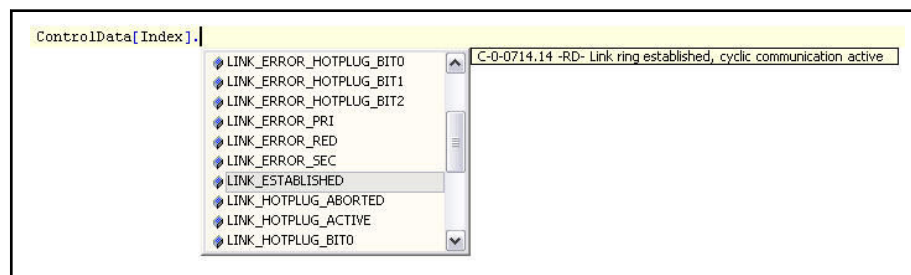


Fig. 4-6: IntelliSense of a variable in the shared memory



Control variables in the shared memory can also be accessed by using references via a user-defined name.

Program:

```
VAR
    MyControlName: REFERENCE TO ML_CONTROLDATA_SM:= ControlData[MB_MIN_CONTROL_NUMBER];
END_VAR

MyVar:= MyControlName.LINK_ESTABLISHED;
MyControlName.LINK_REBUILD_RING:= MyVar;
```

See also:

- [ML_CONTROLDATA_SM, page 111](#)

DV_Control[] - control data via function variable access

Brief description

When creating a new project in IndraWorks, control data is automatically provided. Using the fixedly defined index MB_MIN_CONTROL_NUMBER ... MB_MAX_CONTROL_NUMBER ([GVL_BaseConstant, page 126](#)) control da-

ta can be accessed via function variables ("DV_Control[]") and shared memory ("ControlData[]").



In the Motion mode "Operating mode", control data in shared memory is consistently and cyclically updated. Due to the time required for updating, the scope of this data is limited.

Control data via function variable access (alias "direct variables") is updated irrespective of the Motion mode at the access time.

Via direct variables, control data can be distributed across several Motion cycles (command value generation, actual value provision) depending on the task properties of the POU and are therefore not necessarily consistent among each other.

Control data via function variable access are declared as global array "DV_Control[]" of the function block "ML_DirectVarAxis". The definition is part of the global variable list "GVL_Base" in the "ML_Base" library.

The function block "ML_DirectVarControl" provides direct variables as methods. The existence of the GET and SET properties for the respective method defines the access option. Executing the write-access SET on a function variable depends on Motion mode and does not generate an error message in a negative case.



The function block "ML_DirectVarControl" is not intended for direct use in the user program.

Defining the global array "DV_Control[]"

```
VAR_GLOBAL
...
  DV_Control: ARRAY[MB_MIN_CONTROL_NUMBER..MB_MAX_CONTROL_NUMBER] OF ML_DirectVarControl;
...
END_VAR
```

Functional description

The read-only access to a function variable can be executed via the following equivalent statements (ST) if the respective method is provided with a GET property (READ).

Read-only access to data

```
MyVar:= DV_Control[MB_MIN_CONTROL_NUMBER].C_0_0023;
MyVar:= DV_Control[1].C_0_0023;
```

The write access to a function variable can be executed via the following equivalent statements (ST) if the respective method is provided with a SET property (WRITE).

Write access to data

```
DV_Control[MB_MIN_CONTROL_NUMBER].C_0_0450:= MyVar;
DV_Control[1].C_0_0450:= MyVar;
```

The IntelliSense of a function variable contains the parameter ident number, the access options (READ or READ/WRITE), the parameter names as well as the method to determine the current access status.

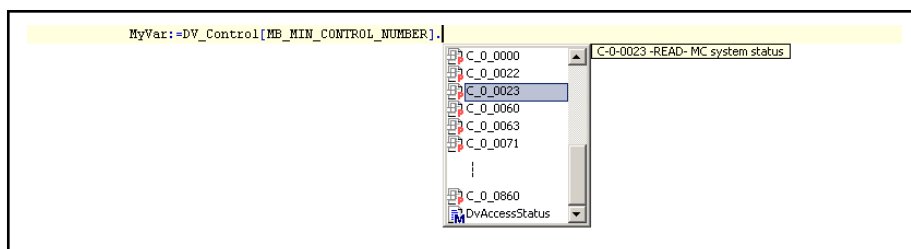


Fig. 4-7: Function variable access using the IntelliSense functionality



Function variables of a control can also be accessed by using references via a user-defined name.

User-defined name by referencing

```
VAR
  MyControlName: REFERENCE TO ML_DirectVarControl:= DV_Control[MB_MIN_CONTROL_NUMBER];
END_VAR
```

```
MyVar:= MyControlName.C_0_0023;
MyControlName.C_0_0450:= MyVar;
```

Status during function variable access

In addition to properties for reading and writing, the function blocks for function variable access also offer a way to determine the **current access status** of a parameter using the **DvAccessStatus** method.

The corresponding parameter constant is transferred to the method.

The method returns the return value of the last access (== current access status).

The current access status of a function variable is read according to the following statement (ST):

Read: current access status of a function variable

```
A_Status:= DV_Axis[MyAxisName.AxisNo].DvAccessStatus(FP_A_0_0000);
C_Status:= DV_Control[0].DvAccessStatus(FP_C_0_0023);
M_Status:= DV_Probe[MB_MIN_PROBE_NUMBER].DvAccessStatus(FP_M_0_0021);
N_Status:= DV_Pls[Index].DvAccessStatus(FP_N_0_0303);
O_Status:= DV_Osci[Index].DvAccessStatus(FP_O_0_0021);
```

Declaration files for Global Constants

- [GVL_A_Param_ID](#), page 130
- [GVL_C_Param_ID](#), page 130
- [GVL_M_Param_ID](#), page 130
- [GVL_N_Param_ID](#), page 131
- [GVL_O_Param_ID](#), page 131

Status (extract, for further information, see SERCOS_TABLE)

- 16# 0000.0000: No error
- 16# F02D1001: IDN parameter not available
- 16# F02Dxxxx: sercos error codes
- 16# F02D7013: command cannot be executed (FW: command not implemented)
- 16# F02D8001: access to parameter currently not possible (BUSY).

Programmable limit switch data via shared memory and function variable access

Programmable limit switch data via shared memory and function variable access, general



N-parameters are not supported by the IndraLogic XLC.

The programmable limit switch data saved in the N-parameters can be accessed by the PLC either via a simplified, blocking instruction "Function variable access" (alias "direct variable") or via the shared memory.

In the Motion mode "Operating mode", programmable limit switch data in shared memory is consistently and cyclically updated. Due to the time required for updating, the scope of this data is limited. The actual values of the programmable limit switch data in the shared memory are always updated when the Motion kernel MK1 has completed its process. The command values are always transferred when the Motion kernel MK2 has completed its process.

PLS data via function variable access is updated irrespective of the Motion mode at the access time.

A part of the PLS data is available in the shared memory as well as for the access via direct variables. Since PLS data in the shared memory is cyclically exchanged irrespective of its use, further redundant access via function variables should be avoided due to performance reasons.



The simplified, blocking access to a function variable is an implicit function call and requires more runtime resources than a simple memory access to a local PLC variable.

It is recommended to avoid accessing a function variable several times and to buffer the operating date to a local PLC variable if required.

Via shared memory and function variable access, PLS data can be distributed across several Motion cycles (command value generation, actual value provision) depending on the task properties of the POU and are therefore not necessarily consistent among each other.



A function variable is not part of the process image of the PLC task.

The simplified, blocking access to a function variable is an implicit function call which is executed at the moment of calling.

Multiple access to a function variable can cause different results in a PLC program (POU) as the operating date can be modified due to task changes.

Shared memory

Programmable limit switch data in the shared memory is declared as a global array "PlsData[]" of the data structure [PlsData\[\] - PLS data in shared memory, page 98](#). The definition is part of the global variable list "GVL_Base" in the "ML_Base" library.

PLS data in shared memory

```
VAR_GLOBAL
...
PlsData AT %MBccc: ARRAY [MB_MIN_PLS_NUMBER..MB_MAX_PLS_NUMBER] OF ML_PLSDATA_SM;
...
END_VAR
```



For the addresses of "AxisData", "ControlData", "PlsData" and "ProbeData", refer to the library manager of the BASELIB= ML_Base / Global Variables List / GVL_Base.

The array limits are located in the system-specific library parts BASELIBCONST= ML_Base_Constant_xxx / Global Variable Lists / GVL_BaseConstant.

Function variable access

Function variables of the programmable limit switch data are declared as global array "DV_Pls[]" of the function block "ML_DirectVarPls". The definition is part of the global variable list "GVL_Base" in the "ML_Base" library, refer to [DV_Pls\[\] - PLS data via function variable access, page 100](#).

Function variables of PLS data

```
VAR_GLOBAL
```

```
...
DV_Pls: ARRAY [MB_MIN_PLS_NUMBER..MB_MAX_PLS_NUMBER] OF ML_DirectVarPls;
...
END_VAR
```



If a parameter can use the function variable access, it is described in the corresponding parameter description (parameter attributes, direct access, example N-0-0000).

PlsData[] - PLS data in shared memory

Brief description

When creating a new project in IndraWorks, programmable limit switch data is automatically provided. Using the specifically defined index MB_MIN_PLS_NUMBER .. MB_MAX_PLS_NUMBER, PLS data can be accessed via

- function variable access ("DV_Pls[]") and
- shared memory ("PlsData[]")



In the Motion mode "Operating mode", programmable limit switch data in shared memory is consistently and cyclically updated. Due to the time required for updating, the scope of this data is limited. As direct variables, programmable limit switch data is updated irrespective of the Motion mode at the access time.

PLS data in shared memory and function variable access can be distributed across several motion cycles (command value generation, actual value provision), depending on the task properties of the POU and are therefore not necessarily consistent with each other.

Programmable limit switch data in the shared memory is declared as the global array "PlsData[]" of the data structure „ML_PLSDATA_SM“. The definition is part of the global variable list "GVL_Base" in the "ML_Base" library.

Defining the global array "PlsData[]"

```
VAR_GLOBAL
```

```
...
PlsData AT %MBccc: ARRAY [MB_MIN_PLS_NUMBER..MB_MAX_PLS_NUMBER] OF ML_PLSDATA_SM;
...
END_VAR
```



For the addresses of "AxisData", "ControlData", "PlsData" and "ProbeData", refer to the library manager of the BASELIB= ML_Base / Global Variables List / GVL_Base.

The array limits are located in the system-specific library parts BASELIBCONST= ML_Base_Constant_xxx / Global Variable Lists / GVL_BaseConstant.

Also refer to

- [ML_PLSDATA_SM, page 112](#)

Functional description

The read-only access to data in the shared memory can be carried out via the following equivalent statements (ST).

Read-only access to data

```
MyVar:= PlsData[MB_MAX_PLS_NUMBER].Register_0_i;
MyVar:= PlsData[Index].Register_0_i;
MyVar:= PlsData[1].Register_0_i;
```

The write access to data in the shared memory can be carried out via the following equivalent statements (ST).

Write access to data

```
PlsData[MB_MAX_PLS_NUMBER].Force_0_q:= MyVar;
PlsData[Index].Force_0_q:= MyVar;
PlsData[1].Force_0_q:= MyVar;
```

The IntelliSense of a variable in the shared memory contains the parameter ident number, the access possibilities (RD or RW) and the parameter names.

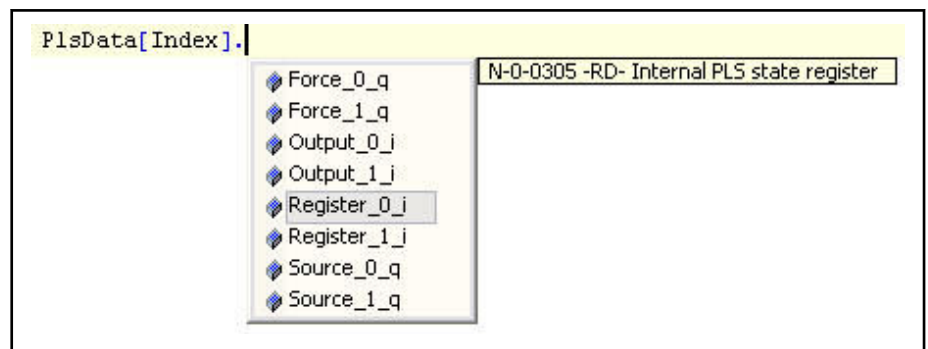


Fig. 4-8: IntelliSense of a variable in the shared memory



Variables of a programmable limit switch in the shared memory can also be accessed by using references via a user-defined name.

User-defined name by referencing

```
VAR
  MyPlsName: REFERENCE TO ML_PLSDATA_SM:= PlsData[MB_MAX_PLS_NUMBER];
END_VAR
```

```
MyVar:= MyPlsName.Register_0_i;
MyPlsName.Force_0_q:= MyVar;
```

See also:

- [ML_PLSDATA_SM](#), page 112

DV_Pls[] - PLS data via function variable access

Brief description When creating a new project in IndraWorks, programmable limit switch data is automatically provided. Using the specifically defined index MB_MIN_PLS_NUMBER .. MB_MAX_PLS_NUMBER ([GVL_BaseConstant](#), page 126), PLS data can be accessed via function variable access (DV_Pls[]) and shared memory (PlsData[]).



In the Motion mode "Operating mode", programmable limit switch data in shared memory is consistently and cyclically updated. Due to the time required for updating, the scope of this data is limited.

PLS data via function variable access (alias "direct variables") is updated irrespective of the Motion mode at the access time.

Programmable limit switch data can be distributed across several motion cycles (command value generation, actual value provision) depending on the task properties of the POU and are therefore not necessarily consistent with each other.

Programmable limit switch data via function variable access is declared as a global array "DV_Pls[]" of the function block "ML_DirectVarPls". The definition is part of the global variable list "GVL_Base" in the "ML_Base" library.

The function block "ML_DirectVarPls" provides function variable access as methods. The existence of the GET and SET properties for the respective method defines the access option. Executing the write-access SET on a function variable depends on Motion mode and does not generate an error message in a negative case.



The function block "ML_DirectVarPls" is not intended for direct use in the user program.

Defining the global array "DV_Pls[]"

```
VAR_GLOBAL
...
DV_Pls: ARRAY [MB_MIN_PLS_NUMBER..MB_MAX_PLS_NUMBER] OF ML_DirectVarPls;
...
END_VAR
```

Functional description The read-only access to a function variable can be executed via the following equivalent statements (ST) if the respective method is provided with a GET property (READ).

Read-only access to data

```
MyVar:= DV_Pls[Index].N_0_0303;
MyVar:= DV_Pls[1].N_0_0303;
```

The write access to a function variable can be executed via the following equivalent statements (ST) if the respective method is provided with a SET property (WRITE).

Write access to data

```
DV_Pls[Index].N_0_0317:= MyVar;
DV_Pls[1].N_0_0317:= MyVar;
```

The IntelliSense of a function variable contains the parameter ident number, the access options (READ or READ/WRITE), the parameter names as well as the method to determine the current access status.

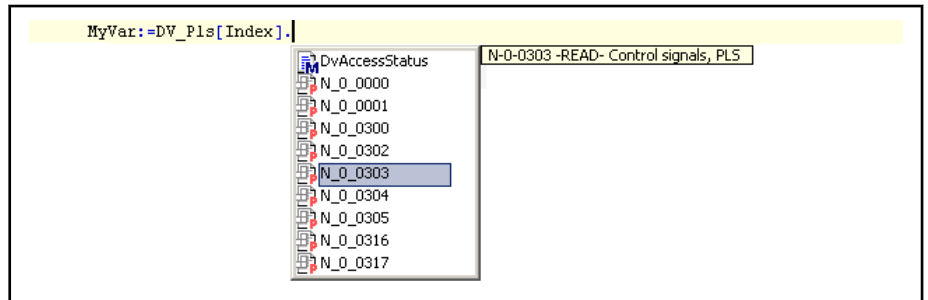


Fig. 4-9: Function variable access using the IntelliSense functionality



Function variables of a PLS can also be accessed by using references via a user-defined name.

User-defined name by referencing

```
VAR
  MyPlsName: REFERENCE TO ML_DirectVarPls:= DV_Pls[MB_MAX_PLS_NUMBER];
END_VAR

MyVar:= MyPlsName.N_0_0303;
MyPlsName.N_0_0317:= MyVar;
```

Status during function variable access

In addition to properties for reading and writing, the function blocks for function variable access also offer a way to determine the **current access status** of a parameter using the **DvAccessStatus** method.

The corresponding parameter constant is transferred to the method.

The method returns the return value of the last access (== current access status).

The current access status of a function variable is read according to the following statement (ST):

Read: current access status of a function variable

```
A_Status:= DV_Axis[MyAxisName.AxisNo].DvAccessStatus(FP_A_0_0000);
C_Status:= DV_Control[0].DvAccessStatus(FP_C_0_0023);
M_Status:= DV_Probe[MB_MIN_PROBE_NUMBER].DvAccessStatus(FP_M_0_0021);
N_Status:= DV_Pls[Index].DvAccessStatus(FP_N_0_0303);
O_Status:= DV_Oscil[Index].DvAccessStatus(FP_O_0_0021);
```

Declaration files for Global Constants

- [GVL_A_Param_ID, page 130](#)
- [GVL_C_Param_ID, page 130](#)
- [GVL_M_Param_ID, page 130](#)
- [GVL_N_Param_ID, page 131](#)
- [GVL_O_Param_ID, page 131](#)

Status (extract, for further information, see SERCOS_TABLE)

- 16# 0000.0000: No error
- 16# F02D1001: IDN parameter not available
- 16# F02Dxxxx: sercos error codes

- 16# F02D7013: command cannot be executed (FW: command not implemented)
- 16# F02D8001: access to parameter currently not possible (BUSY).

Touch probe data via shared memory and function variable access

Touch probe data via shared memory and function variable access, general

The touch probe data saved in the M-parameters can be accessed by the PLC either via a simplified, blocking instruction "Function variable access" (alias "direct variable") or via shared memory.

Touch probe data data in the shared memory is updated consistently and cyclically in the Motion mode **operating mode**. Due to the required update time, its scope is limited. Updating the touch probe data in shared memory can be completely deactivated (C-0-0036) to increase performance. The actual values of the touch probe data in the shared memory are always refreshed when the Motion kernel MK1 has completed its process.

Touch probe data via function variable access is updated irrespective of the Motion mode at the access time.

A part of the touch probe data is available in the shared memory as well as for access via direct variables. Since the touch probe data in the shared memory is cyclically exchanged irrespective of its use, further redundant access via direct variables should be avoided due to performance reasons.



The simplified, blocking access to a function variable is an implicit function call and requires more runtime resources than a simple memory access to a local PLC variable.

It is recommended to avoid accessing a function variable several times and to buffer the operating date to a local PLC variable if required.

Via shared memory and function variable access, touch probe data can be distributed across several Motion cycles (command value generation, actual value provision) depending on the task properties of the POU and are therefore not necessarily consistent with each other.



A function variable is not part of the process image of the PLC task.

The simplified, blocking access to a function variable is an implicit function call which is executed at the moment of calling.

Multiple access to a function variable can cause different results in a PLC program (POU) as the operating date can be modified due to task changes.

Shared memory

Touch probe data in the shared memory is declared as a global array "ProbeData[]" of the data structure [ProbeData\[\] - touch probe data in shared memory, page 103](#). The definition is part of the global variable list "GVL_Base" in the "ML_Base" library.

The touch probes are numbered and the name of the touch probe is not important for this access type.²⁾

²⁾ *MB_MAX_PROBE_NUMBER, maximum number of admissible touch probes, currently 100, control type-dependent.*

Touch probe data in the shared memory

```

VAR_GLOBAL
...
ProbeData AT %MBddd: ARRAY [MB_MIN_PROBE_NUMBER..MB_MAX_PROBE_NUMBER] OF ML_PROBEDATA_SM;
...
END_VAR

```



For the addresses of "AxisData", "ControlData", "PlsData" and "ProbeData", refer to the library manager of the BASELIB= ML_Base / Global Variables List / GVL_Base.

The array limits are located in the system-specific library parts BASELIBCONST= ML_Base_Constant_xxx / Global Variable Lists / GVL_BaseConstant.

Function variable access

Function variables of the touch probe data are declared as global array "DV_Probe[]" of the function block "ML_DirectVarProbe". The definition is part of the global variable list "GVL_Base" in the "ML_Base" library, refer to [DV_Probe\[\] - touch probe data via function variable access, page 105](#).

Function variables of touch probe data

```

VAR_GLOBAL
...
DV_Probe: ARRAY [MB_MIN_PROBE_NUMBER..MB_MAX_PROBE_NUMBER] OF ML_DirectVarProbe;
...
END_VAR

```



If a parameter can use the function variable access, it is described in the corresponding parameter description (parameter attributes, direct access, example M-0-0000).

ProbeData[] - touch probe data in shared memory**Brief description**

When creating a new project in IndraWorks, a defined number of touch probes is generated automatically.

By means of the touch probe number, the touch probe data can access in a subscripted way via

- function variable access ("DV_Probe[]") and
- shared memory ("ProbeData[]")



In the Motion mode "Operating mode", touch probe data in shared memory is consistently and cyclically updated. Due to the time required for updating, the scope of this data is limited. Updating the touch probe data in shared memory can be completely deactivated (C-0-0036) to increase performance. Touch probe data as direct variables is updated irrespective of the Motion mode at the access time.

Touch probe data in shared memory and function variables can be distributed across several Motion cycles (command value generation, actual value provision), depending on the task properties of the POU and are therefore not necessarily consistent with each other.

Touch probe data in the shared memory is declared as a global array "ProbeData[]" of the data structure "ML_PROBEDATA_SM". The definition is part of the global variable list "GVL_Base" in the "ML_Base" library.

Defining the global array "ProbeData[]"

```
VAR_GLOBAL
...
ProbeData AT %MBddd: ARRAY [MB_MIN_PROBE_NUMBER..MB_MAX_PROBE_NUMBER] OF ML_PROBEDATA_SM;
...
END_VAR
```



For the addresses of "AxisData", "ControlData", "PlsData" and "ProbeData", refer to the library manager of the BASELIB= ML_Base / Global Variables List / GVL_Base.

The array limits are located in the system-specific library parts BASELIBCONST= ML_Base_Constant_xxx / Global Variable Lists / GVL_BaseConstant.

See also:

- [ML_PROBEDATA_SM, page 113](#)

Functional description

The read-only access to data in the shared memory can be carried out via the following equivalent statements (ST).

Read-only access to data in the shared memory:

```
MyVar:= ProbeData[TouchProbe???.TouchProbeNo].FlexProfil_Active;
MyVar:= ProbeData[Index].FlexProfil_Active;
MyVar:= ProbeData[1].FlexProfil_Active;
```

Write access to touch probe data in shared memory is not possible.

The IntelliSense of a variable in the shared memory contains the parameter ident number and the parameter names.

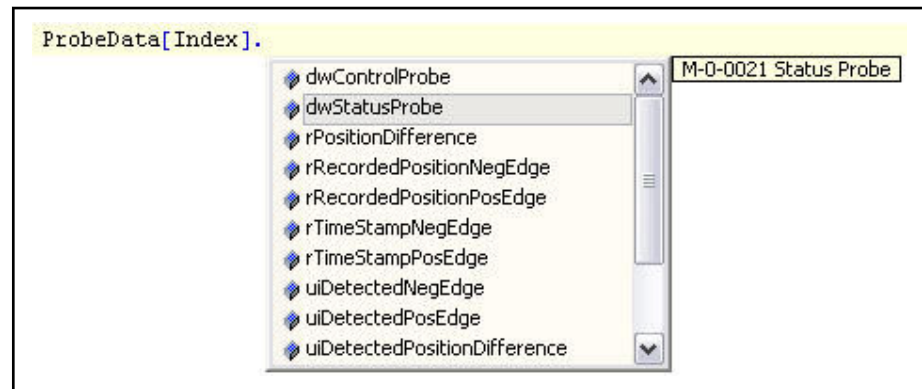


Fig. 4-10: IntelliSense of a variable in the shared memory



Variables of a touch probe in the shared memory can also be accessed by using references via an user-defined name.

User-defined name by referencing:

```
VAR
MyProbeName:REFERENCE TO ML_PROBEDATA_SM:= ProbeData[TouchProbe???.TouchProbeNo];
END_VAR

MyVar:= MyProbeName.dwStatusProbe;
```

Selected areas of the data structure ML_POBEDATA_SM

rRecordedPositionPosEdge	:SM_EXT_TYPES;	(* M-0-0130 Value Positive Edge Probe*)
rRecordedPositionNegEdge	:SM_EXT_TYPES;	(* M-0-0131 Value Negative Edge Probe*)
rPositionDifference	:SM_EXT_TYPES;	(* M-0-0132 Difference Value Probe*)

The data containers "rRecordedPositionPosEdge", "rRecordedPositionNegEdge" and "rPositionDifference" contain configurable real-time data. Depending on the configuration, the containers may contain values of various data types. However, the point operator can be used to access the respective data type.

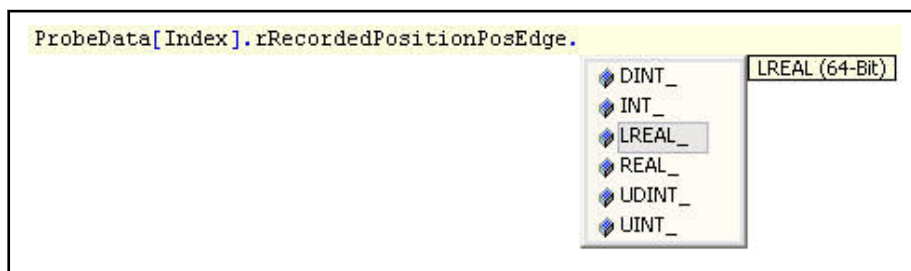


Fig. 4-11: Data type specification via point operator

Also refer to

- [SM_EXT_TYPES](#), page 115
- [ML_PROBEDATA_SM](#), page 113

DV_Probe[] - touch probe data via function variable access

Brief description

When creating a new project in IndraWorks, a defined number of touch probes is generated automatically. With the automatically generated touch probe number MB_MIN_PROBE_NUMBER..MB_MAX_PROBE_NUMBER ([GVL_BaseConstant](#), page 126), the touch probe data can be accessed in a subscripted way via function variable access (DV_Probe[]) and shared memory (ProbeData[]).



Touch probe data data in the shared memory is updated consistently and cyclically in the Motion mode "operating mode". Due to the required update time, its scope is limited. Updating the touch probe data in shared memory can be completely deactivated (C-0-0036) to increase performance.

Touch probe data via function variable access (alias "direct variables") is updated irrespective of the Motion mode at the access time.

Touch probe data data as function variables can be distributed across several motion cycles (command value generation, actual value provision) depending on the task properties of the POU and are therefore not necessarily consistent with each other.

Touch probe data via function variable access is declared as global array "DV_Probe[]" of the function block "ML_DirectVarProbe". The definition is part of the global variable list "GVL_Base" in the "ML_Base" library.

The function block "ML_DirectVarProbe" provides access to direct variables using methods. The GET and SET properties of the respective method define the access option. Executing the write-access SET on a direct variable depends on Motion mode and does not generate an error message in a negative case.



The function block "ML_DirectVarProbe" is not intended for direct use in the user program.

Defining the global array "DV_Probe[]"

```
VAR_GLOBAL
...
DV_Probe:ARRAY[MB_MIN_PROBE_NUMBER..MB_MAX_PROBE_NUMBER] OF ML_DirectVarProbe;
...
END_VAR
```

Functional description

The read-only access to a function variable can be executed via the following equivalent statements (ST) if the respective method is provided with a GET property (READ).

Read-only access to data

```
MyVar:= DV_Probe[TouchProbe???. TouchProbeNo].M_0_0130;
MyVar:= DV_Probe[Index].M_0_0130;
MyVar:= DV_Probe[1].M_0_0130;
```

The write access to a function variable can be executed via the following equivalent statements (ST) if the respective method is provided with a SET property (WRITE).

Write access to data

```
DV_Probe[TouchProbe???. TouchProbeNo].M_0_0205:= MyVar;
DV_Probe[Index].A_0_0205:= MyVar;
DV_Probe[1].A_0_0205:= MyVar;
```

The IntelliSense of a function variable contains the parameter ident number, the access options (READ or READ/WRITE), the parameter names as well as the method to determine the current access status.

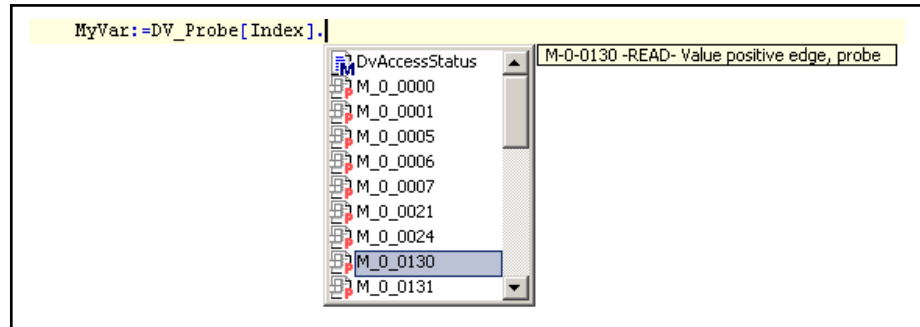


Fig. 4-12: Function variable access using the IntelliSense functionality



Function variables of a touch probe can also be accessed by using references via a user-defined name.

User-defined name by referencing

```
VAR
MyProbeName: REFERENCE TO ML_DirectVarProbe:= DV_Probe[TouchProbe???. TouchProbeNo];
END_VAR

MyVar:= MyProbeName.M_0_0130;
MyProbeName.M_0_0205:= MyVar;
```

Status during function variable access

In addition to properties for reading and writing, the function blocks for function variable access also offer a way to determine the **current access status** of a parameter using the **DvAccessStatus** method.

The corresponding parameter constant is transferred to the method.

The method returns the return value of the last access (== current access status).

The current access status of a function variable is read according to the following statement (ST):

Read: current access status of a function variable

```
A_Status:= DV_Axis[MyAxisName.AxisNo].DvAccessStatus(FP_A_0_0000);
C_Status:= DV_Control[0].DvAccessStatus(FP_C_0_0023);
M_Status:= DV_Probe[MB_MIN_PROBE_NUMBER].DvAccessStatus(FP_M_0_0021);
N_Status:= DV_Pls[Index].DvAccessStatus(FP_N_0_0303);
O_Status:= DV_Osci[Index].DvAccessStatus(FP_O_0_0021);
```

Declaration files for Global Constants

- [GVL_A_Param_ID, page 130](#)
- [GVL_C_Param_ID, page 130](#)
- [GVL_M_Param_ID, page 130](#)
- [GVL_N_Param_ID, page 131](#)
- [GVL_O_Param_ID, page 131](#)

Status (extract, for further information, see SERCOS_TABLE)

- 16# 0000.0000: No error
- 16# F02D1001: IDN parameter not available
- 16# F02Dxxxx: sercos error codes
- 16# F02D7013: command cannot be executed (FW: command not implemented)
- 16# F02D8001: access to parameter currently not possible (BUSY).

Oscilloscope data via function variable access**Oscilloscope data via function variable access, general**

Oscilloscope data saved in O-parameters can be accessed by the PLC via function variable access. Oscilloscope data is not available in the shared memory.

Oscilloscope data via function variable access (alias "direct variables") is updated irrespective of the Motion mode at the access time.



The simplified, blocking access to a function variable is an implicit function call and requires more runtime resources than a simple memory access to a local PLC variable.

It is recommended to avoid accessing a function variable several times and to buffer the operating data to a local PLC variable if required.

Via function variable access, oscilloscope data can be distributed across several motion cycles (command value generation, actual value provision) depending on the task properties of the POU and are therefore not necessarily consistent with each other.

Shared memory Oscilloscope data is not available in the shared memory.

Function variable access Function variables of the oscilloscope data are declared as global array "DV_Osc[]" of the function block "ML_DirectVarOsci". The definition is part of the global variable list "GVL_Base" in the "ML_Base" library, refer to [DV_Osc\[\] - oscilloscope data via function variable access, page 108](#).

Function variables of oscilloscope data

```
VAR_GLOBAL
```

```
    ...
    DV_Osc: ARRAY [MB_MIN_OSCI_NUMBER..MB_MAX_OSCI_NUMBER] OF ML_DirectVarOsci;
    ...
```

```
END_VAR
```



The array limits are located in the system-specific library parts BASELIBCONST= ML_Base_Constant_xxx / Global Variable Lists / GVL_BaseConstant.



If a parameter can use the function variable access, it is described in the corresponding parameter description (parameter attributes, direct access, example O-0-0000).

DV_Osc[] - oscilloscope data via function variable access

Brief description When creating a new project in IndraWorks, oscilloscope data is automatically provided. With the specifically defined index MB_MIN_OSCI_NUMBER .. MB_MAX_OSCI_NUMBER ([GVL_BaseConstant, page 126](#)), oscilloscope data can be accessed via direct variables (DV_Osc[]).



Oscilloscope data via function variable access (alias "direct variables") is updated irrespective of the Motion mode at the access time. Oscilloscope data can be distributed across several Motion cycles (command value generation, actual value provision) depending on the task properties of the POU and are therefore not necessarily consistent with each other.

Oscilloscope data via function variable access is declared as global array "DV_Osc[]" of the function block "ML_DirectVarOsci". The definition is part of the global variable list "GVL_Base" in the "ML_Base" library.

The function block "ML_DirectVarOsci" provides function variable access as methods. The existence of the GET and SET properties for the respective method defines the access option. Executing the write-access SET on a function variable depends on Motion mode and does not generate an error message in a negative case.



The function block "ML_DirectVarOsci" is not intended for direct use in the user program.

Defining the global array "DV_Osc[]"

```
VAR_GLOBAL
```

```
    ...
    DV_Osc: ARRAY [MB_MIN_OSCI_NUMBER..MB_MAX_OSCI_NUMBER] OF ML_DirectVarOsci;
    ...
```

```
END_VAR
```

Functional description The read-only access to a function variable can be executed via the following equivalent statements (ST) if the respective method is provided with a GET property (READ).

Read-only access to data

```
MyVar:= DV_Osc[Index].O_0_0021;
MyVar:= DV_Osc[1].O_0_0021;
```

The write access to a function variable can be executed via the following equivalent statements (ST) if the respective method is provided with a SET property (WRITE).

Write access to data

```
DV_Osc[Index].O_0_0000:= MyVar;
DV_Osc[1].O_0_0000:= MyVar;
```

The IntelliSense of a function variable contains the parameter ident number, the access options (READ or READ/WRITE), the parameter names as well as the method to determine the current access status.

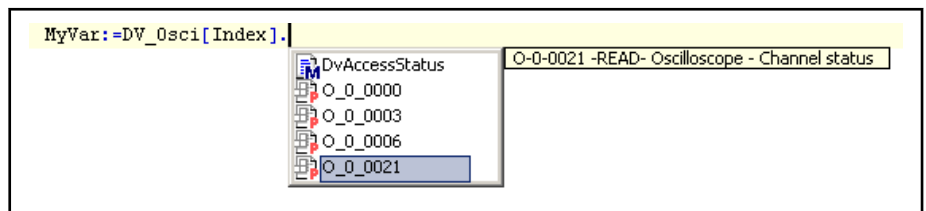


Fig. 4-13: Function variable access using the IntelliSense functionality



Function variables of an oscilloscope can also be accessed by using references via a user-defined name.

User-defined Naming by referencing

```
VAR
  MyOsciName: REFERENCE TO ML_DirectVarOsci:= DV_Osc[MB_MAX_OSCI_NUMBER];
END_VAR

MyVar:= MyOsciName.O_0_0021;
MyOsciName.O_0_0000:= MyVar;
```

Status during function variable access

In addition to properties for reading and writing, the function blocks for function variable access also offer a way to determine the **current access status** of a parameter using the **DvAccessStatus** method.

The corresponding parameter constant is transferred to the method.

The method returns the return value of the last access (== current access status).

The current access status of a function variable is read according to the following statement (ST):

Read: current access status of a function variable

```
A_Status:= DV_Axis[MyAxisName.AxisNo].DvAccessStatus(FP_A_0_0000);
C_Status:= DV_Control[0].DvAccessStatus(FP_C_0_0023);
M_Status:= DV_Probe[MB_MIN_PROBE_NUMBER].DvAccessStatus(FP_M_0_0021);
N_Status:= DV_Pls[Index].DvAccessStatus(FP_N_0_0303);
O_Status:= DV_Osc[Index].DvAccessStatus(FP_O_0_0021);
```


Declaration files for Global Constants

- GVL_A_Param_ID, page 130
- GVL_C_Param_ID, page 130
- GVL_M_Param_ID, page 130
- GVL_N_Param_ID, page 131
- GVL_O_Param_ID, page 131

Status (extract, for further information, see SERCOS_TABLE)

- 16# 0000.0000: No error
- 16# F02D1001: IDN parameter not available
- 16# F02Dxxxx: sercos error codes
- 16# F02D7013: command cannot be executed (FW: command not implemented)
- 16# F02D8001: access to parameter currently not possible (BUSY).

ML_AXISDATA_SM**Brief description**

An instance of the structure ML_AXISDATA_SM contains information on the corresponding axis which can be accessed via **shared memory**.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/shared memory types

Tab. 4-14: *Library assignment*

Structure of the data structure ML_AXISDATA_SM:

```
TYPE ML_AXISDATA_SM :
STRUCT
```

```

wUserCmdDataBOOLA_q :WORD; // A-0-0542 -RW- User defined commanded data BOOL number A
wUserCmdDataBOOLB_q :WORD; // A-0-0545 -RW- User defined commanded data BOOL number B
wUserCmdDataBOOLC_q :WORD; // A-0-0548 -RW- User defined commanded data BOOL number C
wUserCmdDataBOOLD_q :WORD; // A-0-0551 -RW- User defined commanded data BOOL number D

dwUserCmdDataA_q :SM_TYPES; // A-0-0501 -RW- Container user defined commanded data A
dwUserCmdDataB_q :SM_TYPES; // A-0-0503 -RW- Container user defined commanded data B
dwUserCmdDataC_q :SM_TYPES; // A-0-0505 -RW- Container user defined commanded data C
dwUserCmdDataD_q :SM_TYPES; // A-0-0507 -RW- Container user defined commanded data D
dwUserCmdDataE_q :SM_TYPES; // A-0-0509 -RW- Container user defined commanded data E

dwMotionProfileStatusBOOLs_i :DWORD; // A-0-2931 -RD- Motion profile status word
dwFlexProfileEventControlBOOLs_q :DWORD; // A-0-3011 -RW- Condition control variable, FP
FlexProfil_Sync :BIT; // A-0-3010.08 -RD- FlexProfile is in synchronisation
FlexProfil_Active :BIT; // A-0-3010.09 -RD- FlexProfile is active
FlexProfil_WaitForSwitching :BIT; // A-0-3010.10 -RD- FP is waiting for switching position
FlexProfil_ExeSingleFinished :BIT; // A-0-3010.11 -RD- FP EXECUTE SINGLE is finished
FlexProfil_Set0_Checked :BIT; // A-0-3010.16 -RD- FlexProfile Set0 is checked
FlexProfil_Set0_Ok :BIT; // A-0-3010.17 -RD- FlexProfile Set0 is ok
FlexProfil_Set1_Checked :BIT; // A-0-3010.18 -RD- FlexProfile Set0 is checked
FlexProfil_Set1_Ok :BIT; // A-0-3010.19 -RD- FlexProfile Set0 is ok
FlexProfil_Set2_Checked :BIT; // A-0-3010.20 -RD- FlexProfile Set0 is checked
FlexProfil_Set2_Ok :BIT; // A-0-3010.21 -RD- FlexProfile Set0 is ok
FlexProfil_Set3_Checked :BIT; // A-0-3010.22 -RD- FlexProfile Set0 is checked
FlexProfil_Set3_Ok :BIT; // A-0-3010.23 -RD- FlexProfile Set0 is ok
FlexProfil_EvtSet0_Checked :BIT; // A-0-3010.24 -RD- FlexProfile Event Set0 is checked
FlexProfil_EvtSet0_Ok :BIT; // A-0-3010.25 -RD- FlexProfile Event Set0 is ok
FlexProfil_EvtSet1_Checked :BIT; // A-0-3010.26 -RD- FlexProfile Event Set0 is checked
FlexProfil_EvtSet1_Ok :BIT; // A-0-3010.27 -RD- FlexProfile Event Set0 is ok
FlexProfil_EvtSet2_Checked :BIT; // A-0-3010.28 -RD- FlexProfile Event Set0 is checked
FlexProfil_EvtSet2_Ok :BIT; // A-0-3010.29 -RD- FlexProfile Event Set0 is ok
FlexProfil_EvtSet3_Checked :BIT; // A-0-3010.30 -RD- FlexProfile Event Set0 is checked
FlexProfil_EvtSet3_Ok :BIT; // A-0-3010.31 -RD- FlexProfile Event Set0 is ok
dwFlexProfileEventStatusBits_i :DWORD; // A-0-3012 -RD- Condition status variable, FP

Axis_CamTab_0 :BIT; // A-0-0021.00 -RD- Cam shaft table BIT 0
Axis_CamTab_1 :BIT; // A-0-0021.01 -RD- Cam shaft table BIT 1
```



```

Axis_CamTab_2      :BIT; // A-0-0021.02 -RD- Cam shaft table BIT 2 (reserved)
Axis_CamSwitching  :BIT; // A-0-0021.03 -RD- Status cam shaft distance switching
Axis_InVelocity    :BIT; // A-0-0021.04 -RD- Axis cmd velocity == actual cmd velocity (IPO control)
                    // Axis cmd velocity == actual velocity (IPO drive)
                    // depending on parameter "Synchronization run window, velocity" (A-0-0623)
Axis_Standstill    :BIT; // A-0-0021.05 -RD- Axis velocity <= A-0-0222 (standstill)
Axis_InPosition    :BIT; // A-0-0021.06 -RD- Axis cmd position == actual cmd position (IPO control)
                    // Axis cmd position == actual position (IPO drive)
                    // depending on parameter "Synchronization run window, position" (A-0-0622)
Axis_InSynchron    :BIT; // A-0-0021.07 -RD- Axis is in synchronisation
Axis_Warning       :BIT; // A-0-0021.08 -RD- Axis reports warning
Axis_Error         :BIT; // A-0-0021.09 -RD- Axis reports error
Axis_Homed         :BIT; // A-0-0021.10 -RD- Axis is relative / in homing
Axis_InTorque      :BIT; // A-0-0021.11 -RD- Axis cmd torque / force == actual torque
Axis_OperationMode :BIT; // A-0-0021.12 -RD- Axis is in operation mode (P4)
Axis_Inbb          :BIT; // A-0-0021.13 -RD- Axis is ready for power to the amplif. (bb)
Axis_InAb          :BIT; // A-0-0021.14 -RD- Axis is ready for power to the engine (Ab)
Axis_Power         :BIT; // A-0-0021.15 -RD- Axis has power on (AH_AF)
Axis_CmdActive     :BIT; // A-0-0021.16 -RD- Command active
Axis_ErrInProgress :BIT; // A-0-0021.17 -RD- Error reaction in progress
Axis_CyclicPosChannelOpen :BIT; // A-0-0021.18 -RD- Cyclic Position Channel is open for Axis
Axis_InMasterPhaseOffset :BIT; // A-0-0021.19 -RD- Master Phase offset achieved
Axis_CyclicTrqChannelOpen :BIT; // A-0-0021.20 -RD- Cyclic Position Channel is open for Axis
Axis_CyclicVelChannelOpen :BIT; // A-0-0021.21 -RD- Cyclic Velocity Channel is open for Axis
Axis_CyclicAnaChannelOpen :BIT; // A-0-0021.22 -RD- Cyclic Analoge Channel is open for Axis
Axis_NRT           :BIT; // A-0-0021.23 -RD- Connected Axis is connected with NRT channel
                    // Parameter access takes place over NRT channel
Axis_Interrupted   :BIT; // A-0-0021.24 -RD- Axis is not following commanded values
Axis_PLCParaSysActive :BIT; // A-0-0021.25 -RD- PLC parameter system active
Axis_PLCControlled :BIT; // A-0-0021.27 -RD- Interpolation influenced by PLC
Axis_PosDataModulo :BIT; // A-0-0021.28 -RD- Position data in modulo format
Axis_PosDataRotatoryScaled :BIT; // A-0-0021.29 -RD- Position data is rotatorily scaled
Axis_PosDataMasterAxisValid :BIT; // A-0-0021.30 -RD- Position data of the master axis is valid
Axis_CmdValueDecouplingActive :BIT; // A-0-0021.31 -RD- Command value decoupling active

ErrorStop          :BIT; // A-0-0022.00 -RD- Axis is in "ErrorStop"
Stopping           :BIT; // A-0-0022.01 -RD- Axis is in "Stopping"
Homing             :BIT; // A-0-0022.02 -RD- Command homing is active
PowerOn           :BIT; // A-0-0022.03 -RD- Axis got power
DiscreteMotion     :BIT; // A-0-0022.04 -RD- Axis is in "Discrete Motion"
ContinuousMotion   :BIT; // A-0-0022.05 -RD- Axis is in "Continuous Motion"
SynchronizedMotion :BIT; // A-0-0022.06 -RD- Axis is in "Synchronized Motion"
Standstill         :BIT; // A-0-0022.07 -RD- Axis is stopped
CoordinatedMotion  :BIT; // A-0-0022.08 -RD- Axis is in "CoordinatedMotion"
ControlledMotion   :BIT; // A-0-0022.09 -RD- Axis is in "ControlledMotion"

dwAxisCondition_i  :DWORD; // A-0-0024 -RD- Axis condition
rActualPosition_i  :REAL; // A-0-0100 -RD- Actual position
rActualVelocity_i  :REAL; // A-0-0102 -RD- Actual velocity
rActualTorqueForce_i :REAL; // A-0-0110 -RD- Actual torque/force

wUserActualDataBitA_i :WORD; // A-0-0562 -RD- User defined actual data bit number A
wUserActualDataBitB_i :WORD; // A-0-0565 -RD- User defined actual data bit number B
wUserActualDataBitC_i :WORD; // A-0-0568 -RD- User defined actual data bit number C
wUserActualDataBitD_i :WORD; // A-0-0571 -RD- User defined actual data bit number D

dwUserActualDataA_i :SM_TYPES; // A-0-0521 -RD- Container user defined actual data A
dwUserActualDataB_i :SM_TYPES; // A-0-0523 -RD- Container user defined actual data B
dwUserActualDataC_i :SM_TYPES; // A-0-0525 -RD- Container user defined actual data C
dwUserActualDataD_i :SM_TYPES; // A-0-0527 -RD- Container user defined actual data D
dwUserActualDataE_i :SM_TYPES; // A-0-0529 -RD- Container user defined actual data E
dwUserActualDataF_i :SM_TYPES; // A-0-0531 -RD- Container user defined actual data F

```

```

END_STRUCT
END_TYPE

```

Also refer to:

- [AxisData\[\]](#) - axis data in shared memory, page 85,
- [SM_TYPES](#), page 114..

ML_CONTROLDATA_SM

Brief description

An instance of the structure ML_CONTROLDATA_SM contains the information of the corresponding control which can be accessed via **Shared Memory**.

ML_Base.library

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/shared memory types

Tab. 4-15: Library assignment

Data structure design ML_CONTROLDATA_SM:

```

TYPE ML_CONTROLDATA_SM :
STRUCT
  LINK_SYNC1          :BIT; // C-0-0704.00 -RW- Link synchronization bit 0
  LINK_SYNC2          :BIT; // C-0-0704.01 -RW- Link synchronization bit 1
  LINK_SYNC3          :BIT; // C-0-0704.02 -RW- Link synchronization bit 2
  LINK_SYNC4          :BIT; // C-0-0704.03 -RW- Link synchronization bit 3
  LINK_REBUILD_RING   :BIT; // C-0-0704.04 -RW- Rebuild link ring
  LINK_DISABLE_HOTPLUG :BIT; // C-0-0704.05 -RW- Disable hotplug within C2C master

  arLinkSyncBits_i    :ARRAY[0..64] OF ML_SYNC; // C-0-0713 -RD- Link synchr. bits of ctrls
  arLinkValidBits_i    :ARRAY[0..64] OF ML_LINK; // C-0-0712 -RD- Link valid. status bits of ctrls

  wCurrentMotionMode   :WORD; // C-0-0452 - RD - Current motion mode

  LINK_ERROR           :BIT; // C-0-0714.00 -RD- Link error
  LINK_ERROR_PRI       :BIT; // C-0-0714.01 -RD- Link error primary optical ring
  LINK_ERROR_SEC       :BIT; // C-0-0714.02 -RD- Link error secondary optical ring
  LINK_ERROR_RED       :BIT; // C-0-0714.03 -RD- Redundancy loss
  LINK_ERROR_HOTPLUG   :BIT; // C-0-0714.04 -RD- Hotplug error pending (see index 0..2)
  LINK_ERROR_HOTPLUG_BIT0 :BIT; // C-0-0714.05 -RD- Hotplug error; index 0
  LINK_ERROR_HOTPLUG_BIT1 :BIT; // C-0-0714.06 -RD- Hotplug error; index 1
  LINK_ERROR_HOTPLUG_BIT2 :BIT; // C-0-0714.07 -RD- Hotplug error; index 2
  LINK_PHASESWITCH_ACTIVE :BIT; // C-0-0714.08 -RD- Phase switch of link ring active
  LINK_HOTPLUG_ACTIVE  :BIT; // C-0-0714.09 -RD- Hotplug of new device active
  LINK_HOTPLUG_ABORTED :BIT; // C-0-0714.10 -RD- Hotplug of new device aborted
  LINK_HOTPLUG_BIT0    :BIT; // C-0-0714.11 -RD- Hotplug status; status 0
  LINK_HOTPLUG_BIT1    :BIT; // C-0-0714.12 -RD- Hotplug status; status 1
  LINK_HOTPLUG_BIT2    :BIT; // C-0-0714.13 -RD- Hotplug status; status 2
  LINK_MASTER_ACTIVE   :BIT; // C-0-0714.14 -RD- Link master active, cyclic commun. active
  LINK_PLL_LOCKED      :BIT; // C-0-0714.15 -RD- Link device synchron. to link master
  LINK_ACYC_PARA_UPDATE :BIT; // C-0-0714.16 -RD- Exchange of non-cyclic transmitted parameters is active

  arSercos              :ARRAY[0..0] OF ML_SERCOS; // C-0-0514 -RD- Diag of sercos interface

END_STRUCT
END_TYPE

```

ML_CONTROLDATA_SM used

- [ML_LINK](#) on page 113
- [ML_SERCOS](#) on page 114
- [ML_SYNC](#) on page 114

ML_PLSDATA_SM

Brief description

An instance of the structure ML_PLSDATA_SM contains information on the corresponding axis which can be accessed via **shared memory**.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/shared memory types

Tab. 4-16: Library assignment

Data structure design ML_PLSDATA_SM:

```

TYPE ML_PLSDATA_SM :
STRUCT
  Source_0_q          :BYTE; (* N-0-0302 -RW- Source control reg. (0/1 = force reg./PLS reg.) *)
  Source_1_q          :BYTE; (* N-0-0302 -RW- Source control reg. (0/1 = force reg./PLS reg.) *)
  Force_0_q           :BYTE; (* N-0-0303 -RW- Force reg. for output 'X201' *)
  Force_1_q           :BYTE; (* N-0-0303 -RW- Force reg. for output 'X202' *)

  Output_0_i          :BYTE; (* N-0-0304 -RD- Physical output 'X201' *)

```

```

Output_1_i      :BYTE; (* N-0-0304 -RD- Physical output 'X202' *)
Register_0_i    :BYTE; (* N-0-0305 -RD- Internal PLS state register *)
Register_1_i    :BYTE; (* N-0-0305 -RD- Internal PLS state register *)

MasterPosOffset: ARRAY[1..8] OF REAL; (* PLS master position offset *)
END_STRUCT
END_TYPE

```

ML_PROBEDATA_SM

Brief description

An instance of the structure ML_PROBEDATA_SM contains information on the corresponding touch probes can be accessed via **shared memory**.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/shared memory types

Tab. 4-17: Library assignment

Data structure design ML_PROBEDATA_SM:

```

TYPE ML_PROBEDATA_SM :
STRUCT
dwControlProbe      :DWORD;      (* M-0-0007 Control Probe*)
dwStatusProbe       :DWORD;      (* M-0-0021 Status Probe*)
rRecordedPositionPosEdge :SM_EXT_TYPES; (* M-0-0130 Value pos. Edge Probe*)
rRecordedPositionNegEdge :SM_EXT_TYPES; (* M-0-0131 Value neg. Edge Probe*)
rPositionDifference  :SM_EXT_TYPES; (* M-0-0132 Difference Value Probe*)
rTimeStampPosEdge    :SYS_TIME64; (* M-0-0140 Timestamp pos. Edge Probe*)
rTimeStampNegEdge    :SYS_TIME64; (* M-0-0141 Timestamp neg. Edge Probe*)
uiMissingMarksPosEdge :UINT;      (* M-0-0224 Number of pos. Marker Failures Probe*)
uiMissingMarksNegEdge :UINT;      (* M-0-0225 Number of neg. Marker Failures Probe*)
uiStatusInputProbe   :UINT;      (* M-0-0401 Status Input Probe *)
wEnablingProbe       :WORD;      (* M-0-0405 Enabling Probe*)
uiDetectedPosEdge    :UINT;      (* M-0-0409 Counter Values pos. Edge Probe*)
uiDetectedNegEdge    :UINT;      (* M-0-0410 Counter Values neg. Edge Probe*)
uiDetectedPositionDifference :UINT; (* M-0-0411 Counter Difference Values Probe*)
END_STRUCT
END_TYPE

```

See also:

- [SM_EXT_TYPES](#), page 115

ML_LINK

Brief description

The data type contains the validity bit of the control. This bit indicates whether or not the control data in the MLC link is valid.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/shared memory types, types

Tab. 4-18: Library assignment

Type definition

```

TYPE ML_LINK :
STRUCT
iLinkBit0 :BIT; // C-0-0712.00 -RD- Link validation status bit 0 of control
END_STRUCT
END_TYPE

```

ML_SERCOS

Brief description The data type contains the diagnostic bits of the Sercos interface of the control.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/shared memory types, types

Tab. 4-19: Library assignment

Type definition

```

TYPE ML_SERCOS :
STRUCT
  ERROR_PORT1 :BIT; // C-0-0514.01 -RD- Link error Port1
  ERROR_PORT2 :BIT; // C-0-0514.02 -RD- Link error Port2
  ERROR_RED   :BIT; // C-0-0514.03 -RD- Bus error
  REBUILD_RING :BIT; // C-0-0514.04 -RD- Rebuild Ring Flag

  PHASE_SWITCH_ACTIVE :BIT; // C-0-0514.05 -RD- PHASE_SWITCH_ACTIVE

  ACTUAL_PHASE_BIT0 :BIT; // C-0-0514.27 -RD- ACTUAL_PHASE_BIT0
  ACTUAL_PHASE_BIT1 :BIT; // C-0-0514.28 -RD- ACTUAL_PHASE_BIT1
  ACTUAL_PHASE_BIT2 :BIT; // C-0-0514.29 -RD- ACTUAL_PHASE_BIT2
  ACTUAL_PHASE_BIT3 :BIT; // C-0-0514.30 -RD- ACTUAL_PHASE_BIT3
  ACTUAL_PHASE_BIT4 :BIT; // C-0-0514.31 -RD- ACTUAL_PHASE_BIT4
END_STRUCT
END_TYPE

```

ML_SYNC

Brief description The data type contains four input bits of the control which are exchanged in the MLC link.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/shared memory types, types

Tab. 4-20: Library assignment

Type definition

```

TYPE ML_SYNC :
STRUCT
  iSyncBit0 :BIT; // C-0-0713.00 -RD- Link synchronization bit 0 of control
  iSyncBit1 :BIT; // C-0-0713.01 -RD- Link synchronization bit 1 of control
  iSyncBit2 :BIT; // C-0-0713.02 -RD- Link synchronization bit 2 of control
  iSyncBit3 :BIT; // C-0-0713.03 -RD- Link synchronization bit 3 of control
END_STRUCT
END_TYPE

```

SM_TYPES

Brief description The structure is used for example to provide information of a 32 bit data container (AT or MDT) to the user in a type-accurate way.

The desired structural element has to be added separated by ".".

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/shared memory types, types

Tab. 4-21: Library assignment

Data type declaration:

```

TYPE SM_TYPES:
STRUCT
  REAL_      :REAL;    // REAL (32-Bit)
  DINT_      :DINT;    // DINT (32-Bit)
  UDINT_     :UDINT;   // UDINT (32-Bit)
  UINT_      :UINT;    // UINT (16-Bit)
  INT_       :INT;     // INT (16-Bit)
END_STRUCT
END_TYPE

```



The 16 bit values result when the upper 16 bits of the container value are cut off.

Example:

AxisData[1].dwUserActualDataB_i.REAL_ refers to axis 1, AT data container B is interpreted as REAL.

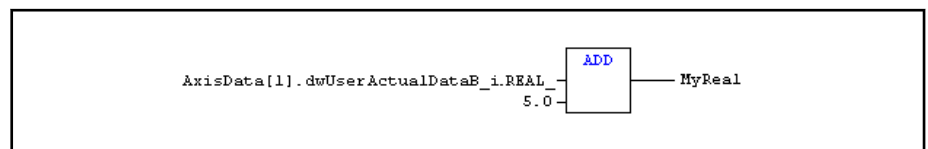


Fig. 4-14: Use of the cyclic Sercos data channel for reading

SM_EXT_TYPES**Brief description**

The structure is used for example in order to provide the information of one parameter to the user in the right type.

The desired structural element has to be added separated by ".".

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/shared memory types, types

Tab. 4-22: Library assignment

Type definition SM_EXT_TYPES

```

TYPE SM_EXT_TYPES:
STRUCT
  LREAL_     :LREAL;   (* LREAL (64-Bit) *)
  REAL_      :REAL;    (* REAL (32-Bit) *)
  DINT_      :DINT;    (* DINT (32-Bit) *)
  UDINT_     :UDINT;   (* UDINT (32-Bit) *)
  UINT_      :UINT;    (* UINT (16-Bit) *)
  INT_       :INT;     (* INT (16-Bit) *)
END_STRUCT
END_TYPE

```

Example:

M-0-0130, the measured value of the positive edge, touch probe, is a date of the length of 4 byte/float.

ProbeData[3].rValuePositiveEdgeProbe.REAL_

means that the measured value of the positive edge of touch probe 3 is provided as a REAL value.

However, to further process this value as an LREAL value, the user has to select the following interpretation:

```
ProbeData[3].rValuePositiveEdgeProbe.LREAL_
```

4.3

Error/diagnostic memory - supporting user diagnostics and logbook access

4.3.1

ML_ReadDiagLog

Brief description

The function block ML_ReadDiagLog retrieves entries from the diagnostic logbook.

Assignment: Target system/library	Target system	Library/folder
	IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/error/diagnostic memory

Tab. 4-23: Library assignment

Interface description

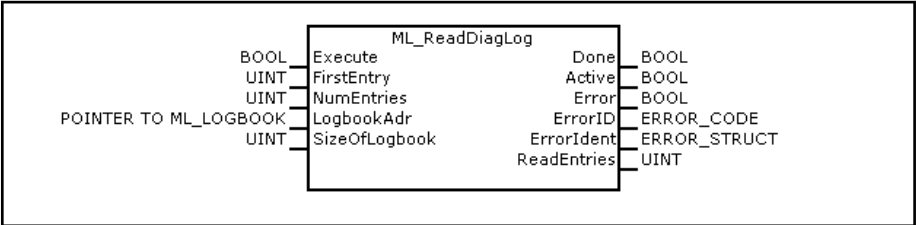


Fig. 4-15: ML_ReadDiagLog function block

	Name	Type	Comment
VAR_INPUT	Execute	UDINT	Processing enabled at a rising edge
	FirstEntry	UINT	Start index from which the diagnostic logbook is to be stored
	NumEntries	UINT	Maximum number of entries to be read
	LogbookAdr	POINTER TO ML_LOGBOOK	Address on which the entries of the diagnostic logbook is to be saved See ML_LOGBOOK, page 121
	SizeOfLogbook	UINT	Size of the data range created in "LogbookAdr" in bytes
VAR_OUTPUT	Done	BOOL	Operation executed
	Active	BOOL	Confirmation of processing enabled
	Error	BOOL	Indicates that an error occurred in the function block
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information Refer to Troubleshooting, page 117
	ReadEntries	UINT	Number of logbook entries read

Tab. 4-24: ML_ReadDiagLog interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
FirstEntry	UINT	0	999	0	Rising edge at "Execute"

Name	Type	Min. value	Max. value	Default value	Effective
NumEntries	UINT	1	1000	10	Rising edge at "Execute"
LogbookAdr	POINTER TO ML_LOGBOOK	>0		0	Rising edge at "Execute"
SizeOfLogbook	UINT	SIZEOF (ML_LOGBOOK)		0	Rising edge at "Execute"

Tab. 4-25: Input behavior of the ML_ReadDiagLog

Functional description

The function block ML_ReadDiagLog reads entries from the control log starting from the entry "NumEntries" specified in "FirstEntry" and transfers those entries to the array specified with "LogbookAdr".

The inputs "FirstEntry" and "NumEntries" determine the part of the log to be read. The latest entry has the number zero. The log can save up to 1000 entries.

An address of an array of the ML_LOGBOOK data type has to be assigned to the "LogbookAdr" input. The address can be determined using the ADR() operator. The array size in byte is transferred at the "SizeOfCoilData" input. This size can be determined using the SIZEOF() operator.

At a rising edge at "Execute", the inputs "FirstEntry", "NumEntries", "LogbookAdr" and "SizeOfLogbook" are applied and processing is started. During processing, the "Active" output is set. The "Done" output is set after successful processing. The data in the array applied at the "LogbookAdr" input are valid. "ReadEntries" contains the number of the logbook entries transferred to the array.

If an error occurs while processing, the "Error" output is set. For further information on the error, refer to "ErrorID" and "ErrorIdent".

Troubleshooting

The ML_ReadDiagLog function block returns the following error codes of the F_RELATED_TABLE, ERROR_TABLE, table number 16#0170:

ErrorID	Additional1	Additional2	Description
INPUT_INVALID_ERROR	16#00002800	16#00000000	"LogbookAdr" input is 0
INPUT_INVALID_ERROR	16#00002801	16#00000000	"SizeOfLogbook" input is 0
INPUT_INVALID_ERROR	16#00002801	16#00000001	"SizeOfLogbook" is too small for the requested number of logbook entries
INPUT_INVALID_ERROR	16#00002801	16#00000002	"SizeOfLogbook" is no multiple of the ML_LOGBOOK structure
INPUT_INVALID_ERROR	16#00002802	16#00000000	"FirstEntry" input is higher than 999
INPUT_INVALID_ERROR	16#00002803	16#00000000	"NumEntries" input is higher than 1000
INPUT_INVALID_ERROR	16#00002803	16#00000001	"NumEntries" input is lower than 1

Tab. 4-26: Error codes of the ML_ReadDiagLog function block

4.3.2 ML_SetAxisDiagEntry

Brief description

The function causes an entry in the control logbook. At the same time, the diagnostics is displayed on the control display.

Depending on the error class, the axis (drive) referenced by the "Axis" input is affected as follows:



The function may **not** be called cyclically in the PLC program, since an entry is made in the logbook with every call and the logbook overflows!

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/error/diagnostic memory

Tab. 4-27: Library assignment

Interface description

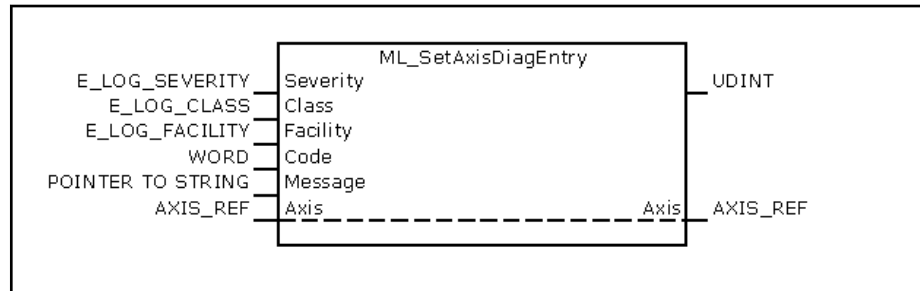


Fig. 4-16: ML_SetAxisDiagEntry function

	Name	Type	Comment
VAR_INPUT	Severity	E_LOG_SEVERITY	Entry type: MESSAGE: 16#A, WARNING: 16#E, ERROR: 16#F E_LOG_SEVERITY on page 121
	Class	E_LOG_CLASS	Class 0..9, determines the error reaction of the control E_LOG_CLASS on page 120
	Facility	E_LOG_FACILITY	Facility for user-defined entries permissible: 16#003B0000 E_LOG_FACILITY on page 121
	Code	WORD	User code 16#0000 16#FFFF
	Message	POINTER TO STRING	User-defined text entry for the logbook. Not more than 59 characters....
RETURN		UDINT	Brief note on error cause
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the desired control and the axis controlled by it.

Tab. 4-28: Interface of the FUN ML_SetAxisDiagEntry

Functional description

The function causes an entry in the control logbook. At the same time, the diagnostics is displayed on the control display.

The diagnostics can be a message or warning, with both types being irrespective of the severity without error reaction.

In case of errors, the system additionally triggers an error reaction according to the error class.

In case of errors and depending on the error class, the axis (drive) referenced by the "Axis" input is affected as follows:

- F03B....: Non-fatal error, no error reaction,
- F13B....: Axis group error, deceleration to standstill of the axis and all other axes belonging to the axis group,

- F23B.... / F33B.... / F43B....: Axis error, the referenced axis is decelerated the best possible,
- F53B.... / F63B.... / F73B....: Control error, **all** axes are decelerated the best possible,
- F83B....: Fatal control error, **all** axes are decelerated the best possible.

Example *F23B4711, Incorrect direction of rotation, RA1*

- **F**: Severity error
- **2**: Class 2, error reaction of the system
 - Logbook entry is generated; message is shown on the display.
 - Axis (or drive) is decelerated the best possible (velocity to 0); all other axes are not affected.
- **4711**: Number according to user specification.
- **Incorrect direction of rotation**: Text in the user text list reference via ADR / POINTER TO STRING
- **RA1**: Axis, excerpt from **MlcVarGlobal** of the control:
(*real axis "RA1" with axis number 1*)
RA1: AXIS_REF := (CntrlNo:=LOCAL_CNTRL, AxisNo:=AXIS_1);

4.3.3 ML_SetDiagEntry

Brief description The function causes an entry in the control logbook. At the same time, the diagnostics is displayed on the control display.



The function may **not** be called cyclically in the PLC program, since an entry is made in the logbook with every call and the logbook overflows!

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/error/diagnostic memory

Tab. 4-29: Library assignment

Interface description

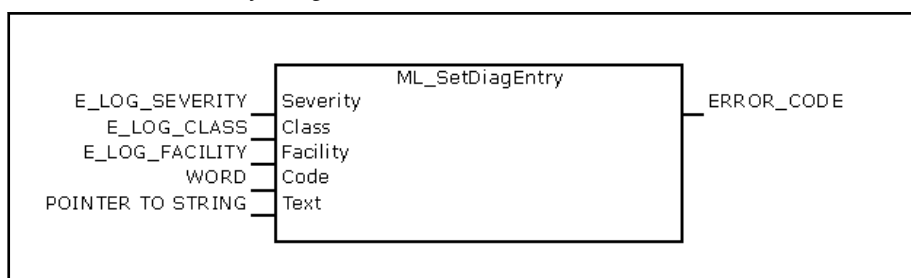


Fig. 4-17: ML_SetDiagEntry function

	Name	Type	Comment
VAR_INPUT	Severity	E_LOG_SEVERITY	Entry type: MESSAGE: 16#A, WARNING: 16#E, ERROR: 16#F E_LOG_SEVERITY on page 121
	Class	E_LOG_CLASS	Class 0..9, determines the error reaction of the control E_LOG_CLASS on page 120

	Name	Type	Comment
	Facility	E_LOG_FACILITY	Facility for user-defined entries permissible: 16#003B0000 E_LOG_FACILITY on page 121
	Code	WORD	User code 16#0000 16#FFFF
	Text	POINTER TO STRING	User-defined text entry for the logbook. Not more than 59 characters....
RETURN		ERROR_CODE	Brief note on error cause

Tab. 4-30: Interface of the FUN ML_SetDiagEntry

Functional description The function causes an entry in the control logbook. At the same time, the diagnostics is displayed on the control display.

The diagnostics can be a message or warning, with both types being irrespective of the severity without error reaction.

In case of errors, the system additionally triggers an error reaction according to the error class.

Useful error classes

- F03B....: Non-fatal error, no error reaction,
- F53B..../F63B..../F73B....: Control error, **all** axes are decelerated the best possible,
- F83B....: Fatal control error, **all** axes are decelerated the best possible.

Example *F53B0815, Synchronism loss*

- **F**: Severity error
- **5**: Class 5, error reaction of the system
 - Logbook entry is generated; message is shown on the display.
 - **All** axes are decelerated the best possible.
- **0815**: Number according to user specification.
- **Synchronicity loss**: Text in the user text list reference via ADR / POINTER TO STRING

4.3.4 E_LOG_CLASS

Brief description The enumeration data type E_LOG_CLASS describes the classes (error reactions) of the diagnostic system of the IndraLogic XLC/IndraMotion MLC system.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/error/diagnostic memory

Tab. 4-31: Library assignment

E_LOG_CLASS (ENUM)

```

TYPE E_LOG_CLASS:
(
  CLASS_0    := 16#0, // no error reaction
  CLASS_1    := 16#1, // see online help for error reaction
  CLASS_2    := 16#2, // see online help for error reaction
  CLASS_3    := 16#3, // see online help for error reaction
  CLASS_4    := 16#4, // see online help for error reaction
  CLASS_5    := 16#5, // see online help for error reaction
  CLASS_6    := 16#6, // see online help for error reaction
  CLASS_7    := 16#7, // see online help for error reaction
)

```

```

CLASS 8      := 16#8      // see online help for error reaction
)DWORD;
END_TYPE

```



Class "9" is not available for the user.

4.3.5 E_LOG_FACILITY

Brief description The enumeration data type E_LOG_SEVERITY describes the diagnostic groups of the diagnostic system of the IndraLogic XLC/IndraMotion MLC system.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/error/diagnostic memory

Tab. 4-32: Library assignment

E_LOG_FACILITY (ENUM)

```

TYPE E_LOG_FACILITY:
(
  USER      := 16#003B0000 // PLC user message
)DWORD;
END_TYPE

```

4.3.6 E_LOG_SEVERITY

Brief description The enumeration data type E_LOG_SEVERITY describes the categories of the diagnostic system of the IndraLogic XLC/IndraMotion MLC system.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/error/diagnostic memory

Tab. 4-33: Library assignment

E_LOG_SEVERITY (ENUM)

```

TYPE E_LOG_SEVERITY:
(
  MESSAGE := 16#A, // Diagnostic identifier for Message "A"
  WARNING := 16#E, // Diagnostic identifier for Warning "E"
  ERROR   := 16#F, // Diagnostic identifier for Error "F"
)DWORD;
END_TYPE

```



Command errors cannot be output by the user.

4.3.7 ML_LOGBOOK

Brief description This structure contains an entry of the diagnostic logbook. It is used by the function block [ML_ReadDiagLog](#) on page 116.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/error/diagnostic memory

Tab. 4-34: Library assignment

Structure element	Type	Description
ErrorNumber	DWORD	Error number
State	UDINT	Status of the diagnostic entry
LogTime	DT	Time of the diagnostics
DiagText	STRING	Diagnostic text

Tab. 4-35: ML_LOGBOOK data structure

4.4 Execute

4.4.1 ML_TaskExecuteFile

Brief description This function executes a file from the file system.

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base / Execute

Tab. 4-36: Library assignment

Interface description

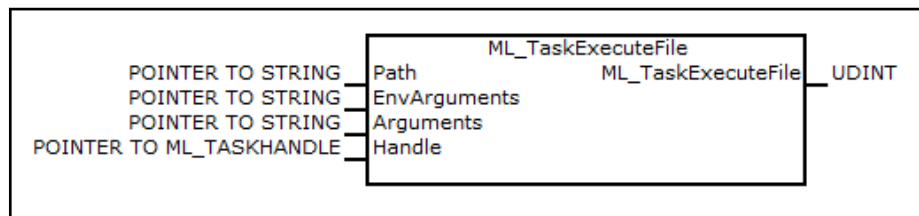


Fig. 4-18: ML_TaskExecuteFile interface

	Name	Type	Comment
VAR_INPUT	Path	POINTER TO STRING	Path to the file to be executed
	EnvArguments	POINTER TO STRING	Arguments for the execution environment
	Arguments	POINTER TO STRING	Arguments for the file to be executed
	Handle	POINTER TO ML_TASKHANDLE	Pointer whereto ML_TASKHANDLE is to be saved ML_TASKHANDLE, page 124
Return value		UDINT	Brief note on error cause

Tab. 4-37: ML_TaskExecuteFile interface

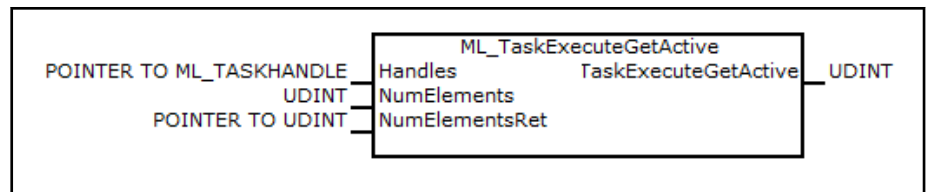
4.4.2 ML_TaskExecuteGetActive

Brief description This function returns the list of all currently executed files

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base / Execute

Tab. 4-38: Library assignment

Interface descriptionFig. 4-19: *ML_TaskExecuteGetActive interface*

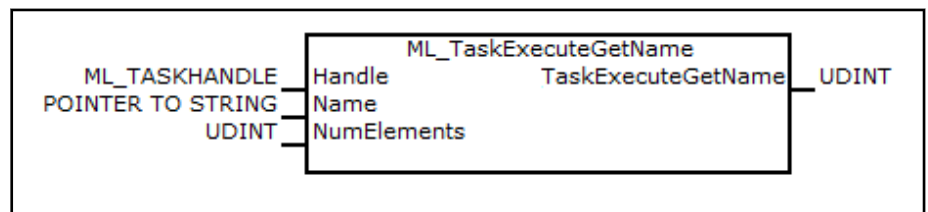
	Name	Type	Comment
VAR_INPUT	Handles	POINTER TO ML_TASKHANDLE	Array saving the task handle. ML_TASKHANDLE, page 124
	NumElements	UDINT	Number of elements in handles
	NumElementsRet	POINTER TO UDINT	Number of elements used in handles
Return value		UDINT	Brief note on error cause

Tab. 4-39: *ML_TaskExecuteGetActive interface***4.4.3 ML_TaskExecuteGetName**

Brief description This function returns the name of a file to be executed.

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base / Execute

Tab. 4-40: *Library assignment***Interface description**Fig. 4-20: *ML_TaskExecuteGetName interface*

	Name	Type	Comment
VAR_INPUT	Handle	ML_TASKHANDLE	Name of the handle to be output ML_TASKHANDLE, page 124
	Name	POINTER TO STRING	Pointer to a string saving the name
	NumElements	UDINT	Number of characters in a name
Return value		UDINT	Brief note on error cause

Tab. 4-41: *ML_TaskExecuteGetName interface***4.4.4 ML_TaskExecuteGetStatus**

Brief description This function returns the status of a file to be executed.

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base / Execute

Tab. 4-42: Library assignment

Interface description



Fig. 4-21: ML_TaskExecuteGetStatus interface

	Name	Type	Comment
VAR_INPUT	Handle	ML_TASKHANDLE	Status of the task to be output ML_TASKHANDLE, page 124
	State	POINTER TO ML_ProcessState	Task handle status ML_ProcessState, page 125
Return value		UDINT	Brief note on error cause

Tab. 4-43: ML_TaskExecuteGetStatus interface

4.4.5 ML_TaskExecuteKill

Brief description

This function stops the execution of a file.

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base / Execute

Tab. 4-44: Library assignment

Interface description

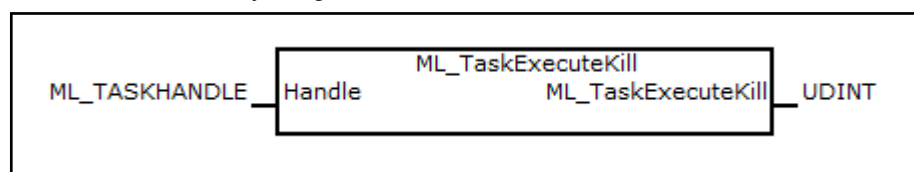


Fig. 4-22: ML_TaskExecuteKill interface

	Name	Type	Comment
VAR_INPUT	Handle	ML_TASKHANDLE	Task handle to be closed ML_TASKHANDLE, page 124
Return value		UDINT	Brief note on error cause

Tab. 4-45: ML_TaskExecuteKill interface

4.4.6 ML_TASKHANDLE

Brief description

This data type contains a definition of a task handle.

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base / Data Types / Execute Types

Tab. 4-46: Library assignment

Type definition

TYPE ML_TASKHANDLE : ULINT; END_TYPE

4.4.7 ML_ProcessState

Brief description

This data type returns the status of the executed file.

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base / Internal (Do not use!) / ParameterImportExport / Datatypes

Tab. 4-47: Library assignment

	Value (UDINT)	Description
PROCESSSTATE_NOT_STARTED	0	File processing not yet started
PROCESSSTATE_INITIALIZATION	32	Initialization running
PROCESSSTATE_INWORK	64	Execution running
PROCESSSTATE_FINISHED	128	Processing completed
PROCESSSTATE_ERROR	256	Processing aborted with errors.

Tab. 4-48: ENUM ML_ProcessState

4.5 Global Variable List

4.5.1 GVL_Base

Brief description

The global variable list **GVL_Base** contains the data to be assigned to an IndraLogic XLC/IndraMotion MLC.

Data referring to axis groups (RobotControl) are not contained (see library ML_Robot.library).

The constants contained in **GVL_Base** are listed in the global list [GVL_Base-Constant](#), page 126, .

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Global Variable List

Tab. 4-49: Library assignment

Declaration of the global variable list "GVL_Base"

```

VAR_GLOBAL
  NoAxis:      AXIS_REF:=      (CntrlNo:= LOCAL_CNTRL, AxisNo:= NO_OBJECT);
  NoTouchProbe: TOUCHPROBE_REF:= (CntrlNo:= LOCAL_CNTRL, TouchProbeNo:= NO_TOUCHPROBE);

  AxisData    AT %MBaaa: ARRAY[MB_MIN_AXIS_NUMBER..MB_MAX_AXIS_NUMBER] OF ML_AXISDATA_SM;
  ControlData AT %MBbbb: ARRAY[MB_MIN_CONTROL_NUMBER..MB_MAX_CONTROL_NUMBER] OF ML_CONTROLDATA_SM;
  PlsData     AT %MBccc: ARRAY[MB_MIN_PLS_NUMBER..MB_MAX_PLS_NUMBER] OF ML_PLSDATA_SM;
  ProbeData   AT %MBddd: ARRAY[MB_MIN_PROBE_NUMBER..MB_MAX_PROBE_NUMBER] OF ML_PROBEDATA_SM;

  DV_Osci:    ARRAY[MB_MIN_OSCI_NUMBER..MB_MAX_OSCI_NUMBER] OF ML_DirectVarOsci;
  DV_Probe:   ARRAY[MB_MIN_PROBE_NUMBER..MB_MAX_PROBE_NUMBER] OF ML_DirectVarProbe;
  DV_Pls:     ARRAY[MB_MIN_PLS_NUMBER..MB_MAX_PLS_NUMBER] OF ML_DirectVarPls;
  DV_Control: ARRAY[MB_MIN_CONTROL_NUMBER..MB_MAX_CONTROL_NUMBER] OF ML_DirectVarControl;
  DV_Axis:    ARRAY[MB_MIN_AXIS_NUMBER..MB_MAX_AXIS_NUMBER] OF ML_DirectVarAxis;

  GV_LinkAxis:ARRAY[MB_MIN_GLOBAL_CONTROL_NUMBER..MB_MAX_GLOBAL_CONTROL_NUMBER,
                    MB_MIN_AXIS_NUMBER..MB_MAX_AXIS_NUMBER] OF ML_GlobalVarAxis;

  GlobalVarDirectVarData: ML_GlobalVarDirectVarData;
  BaseDeviceSetup: BaseDeviceSetup;
END_VAR

```



For the addresses of "AxisData", "ControlData", "PlsData" and "ProbeData", refer to the library manager of the BASELIB= ML_Base / Global Variables List / GVL_Base.

The array limits are located in the system-specific library parts BASELIBCONST= ML_Base_Constant_xxx / Global Variable Lists / GVL_BaseConstant.

NoAxis	Default initial value
NoTouchProbe	Default initial value
GV_LinkAxis	GV_LinkAxis[] - axis data via synchronous function variable access in C2C link , page 90
GlobalVarDirectVarData	Internal use
Axis data via shared memory and function variable access, page 83	
AxisData	ML_AXISDATA_SM , page 110
DV_Axis	DV_Axis[] - axis data via function variable access, page 87
Control data via shared memory and function variable access, page 92	
ControlData	ML_CONTROLDATA_SM , page 111
DV_Control	DV_Control[] - control data via function variable access , page 94
Programmable limit switch data via shared memory and function variable access, page 97	
PlsData	ML_PLSDATA_SM , page 112
DV_Pls	DV_Pls[] - PLS data via function variable access, page 100
Touch probe data via shared memory and function variable access, page 102	
ProbeData	ML_PROBEDATA_SM , page 113
DV_Probe	DV_Probe[] - touch probe data via function variable access, page 105
Oscilloscope data via function variable access, page 107	
DV_Osci	DV_Osci[] - oscilloscope data via function variable access, page 108

Tab. 4-50: Components of the "GVL_Base"

4.5.2 GVL_BaseConstant

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base_Constant_xxx/Global Variable List xxx= xlc, mlc, vpx, xm2x

Tab. 4-51: Library assignment

Brief description

The global list **GVL_BaseConstant** contains the system-dependent constant with regard to the respective hardware (e.g. the maximum number of axes the control can operate, the number of kinematics and programmable limit switches)

The constants supplement information on the memory areas for the respective control type:

- [Axis data via shared memory and function variable access on page 83](#)
- [Control data via shared memory and function variable access on page 92](#)

- [Touch probe data via shared memory and function variable access on page 102](#)
- [Programmable limit switch data via shared memory and function variable access on page 97](#)
- [chapter "Oscilloscope data via function variable access" on page 107](#)
- Kinematic data in the shared memory access and in the function variable access (ML_Robot.library).



For more information on the respective systems and versions, refer to the data in the library manager.

New: The constant "IL_TARGET_NAME" includes the control name.

Currently available are:

- IndraMotion_MLD_E // MLD electrical
- IndraMotion_MLD_H // MLD hydraulic
- IndraMotion_MLC_L25
- IndraMotion_MLC_L45
- IndraMotion_MLC_L65
- IndraMotion_MLC_L75
- IndraLogic_XLC_L25
- IndraLogic_XLC_L45
- IndraLogic_XLC_L65
- IndraLogic_XLC_L75
- IndraMotion_MLC_VPx
- IndraMotion_MLC_XM2

4.5.3 Global Variable Lists/Global Variable Index

GVL_A_GlobalVarIdx

Brief description

The global constant list contains the address constant assignment to the respective parameter in "Synchronous status during function variable access in the C2C link".

The parameter constant name is

A-0-0021 ⇒ GV_A_0_0021, etc. ...

Number and parameter constant name can be used alternatively.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Global Variable List, Global Variable Index

Tab. 4-52: Library assignment

Declaring the global constant list GVL_A_GlobalVarIdx

```
VAR_GLOBAL CONSTANT
  GV_A_0_0021: UDINT:= 0 ; (* Axis status *)
  GV_A_0_0045: UDINT:= 1 ; (* Modulo value *)
  GV_A_0_0046: UDINT:= 2 ; (* Feed distance of the slave drive *)
  GV_A_0_0101: UDINT:= 3 ; (* Actual position value, 8 bytes *)
  GV_A_0_0102: UDINT:= 4 ; (* Actual velocity value *)
  GV_A_0_2121: UDINT:= 5 ; (* Axis status *)
  GV_A_0_2800: UDINT:= 6 ; (* Master axis encoder, configuration *)
```

```

GV_A_MIN_IDX: UDINT:= 0 ;
GV_A_MAX_IDX: UDINT:= 6 ;
GV_A_MAX_CNT: UDINT:= 7 ;
END_VAR

```

4.5.4 Parameter constants

Access to data of an axis, control or another object

Structure of and access to parameters of the PLC program

The parameter data of an axis, a control or a programmable limit switch can be accessed by read or write access. The following parameter types exist in IndraMotion MLC:

- C-parameters (control parameters), referring to an actual control affecting the axes.
- A-parameters (axis parameters), each referring to one particular axis.
- K-parameters (kinematics parameters) that relate to a kinematics (axis group), see ML_Robot.library.
- M parameters (touch probe parameters), each relating to one touch probe. They can only be accessed via PLCopen function blocks.
- N-parameters (pls parameters), each referring to one programmable limit switch.
- O-parameters (oscilloscope parameters), each referring to one oscilloscope function.

In addition to these parameters, there are other groups which relate to devices on a specific bus (S-parameters, Sercos interface) or to a specific product.

The structure of all IndraMotion parameters is identical:

Example: A-0-0001, with "A" as a parameter type ID which is followed by the group number "0" and the four-digit parameter number "0001".

Based on the parameter number, access to these parameters is possible in the PLC program.

Declaration files for Global Constants

- [GVL_A_Param_ID](#), page 130,
- [GVL_C_Param_ID](#), page 130,
- [GVL_O_Param_ID](#), page 131,
- [GVL_P_Param_ID](#), page 132,
- [GVL_S_Param_ID](#), page 132.

If the parameter number is known:

A-parameters address constants

```

VAR GLOBAL CONSTANT
  FP_A_0_0000: MB_IDN:= 4294967296 ; (* Dummy A parameter *)
  FP_A_0_0001: MB_IDN:= 4294967297 ; (* Logical axis number *)
  FP_A_0_0002: MB_IDN:= 4294967298 ; (* Axis name *)
  (*..... *)
END_VAR

```

As an alternative to the constant, the number can also be attached to the input "ParameterNumber" of the respective function block.

Offsets der Parameter

Parameter	Decimal	Hexadecimal
S-0-0000	0	16#0
P-0-0000	32768	16#8000
A-0-0000	4294967296	16#1 0000 0000
C-0-0000	8589934592	16#2 0000 0000
K-0-0000	184683593728	16#2B 0000 0000
M-0-0000	193273528320	16#2D 0000 0000
N-0-0000	197568495616	16#2E 0000 0000
O-0-0000	201863462912	16#2F 0000 0000

Tab. 4-53: Parameter offsets

Several axes can be actuated with the control. Thus, the function blocks are provided with an address input (VAR_IN_OUT) of type AXIS_REF, RIL_CommonTypes.library which has to be assigned as follows:

Access to A-parameters:

- The address identifier is automatically generated by IndraWorks ML* (axis name/axis number), type AXIS_REF.

Access to the N Parameters:

- The address identifier is automatically generated by IndraWorks ML* or can instead be defined by the user (object name/object number), type AXIS_REF.

Access to O parameters:

- The address identifier is automatically generated by IndraWorks ML* or can instead be defined by the user (object name/object number), type AXIS_REF.

Access to C-parameters:

- Use the address identifier of an arbitrary axis of this control or declaration with values.
 - CntrlNo:= LOCAL_CNTRL or CNTRL_1...
 - AxisNo:= NO_OBJECT.

References at axis input, example: MB_ReadParameter

```

VA1:AXIS_REF:=(CntrlNo:=LOCAL_CNTRL,AxisNo:=AXIS_1)
(*A parameters, axis VA1, axis number 1*);

MLC_A: AXIS_REF:=(CntrlNo:=LOCAL_CNTRL,AxisNo:=NO_OBJECT)
(* C parameters, control MLC_A *);

PLS_A: AXIS_REF:=(CntrlNo:=LOCAL_CNTRL,AxisNo:=PLS_1)
(*N parameters, programmable limit switch PLS_A, obj.no.1*);

OSZ1: AXIS_REF:=(CntrlNo:=LOCAL_CNTRL,AxisNo:=OSCI_CHANNEL_1)
(*O parameters, oscilloscope OSZ1, obj.no.1*);

```

GVL_A_Param_ID

- Brief description** The global constant list contains the address constant assignment to the respective parameter.
The parameter constant name is
A-0-0001 ⇒ FP_A_0_0001, etc.
Number and parameter constant name can be used alternatively.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Global Variable List, parameter ID

Tab. 4-54: Library assignment

Declaring the global constant list GVL_A_Param_ID

```
VAR GLOBAL CONSTANT
  FP_A_0_0000: MB_IDN:= 4294967296 ; (* Dummy A parameter *)
  FP_A_0_0001: MB_IDN:= 4294967297 ; (* Logical axis number *)
  FP_A_0_0002: MB_IDN:= 4294967298 ; (* Axis name *)
  (*..... *)
END_VAR
```

See also:

- [MB_IDN, page 75](#)

GVL_C_Param_ID

- Brief description** The global constant list contains the address constant assignment to the respective parameter.
The parameter constant name is
C-0-0001 ⇒ FP_C_0_0001 etc. ...
Number and parameter constant name can be used alternatively.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Global Variable List, parameter ID

Tab. 4-55: Library assignment

Declaring the global constant list GVL_C_Param_ID

```
VAR GLOBAL CONSTANT
  FP_C_0_0000: MB_IDN:= 8589934592 ; (* Dummy C parameter *)
  FP_C_0_0001: MB_IDN:= 8589934593 ; (* Language selection *)
  FP_C_0_0002: MB_IDN:= 8589934594 ; (* Name of parameter file *)
  (*..... *)
END_VAR
```

C-0-0000, C-0-0001, C-0-0002,...

See also:

- [MB_IDN, page 75](#)

GVL_M_Param_ID

- Brief description** The global constant list contains the address constant assignment to the respective parameter.
The parameter constant name is

M-0-0001 ⇒ FP_M_0_0001, etc.

Number and parameter constant name can be used alternatively.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Global Variable List, parameter ID

Tab. 4-56: Library assignment

Declaring the global constant list GVL_M_Param_ID

```
VAR GLOBAL CONSTANT
  FP_M_0_0000: MB_IDN:= 193273528320 ; (* Dummy M parameter *)
  FP_M_0_0001: MB_IDN:= 193273528321 ; (* Logical probe number *)
  FP_M_0_0002: MB_IDN:= 193273528322 ; (* Probe name *)
  (* ..... *)
END_VAR
```

See also:

- [MB_IDN, page 75](#)

GVL_N_Param_ID

Brief description

The global constant list contains the address constant assignment to the respective parameter.

The parameter constant name is

N-0-0001 ⇒ FP_N_0_0001, etc.

Number and parameter constant name can be used alternatively.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Global Variable List, parameter ID

Tab. 4-57: Library assignment

Declaring the global constant list GVL_N_Param_ID

```
VAR GLOBAL CONSTANT
  FP_N_0_0000: MB_IDN:= 197568495616 ; (* Dummy N parameter *)
  FP_N_0_0001: MB_IDN:= 197568495617 ; (* Logical number, PLS *)
  FP_N_0_0002: MB_IDN:= 197568495618 ; (* Name, PLS *)
  (* ..... *)
END_VAR
```

See also:

- [MB_IDN, page 75](#)

GVL_O_Param_ID

Brief description

The global constant list contains the address constant assignment to the respective parameter.

The parameter constant name is

O-0-0001 ⇒ FP_O_0_0001, etc.

Number and parameter constant name can be used alternatively.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Global Variable List, parameter ID

Tab. 4-58: Library assignment

Declaring the global constant list GVL_O_Param_ID

```

VAR_GLOBAL CONSTANT
  FP_O_0_0000: MB_IDN:= 201863462912 ; (* Dummy O parameter *)
  FP_O_0_0001: MB_IDN:= 201863462913 ; (* Logical channel number, Oscilloscope *)
  FP_O_0_0002: MB_IDN:= 201863462914 ; (* Channel name, Oscilloscope *)
  (* ..... *)
END_VAR

```

See also:

- [MB_IDN, page 75](#)

GVL_P_Param_ID**Brief description**

The global constant list contains the address constant assignment to the respective parameter.

The parameter constant name is

P-0-0001 ⇒ FP_P_0_0001 etc. ...

Number and parameter constant name can be used alternatively.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Global Variable List, parameter ID

Tab. 4-59: Library assignment

Declaring the global constant list GVL_P_Param_ID

```

VAR_GLOBAL CONSTANT
  FP_P_0_0001: MB_IDN:= 32769 ; (* Switching frequency of the power output stage *)
  FP_P_0_0002: MB_IDN:= 32770 ; (* SPI flash aging counter *)
  FP_P_0_0003: MB_IDN:= 32771 ; (* Status of parameter buffering *)
  FP_P_0_0004: MB_IDN:= 32772 ; (* Velocity loop smoothing time constant *)
  (* ..... *)
END_VAR

```

See also:

- [MB_IDN, page 75](#)

GVL_S_Param_ID**Brief description**

The global constant list contains the address constant assignment to the respective parameter.

The parameter constant name is

S-0-0001 ⇒ FP_S_0_0001, etc.

Number and parameter constant name can be used alternatively.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/Global Variable List, parameter ID

Tab. 4-60: Library assignment

Declaring the global constant list GVL_S_Param_ID

```

VAR_GLOBAL CONSTANT
  FP_S_0_0000: MB_IDN:= 0 ; (* Dummy S parameter *)
  FP_S_0_0001: MB_IDN:= 1 ; (* NC cycle time (TNcyc) *)
  FP_S_0_0002: MB_IDN:= 2 ; (* SERCOS cycle time (TScyc) *)
  (* ..... *)
END_VAR

```

See also:

- [MB_IDN](#), page 75

4.6 Interfaces, functions and function blocks

4.6.1 IL_SercosAttribute

Brief description

The IL_SIIAttribute function block breaks down the elements of the Sercos attribute.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/interface

Tab. 4-61: Library assignment

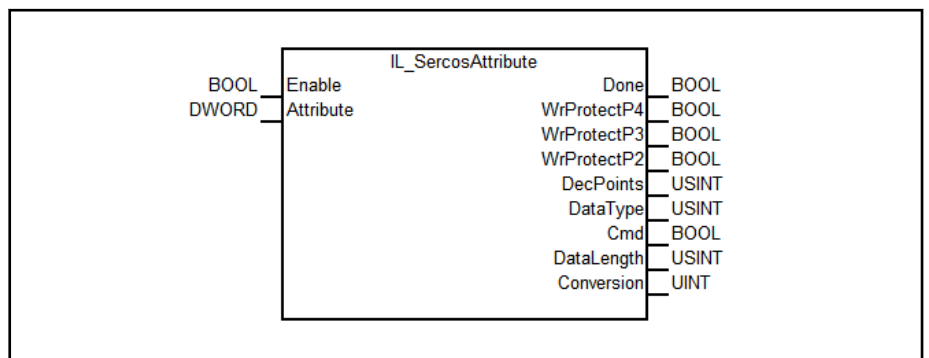
Interface description

Fig. 4-23: IL_SercosAttribute function block

	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Release for processing of the function (cyclic, state-controlled)
	Attributes	DWORD	Sercos attribute
VAR_OUPUT	Done	BOOL	Breakdown of the Sercos attribute completed, output variables valid
	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

	Name	Data type	Description
	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	USINT	Data type and display format of the operating date of the parameter
	Cmd	BOOL	Operating date calls for command function
	DataLength	USINT	Data length of the operating date of the parameter
	Conversion	UINT	Evaluation factor of the operating date of the parameter

Tab. 4-62: IL_SercosAttribute interface

Sercos attribute The following table shows the structure of the Sercos attribute.

Bit	Interface variable	Description
30:	WrProtectP4	Write-protection of the parameter operating date in the operating mode (Sercos phase 4)
		0: Operating date can be written 1: Operating date is write-protected
29:	WrProtectP3	Write-protection of the operating date of the parameter (Sercos phase 3)
		0: Operating date can be written 1: Operating date is write-protected
28:	WrProtectP2	Write-protection of the parameter operating date in the parameterization mode (Sercos phase 2)
		0: Operating date can be written 1: Operating date is write-protected
27..24:	DecPoints	The decimal places define the position of the comma for the display and input of the respective operating date.
		0000 = No decimal places
		1111: 15 decimal places (maximum)
23:		Reserved
22..20:	DataType	Data type and display format are used to convert the operating date as well as the minimum and maximum input value to the correct display format.
		Data type
		Display format
		000: Binary number
		Binary
		001: Unsigned integer
		Decimal places without signs
		010: Integer
		Decimal places with signs
		011: Unsigned integer
		hexadecimal
		100: Extended character set
		Text
		101: Unsigned integer
		Parameter ident number format (Sercos standard, 2 byte)
		110: Floating point number
		Decimal places with signs
		111: Unsigned integer
		Parameter ident number format (Sercos standard, 4 bytes)

Bit	Interface variable	Description
19:	Cmd	The function of the operating date indicates that a command function is called for in the drive with this operating date. 0: Operating date or parameter 1: Command
18..16:	DataLength	The data length is required by the control for the regular completion of the data transmission via the service channel. 000: Reserved 001: Operating date is 2 byte long 010: Operating date is 4 byte long 011: Reserved 100: Variable length with 1 byte data 101: Variable length with 2-byte data 110: Variable length with 4-byte data 111: Reserved
15..0:	Conversion	The evaluation factor is an unsigned integer which serves for the conversion of numeric data to a display format. The evaluation factor is set to the value 1 if it is not used for the display of the dates.

Tab. 4-63: Elements of the Sercos attribute

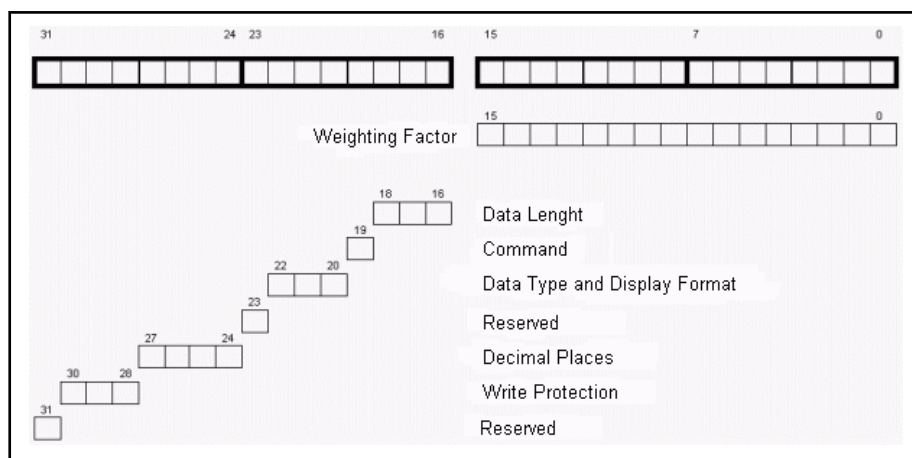


Fig. 4-24: Structure of the Sercos attribute

Functional description The function block IL_SercosAttribute cyclically interprets the Sercos attribute created at the input "Attribute" after release for processing of the function block with "Enable", the validity of the elements at the corresponding outputs is signaled by the output "Done".

Troubleshooting No error handling is available for this function block.

4.6.2 IL_SercosElementsToParaNo

Brief description The function IL_SercosElementsToParaNo is used to combine the individual elements of a Sercos-parameter string and the simultaneous conversion to the Sercos-parameter ident number.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/interface

Tab. 4-64: Library assignment

Interface description

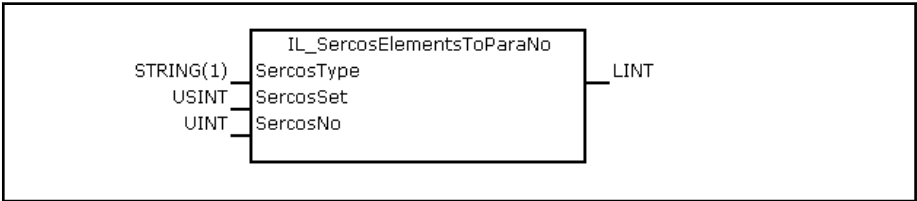


Fig. 4-25: Function IL_SercosElementsToParaNo

	Name	Data type	Description
VAR_INPUT	SercosType	STRING(1)	Parameter type
	SercosSet	USINT	Parameter set
	SercosNo	UINT	Parameter number
RETURN		LINT	Parameter ident number

Tab. 4-65: Interface IL_SercosElementsToParaNo

- Specification

The function IL_SercosElementsToParaNo allows for the combination of the following individual elements of a Sercos parameter string:
 - SercosType: "S", "P", "A", "C", "Y", "T", "s", "p", "a", "c", "y", "t"
 - SercosSet: 0...7
 - SercosNo: 0...4095
- Functional description

The function IL_SercosElementsToParaNo continuously puts the connected input elements together to get the corresponding Sercos parameter ident number.
- Troubleshooting

The function IL_SercosElementsToParaNo generates no error for the operation time. In case of invalid individual elements at the inputs (see specification), the function returns "-1".
- Use case

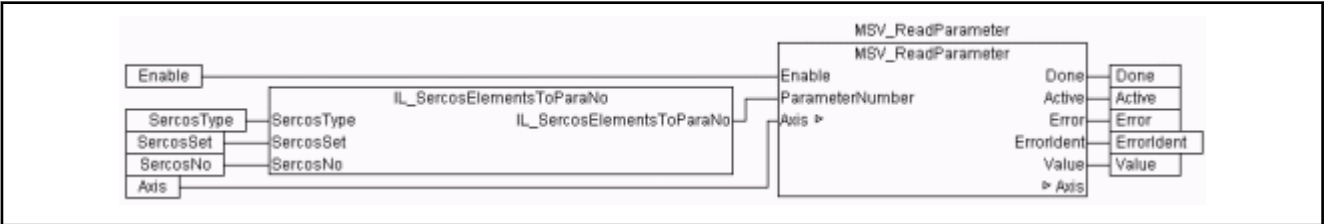


Fig. 4-26: Application of the function (CFC): IL_SercosElementsToParaNo

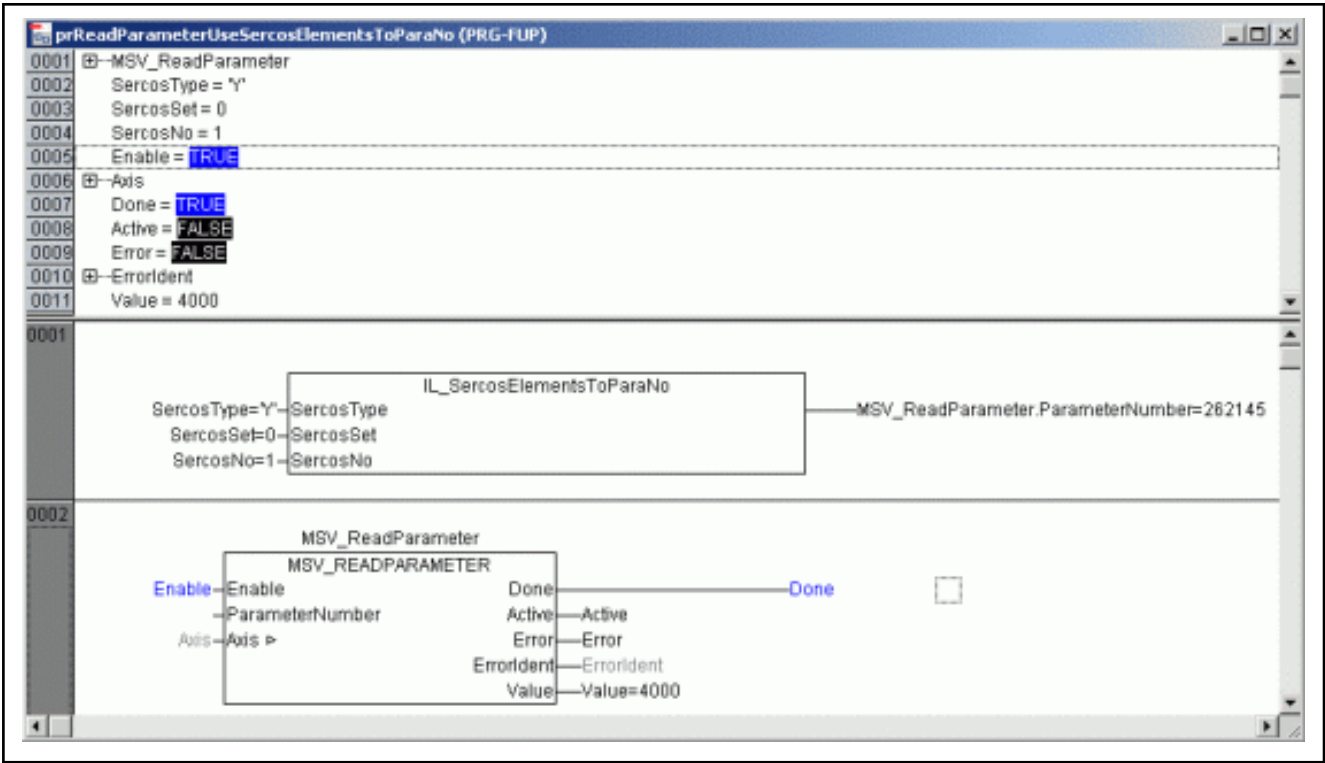


Fig. 4-27: Application of the function (FBD): IL_SercosElementsToParaNo

4.6.3 IL_SercosStringToParaNo

Brief description The function IL_SercosStringToParaNo is used to convert a Sercos parameter string to the Sercos parameter ident number.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/interface

Tab. 4-66: Library assignment

Interface description

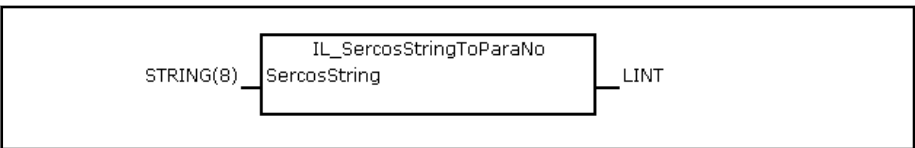


Fig. 4-28: Function IL_SercosStringToParaNo

	Name	Data type	Description
VAR_INPUT	SercosString	STRING[8]	Sercos parameter string
RETURN		LINT	Parameter ident number

Tab. 4-67: Interface IL_SercosStringToParaNo

Specification The function IL_SercosStringToParaNo can be used to convert a Sercos parameter string with the following string elements (formatting: "X Y ZZZZ" and/or "XZZZZ"):

- SercosType "X": "S", "P", "A", "C", "Y", "T", "s", "p", "a", "c", "y", "t"
- SercosSet "Y": "0"... "7"
- SercosNo "ZZZZ": "0"... "4095"
- Separator: "-"

- Functional description

The function IL_SercosElementsToParaNo continuously combines the connected input elements to achieve the corresponding Sercos parameter ident number.
- Troubleshooting

The function IL_SercosStringToParaNo generates no error for the operation time. In case of an invalid Sercos parameter string (see specification), the function returns "-1".

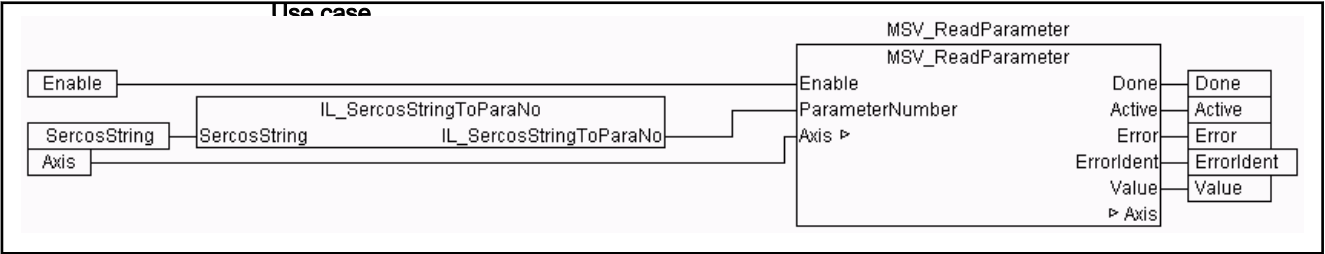


Fig. 4-29: Application of the function (CFC): IL_SercosStringToParaNo

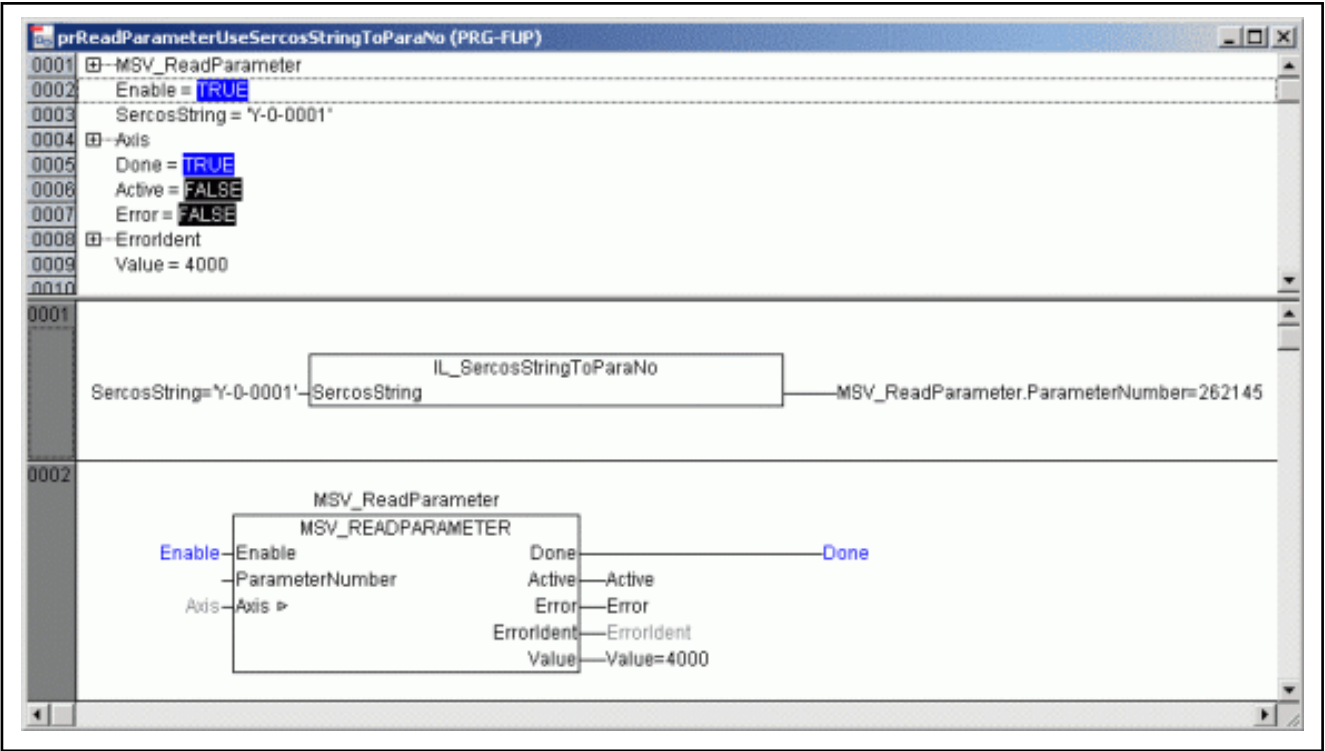


Fig. 4-30: Application of the function (FBD): IL_SercosStringToParaNo

4.7 IP settings - IP configuration

4.7.1 ML_GetGateway

- Brief description

This function block reads the gateway address of the control. The gateway is determined by reading the parameter C-0-0202.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS	ML_Base/IP settings
PLC basic package	

Tab. 4-68: Library assignment

Interface description

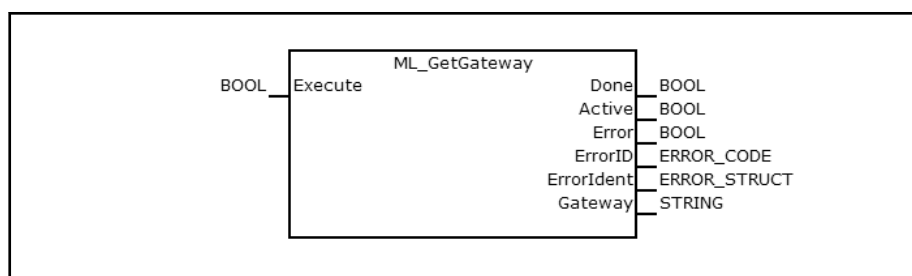


Fig. 4-31: ML_GetGateway function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing enabled
VAR_OUTPUT	Done	BOOL	Operation executed
	Active	BOOL	Confirmation of processing enabled
	Error	BOOL	Indicates that an error occurred in the function block.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Gateway	STRING[255]	Read gateway address:

Tab. 4-69: ML_GetGateway interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous

Tab. 4-70: Input behavior of the ML_GetGateway

Functional description

This function block reads the gateway address of the control. The gateway is determined by reading the parameter C-0-0202.

The ML_GetGateway function block transfers the gateway address of the control after enabling the processing of the function block via "Execute". After a successful processing of the function block, the output is set to "Done" and the "Gateway" output is valid.

If an error occurs while processing the function block, this is signaled by the "Error" output and specified by the outputs "ErrorID" and "ErrorIdent" updated in this case.

Gateway

The "Gateway" output returns the gateway address of the control. The specification is generally given as IP address. It consist of four decimal numbers between 0 and 255 that are separated by points.

Example:

192.168.1.254

Error codes

The function block uses the error table [MLC_TABLE, 16#0030, page 360](#).

4.7.2 ML_GetIpAddress

Brief description

This function block reads the IP address of the control. The IP address is determined by reading the parameter C-0-0200.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/IP settings

Tab. 4-71: Library assignment

Interface description

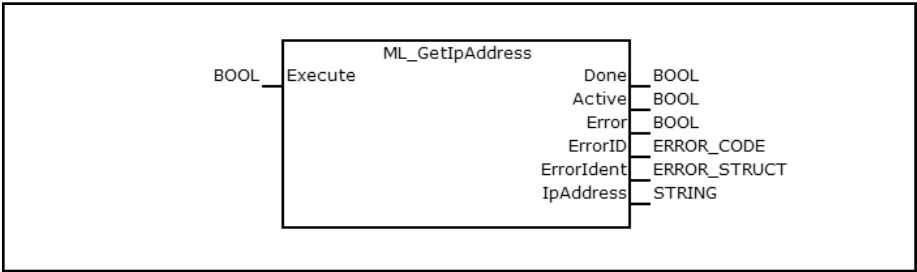


Fig. 4-32: ML_GetIpAddress function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing enabled
VAR_OUTPUT	Done	BOOL	Operation executed
	Active	BOOL	Confirmation of processing enabled
	Error	BOOL	Indicates that an error occurred in the function block.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	IpAddress	STRING[255]	Read IP address

Tab. 4-72: ML_GetIpAddress Interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous

Tab. 4-73: Input behavior of the ML_GetIpAddress

Functional description

This function block reads the IP address of the control. The IP address is determined by reading the parameter C-0-0200.

The "ML_GetIpAddress" function block transfers the IP address mask of the engineering control port control after enabling the processing of the function block via "Execute". After a successful processing of the function block, the output is set to "Done" and the "IpAddress" output is valid.

If an error occurs while processing the function block, this is signaled by the "Error" output and specified by the outputs "ErrorID" and "ErrorIdent" updated in this case.

IpAddress

The "IpAddress" output returns the IP address of the control. This is generally specified as four decimal numbers between 0 and 255 that are separated by points.

Example:

192.168.1.1

Error codes

The function block uses the error table [MLC_TABLE](#), 16#0030, page 360.

4.7.3 ML_GetMacAddress

Brief description This function block reads the MAC address of the control. The MAC address is determined by reading the parameter C-0-0205.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/IP settings

Tab. 4-74: Library assignment

Interface description

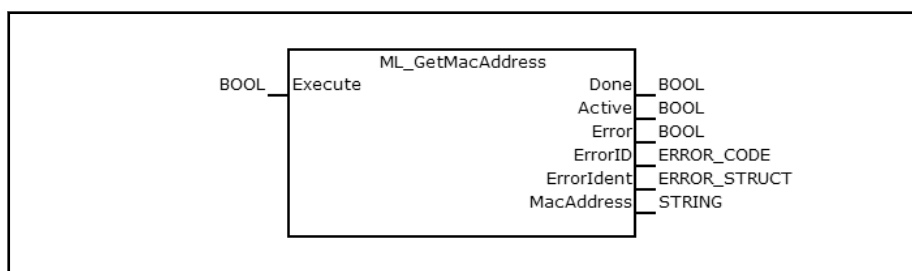


Fig. 4-33: ML_GetMacAddress function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing enabled
VAR_OUTPUT	Done	BOOL	Operation executed
	Active	BOOL	Confirmation of processing enabled
	Error	BOOL	Indicates that an error occurred in the function block.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	MacAddress	STRING[255]	Read MAC address

Tab. 4-75: ML_GetMacAddress interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous

Tab. 4-76: Input behavior of the ML_GetMacAddress

Functional description This function block reads the MAC address of the control. The MAC address is determined by reading the parameter C-0-0205.

The "ML_GetMacAddress" function block transfers the MAC address of the engineering control port after enabling the processing of the function block via "Execute". After a successful processing of the function block, the output is set to "Done" and the "MacAddress" output is valid.

If an error occurs while processing the function block, this is signaled by the "Error" output and specified by the outputs "ErrorID" and "ErrorIdent" updated in this case.

MacAddress

The "MacAddress" output returns the MAC address of the control. This is generally specified as six hexadecimal numbers between 00 and FF that are separated by colons.

Example:

00:30:D6:02:8D:FC

Error codes The function block uses the error table [MLC_TABLE, 16#0030, page 360](#).

4.7.4 ML_GetSubnetMask

Brief description This function block reads the subnet mask of the control. The subnet mask is determined by reading the parameter C-0-0201.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/IP settings

Tab. 4-77: Library assignment

Interface description

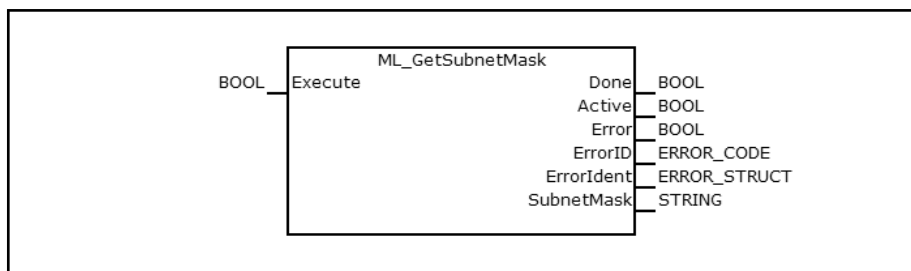


Fig. 4-34: ML_GetSubnetMask function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing enabled
VAR_OUTPUT	Done	BOOL	Operation executed
	Active	BOOL	Confirmation of processing enabled
	Error	BOOL	Indicates that an error occurred in the function block.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	SubnetMask	STRING[255]	Read Subnet mask

Tab. 4-78: ML_GetSubnetMask interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous

Tab. 4-79: Input behavior of the ML_GetSubnetMask

Functional description This function block reads the subnet mask of the control. The subnet mask is determined by reading the parameter C-0-0201.

The "ML_GetSubnetMask" function block transfers the subnet mask of the engineering control port after enabling the processing of the function block via "Execute". After a successful processing of the function block, the output is set to "Done" and the "SubnetMask" output is valid.

If an error occurs while processing the function block, this is signaled by the "Error" output and specified by the outputs "ErrorID" and "ErrorIdent" updated in this case.

SubnetMask

The "SubnetMask" output returns the subnet mask of the control. This is generally specified as four decimal numbers between 0 and 255 that are separated by points.

Example:

192.168.1.255

Error codes The function block uses the error table [MLC_TABLE, 16#0030, page 360](#).

4.7.5 ML_SetGateway

Brief description This function block sets the gateway address of the control. The gateway is set by writing the parameter C-0-0202.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/IP settings

Tab. 4-80: Library assignment

Interface description

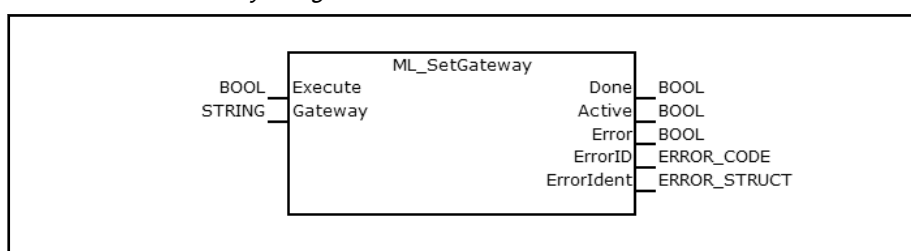


Fig. 4-35: ML_SetGateway function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing enabled
	Gateway	STRING[255]	Gateway address to be written
VAR_OUTPUT	Done	BOOL	Operation executed
	Active	BOOL	Confirmation of processing enabled
	Error	BOOL	Indicates that an error occurred in the function block.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 4-81: ML_SetGateway interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
Gateway	STRING			"	Rising edge at "Execute"

Tab. 4-82: Input behavior of the ML_SetGateway

Functional description This function block sets the gateway address of the control. The gateway is set by writing the parameter C-0-0202.

The "ML_SetGateway" function block sets the gateway address of the control after enabling the processing of the function block via "Execute". After a successful processing of the function block, the output is set to "Done".

If an error occurs while processing the function block, this is signaled by the "Error" output and specified by the outputs "ErrorID" and "ErrorIdent" updated in this case.

Gateway

At the "Gateway" input, the gateway address to be set is transferred to the control. The specification is generally given as IP address. It consists of four decimal numbers between 0 and 255 that are separated by points.

Example:

192.168.1.254

Error codes The function block uses the error table [MLC_TABLE, 16#0030, page 360](#).

4.7.6 ML_SetIpAddress

Brief description This function block sets the IP address of the engineering control port. The gateway is set by writing the parameter C-0-0200.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/IP settings

Tab. 4-83: Library assignment

Interface description

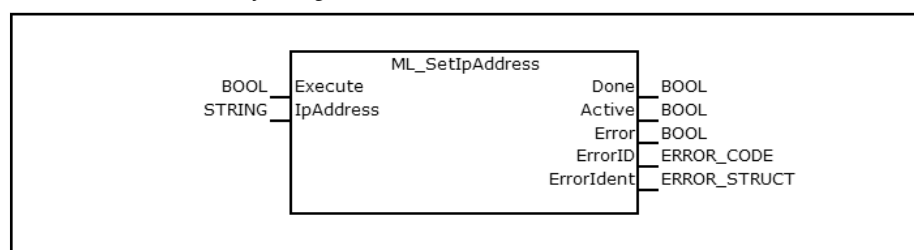


Fig. 4-36: ML_SetIpAddress function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing enabled
	IpAddress	STRING[255]	IP address to be written
VAR_OUTPUT	Done	BOOL	Operation executed
	Active	BOOL	Confirmation of processing enabled
	Error	BOOL	Indicates that an error occurred in the function block.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 4-84: ML_SetIpAddress Interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
IpAddress	STRING			"	Rising edge at "Execute"

Tab. 4-85: Input behavior of the ML_SetIpAddress

Functional description This function block sets the IP address of the engineering control port. The IP address is set by writing the parameter C-0-0200.

The "ML_SetIpAddress" function block sets the IP address of the control after enabling the processing of the function block via "Execute". After a successful processing of the function block, the output is set to "Done".

If an error occurs while processing the function block, this is signaled by the "Error" output and specified by the outputs "ErrorID" and "ErrorIdent" updated in this case.

IpAddress

At the "IpAddress" input, the IP address to be set is transferred to the control. This is generally specified as four decimal numbers between 0 and 255 that are separated by points.

Example:

192.168.1.1

Error codes The function block uses the error table [MLC_TABLE, 16#0030, page 360](#).

4.7.7 ML_SetSubnetMask

Brief description This function block sets the subnet mask of the control. The subnet mask is set by writing the parameter C-0-0201.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/IP settings

Tab. 4-86: Library assignment

Interface description

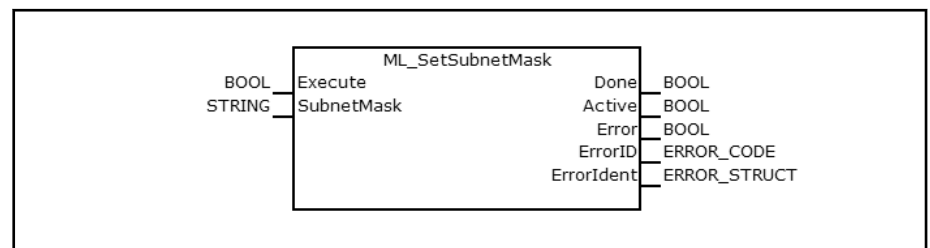


Fig. 4-37: ML_SetSubnetMask function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing enabled
	SubnetMask	STRING[255]	Subnet mask to be written
VAR_OUTPUT	Done	BOOL	Operation executed
	Active	BOOL	Confirmation of processing enabled
	Error	BOOL	Indicates that an error occurred in the function block.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 4-87: ML_SetSubnetMask interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
SubnetMask	STRING			"	Rising edge at "Execute"

Tab. 4-88: Input behavior of the ML_SetSubnetMask

Functional description This function block sets the subnet mask of the control. The subnet mask is set by writing the parameter C-0-0201.

The "ML_SetSubnetMask" function block sets the subnet mask of the control after enabling the processing of the function block via "Execute". After a successful processing of the function block, the output is set to "Done".

If an error occurs while processing the function block, this is signaled by the "Error" output and specified by the outputs "ErrorID" and "ErrorIdent" updated in this case.

SubnetMask

At the "SubnetMask" input, the subnet mask to be set is transferred to the control. The specification is generally given as IP address. It consist of four decimal numbers between 0 and 255 that are separated by points.

Example:

192.168.1.255

Error codes The function block uses the error table [MLC_TABLE, 16#0030, page 360](#).

4.8 SystemTime - system time

4.8.1 IL_SetDateTime

Brief description This function block sets the system time of the control and of the connected drives.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/SystemTime

Tab. 4-89: Library assignment

Interface description

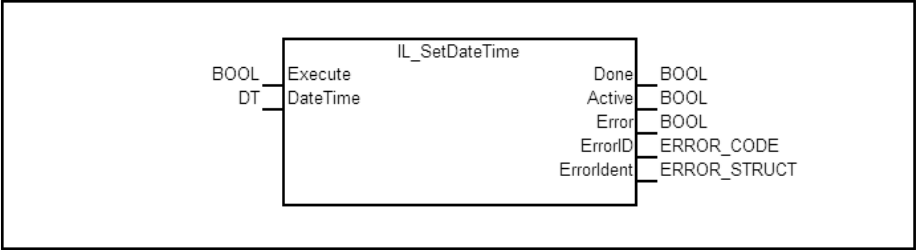


Fig. 4-38: IL_SetDateTime function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing enabled
	DateTime	DT	Time to be set
VAR_OUTPUT	Done	BOOL	Operation executed
	Active	BOOL	Confirmation of processing enabled
	Error	BOOL	Indicates that an error occurred in the function block.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 4-90: IL_SetDateTime interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
DateTime	DT	DT#1970-01-01-00:00:0		DT#1970-01-01-00:00:00	Rising edge at "Execute"

Tab. 4-91: IL_SetDateTime input behavior

The IL_SetDateTime function block sets the time on the control, the connected drives as well as the battery-buffered real-time clock at the "DateTime" input.

If "DateTime" DT#1970-01-01-00:00:00, the current control time is applied to the real-time clock and the drives.

Processing is started in case of a rising edge at "Execute".

The time specified at the "DateTime" input is first set in the control and then applied to the real-time clock. Subsequently, the time is transmitted to all connected drives.

Not all drives support setting of the time. These drives are not described. Processing takes several PLC cycles.

The "Done" output is set after successful processing.

If an error occurs during processing, the "Error" output is set and the error code is reported at the "ErrorID" and "ErrorIdent" outputs.

Error codes The function block uses the error code table [MLC_TABLE, 16#0030, page 360](#) as well as the error code table F_RELATED_TABLE, 16#0170

4.8.2 ML_GetSystemTimeUTC

Brief description This function block reads the system time of the control in UTC. The system time is determined by reading the parameter C-0-0050 .

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/SystemTime

Tab. 4-92: Library assignment

Interface description

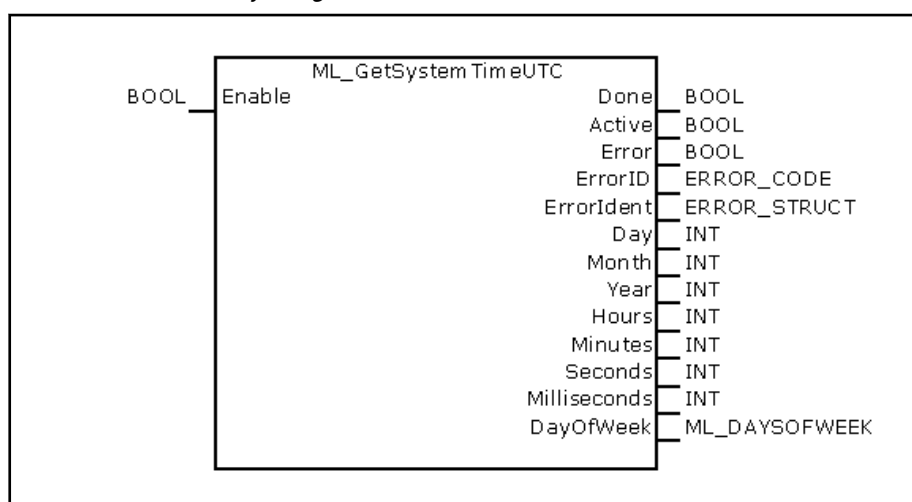


Fig. 4-39: Function block ML_GetSystemTimeUTC

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Processing enabled
VAR_OUTPUT	Done	BOOL	Operation executed
	Active	BOOL	Confirmation of processing enabled
	Error	BOOL	Indicates that an error occurred in the function block.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Day	INT	Day of the month
	Month	INT	Month of the year
	Year	INT	Year
	Hours	INT	Hours
	Minutes	INT	Minutes
	Seconds	INT	Seconds
	Milliseconds	INT	Milliseconds
	DayOfWeek	ML_DAYSOFWEEK	Week day

Tab. 4-93: ML_GetSystemTimeUTC interface

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

Tab. 4-94: Input behavior of the ML_GetSystemTimeUTC

Functional description

This function block reads the system time of the control in UTC. The system time is determined by reading the parameter C-0-0050 .

The "ML_GetSystemTimeUTC" function block transfers the system time of the control after enabling the processing of the function block via "Enable". After the function block has been successfully processed, the output is set to "Done" and the outputs "Day", "Month", "Year", "Hours", "Minutes", "Seconds", "Milliseconds" and "DayOfWeek" are valid.

If an error occurs while processing the function block, this is signaled by the "Error" output and specified by the outputs "ErrorID" and "ErrorIdent" updated in this case.

Day, Month, Year

The outputs "Day", "Month" and "Year" map the date of the system time. It is specified as integer.

Example:

Day : 22, Month: 6, Year: 2010

Hours, Minutes, Seconds

The outputs "Hours", "Minutes", "Seconds" and "Milliseconds" map the time of the system time. It is specified as integer.

Example:

Hours: 13, Minutes: 36, Seconds: 42, milliseconds: 123

DayOfWeek

The output "DayOfWeek" specifies the week day of the system time. It is specified as enumeration type. The enumeration type used is explained in [ML_DAYSOFWEEK, page 150](#).

Example:

DayOfWeek: Friday

Error codes The function block uses the error table [MLC_TABLE, 16#0030, page 360](#).

4.8.3 ML_SetSystemTimeUTC

Brief description This function block sets the system time of the control in UTC. The system time is set by writing the parameter C-0-0050 .

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/SystemTime

Tab. 4-95: Library assignment

Interface description

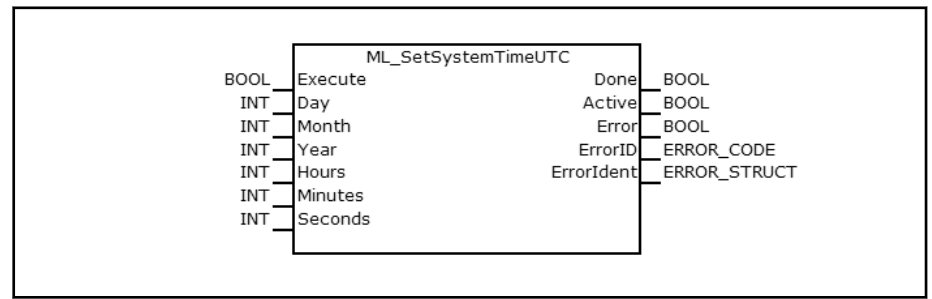


Fig. 4-40: ML_SetSystemTimeUTC function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing enabled
	Day	INT	Day of the month
	Month	INT	Month of the year
	Year	INT	Year
	Hours	INT	Hour
	Minutes	INT	Minute
	Seconds	INT	Seconds
VAR_OUTPUT	Done	BOOL	Operation executed
	Active	BOOL	Confirmation of processing enabled
	Error	BOOL	Indicates that an error occurred in the function block.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 4-96: ML_SetSystemTimeUTC interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
Day	INT	1	31	1	Rising edge at "Execute"
Month	INT	1	12	1	Rising edge at "Execute"
Year	INT	1	9999	1	Rising edge at "Execute"
Hours	INT	0	23	0	Rising edge at "Execute"
Minutes	INT	0	59	0	Rising edge at "Execute"
Seconds	INT	0	59	0	Rising edge at "Execute"

Tab. 4-97: Input behavior of the ML_SetSystemTimeUTC

Functional description

This function block sets the system time of the control in UTC. The system time is set by writing the parameter C-0-0050 .

The "ML_SetSystemTimeUTC" function block sets the system time of the control after enabling the processing of the function block via "Execute". The inputs "Day", "Month", "Year", "Hour", "Minutes" and "Seconds" are applied at a rising edge at "Execute". After a successful processing of the function block, the output is set to "Done".

If an error occurs while processing the function block, this is signaled by the "Error" output and specified by the outputs "ErrorID" and "ErrorIdent" updated in this case.

Day, Month, Year

The outputs "Day", "Month" and "Year" map the date of the system time. It is specified as integer.

Example:

Day : 22, Month: 6, Year: 2010

Hours, Minutes, Seconds

The outputs "Hours", "Minutes" and "Seconds" map the time of the system time. It is specified as integer.

Example:

Hours: 13, Minutes: 36, Seconds: 42

Error codes

The function block uses the error table [MLC_TABLE, 16#0030, page 360](#).

4.8.4 ML_DAYSOFWEEK

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/SystemTime

Tab. 4-98: Library assignment

The data type "MB_DAYSOFWEEK" is an enumeration type with the following entries:

Element	Value	Description
Monday	1	Monday
Tuesday	2	Tuesday
Wednesday	3	Wednesday
Thursday	4	Thursday
Friday	5	Friday
Saturday	6	Saturday
Sunday	7	Sunday

Tab. 4-99: Definition of the ML_DAYSOFWEEK enumeration type

4.9 Target - Target system

4.9.1 ML_GetCpuLoad

Brief description This function returns the current CPU load in percent.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-100: Library assignment

Interface description

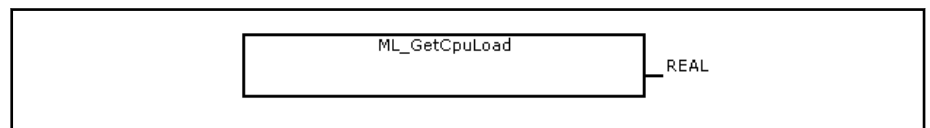


Fig. 4-41: Interface of the ML_GetCpuLoad

	Name	Type	Comment
RETURN		REAL	CPU load in percent

Tab. 4-101: Interface of the ML_GetCpuLoad

Functional description This function returns the current CPU load in percent.

The function "ML_GetCpuLoad" reads the parameter C-0-0415 and returns the contained CPU load in per cent.

4.9.2 ML_GetDiagnosisLog

Brief description The function ML_GetDiagnosisLog retrieves entries from the diagnostic log-book.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-102: Library assignment

Interface description

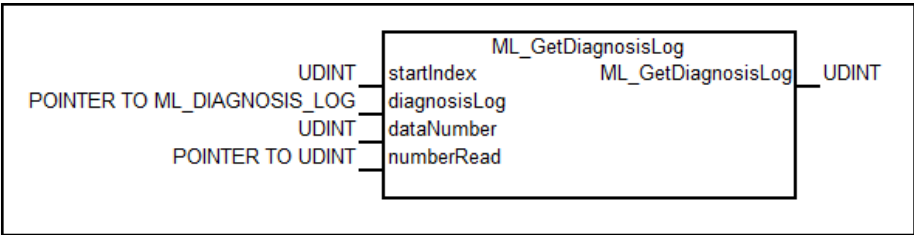


Fig. 4-42: ML_GetDiagnosisLog function

	Name	Type	Comment
VAR_INPUT	startIndex	UDINT	Start index from which the diagnostic logbook is to be stored
	diagnosisLog	POINTER TO ML_DIAGNOSIS_LOG	Address on which the entries of the diagnostic logbook is to be saved See ML_DIAGNOSIS_LOG , page 160
	dataNumber	UDINT	Size of diagnosisLog (number of entries)
	numberRead	POINTER TO UDINT	Address of the number of the read logbook entries
RETURN		UDINT	Brief note on error cause

Tab. 4-103: Interface of the FUN ML_GetDiagnosisLog

Functional description The function reads a number of diagnostic log entries starting from "startIndex".

The read number of entries is returned in "numberRead".

If less entries than "dataNumber" are available, only this number of entries is read.

4.9.3 ML_GetFirmwareString

Brief description This function returns the firmware string of the control.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-104: Library assignment

Interface description

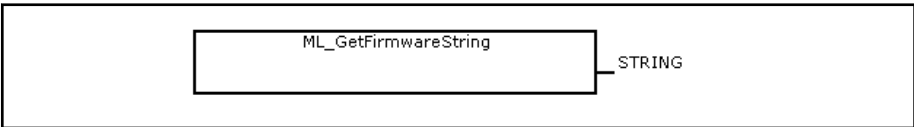


Fig. 4-43: Interface of the ML_GetFirmwareString

	Name	Type	Comment
RETURN		STRING	Firmware ID of the control.

Tab. 4-105: Interface of the ML_GetFirmwareString

Functional description This function returns the firmware string of the control.

The function "ML_GetFirmwareString" reads the parameter C-0-0012 and returns the contained firmware string as character string.

Example:

CML65s-MLC-14V04.0072

4.9.4 ML_GetHardwareInfoCtrl

Brief description This function returns information on the control hardware.



The function is not provided with any input, any return value, but four outputs (VAR_OUTPUT).

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-106: Library assignment

Interface description

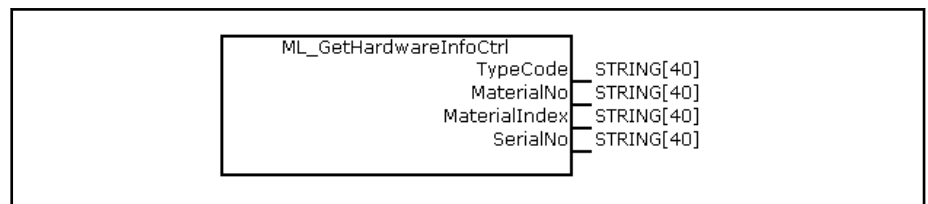


Fig. 4-44: ML_GetHardwareInfoCtrl

	Name	Type	Comment
VAR_OUTPUT	TypeCode	STRING[40]	Type code of the control hardware (C-0-0010)
	MaterialNo	STRING[40]	Parts number of the control hardware (C-0-0021)
	MaterialIndex	STRING[40]	Material index of the control hardware (C-0-0021)
	SerialNo	STRING[40]	Serial number of the control

Tab. 4-107: Interface of the ML_GetHardwareInfoCtrl

Functional description This function returns information on the control hardware.

Using the function in a program

```

VAR
  strTypeCode: STRING;
  strMaterialNo: STRING;
  strMaterialIndex: STRING;
  strSerialNo: STRING;
END_VAR
  
```

```

ML_GetHardwareInfoCtrl (TypeCode=> strTypeCode, MaterialNo=> strMaterialNo,
MaTerialIndex=> strMaterialIndex, SerialNo=> strSerialNo);
  
```

Expression	Type	Value	Prepared value	Comment
strTypeCode	STRING	'CML65.1-3P-500-NA-NNNN-NW'		
strMaterialNo	STRING	'R911170900'		
strMaterialIndex	STRING	'301'		
strSerialNo	STRING	'006277436'		


```

1 ML_GetHardwareInfoCtrl (TypeCode=> strTypeCode, MaterialNo=> strMaterialNo,
2 MaterialIndex=> strMaterialIndex, SerialNo=> strSerialNo); RETURN
  
```

Fig. 4-45: Response of the ML_GetHardwareInfoCtrl function

4.9.5 ML_GetInfoApplication

Brief description This function blocks read the PLC application available on the control and reads and stores the status in the "Info" output.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-108: Library assignment

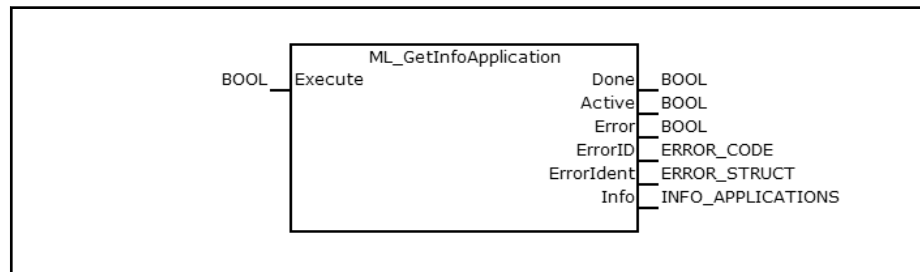
Interface description

Fig. 4-46: ML_GetInfoApplication function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Processing enabled
VAR_OUTPUT	Done	BOOL	Operation executed
	Active	BOOL	Confirmation of processing enabled
	Error	BOOL	Indicates that an error occurred in the function block.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Info	INFO_APPLICATIONS	Application information INFO_APPLICATIONS on page 160

Tab. 4-109: ML_GetInfoApplication interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous

Tab. 4-110: Input behavior of the ML_GetInfoApplication

Functional description

This function blocks read the PLC application available on the control and reads and stores the status in the "Info" output.

The function block "ML_GetInfoApplication" transfers the applications existing on the control and thus their statuses via "Execute" after the processing of the function block has been enabled. After a successful processing of the function block, the output is set to "Done" and the "Info" output is valid.

If an error occurs while processing the function block, this is signaled by the "Error" output and specified by the outputs "ErrorID" and "ErrorIdent" updated in this case.

IpAddress

The output "Info" outputs the structure [INFO_APPLICATIONS on page 160](#) containing the number of applications, their names and statuses.

Error codes

The function block uses the error table [MLC_TABLE, 16#0030, page 360](#).

4.9.6 ML_GetMaxTemperature

Brief description

This function returns the highest temperature reached of the control.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-111: Library assignment

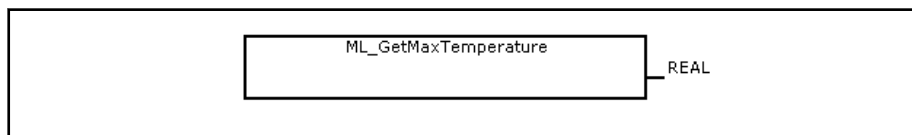


Fig. 4-47: ML_GetMaxTemperature interface

Interface description

	Name	Type	Comment
RETURN		REAL	Maximum temperature of the control

Tab. 4-112: ML_GetMaxTemperature interface

Functional description

This function returns the maximum temperature of the control.

The function "ML_GetMaxTemperature" reads the parameter C-0-0071 and returns the contained maximum temperature of the control.

The temperature unit can be read out with the function [ML_GetScalingType](#) on page 156.

The maximum temperature is reset when switching on the control.

4.9.7 ML_GetNewestDiagnosisIndex

The ML_GetNewestDiagnosisIndex function returns the latest index of the diagnostic logbook.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-113: Library assignment

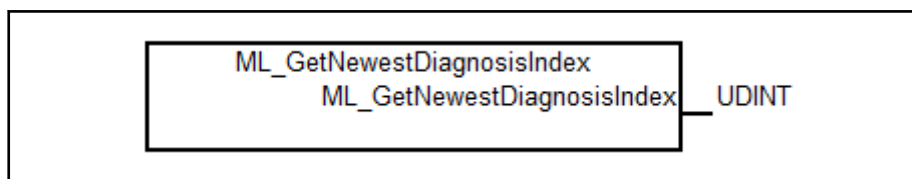


Fig. 4-48: ML_GetNewestDiagnosisIndex function

	Name	Type	Comment
RETURN		UDINT	Latest diagnostic logbook index

Tab. 4-114: Interface of the FUN ML_GetNewestDiagnosisIndex

4.9.8 ML_GetOhcFan

Brief description

This function returns the operating hours of the fan of the control.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-115: Library assignment

Interface description

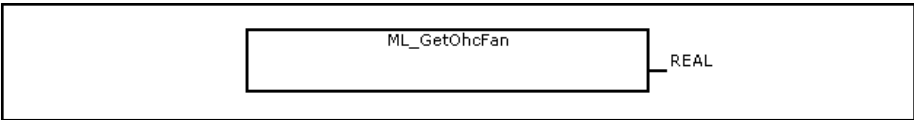


Fig. 4-49: Interface of the ML_GetOhcFan

	Name	Type	Comment
RETURN		REAL	Operating hours of the fan of the control

Tab. 4-116: Interface of the ML_GetOhcFan

Functional description

This function returns the operating hours of the fan of the control.

The function "ML_GetOhcFan" reads the parameter C-0-0052 and returns the contained operating hours of the control fan.

4.9.9 ML_GetScalingType

Brief description

This function returns the unit of all temperature specifications of the control.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-117: Library assignment

Interface description

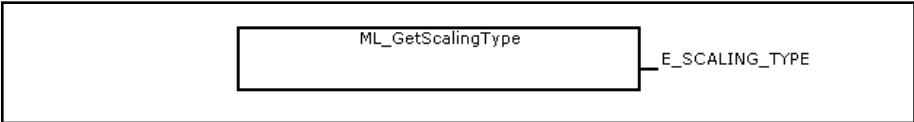


Fig. 4-50: ML_GetScalingType interface

	Name	Type	Comment
RETURN		E_SCALING_TYPE	Unit of all temperature specification of the control

Tab. 4-118: ML_GetScalingType interface

Functional description

This function returns the unit of all temperature specifications of the control.

The function "ML_GetScalingType" reads the parameter C-0-0063 and returns the contained temperature unit of the control.

The return value of the function is of the enumeration type [E_SCALING_TYPE](#) on page 159.

4.9.10 ML_GetSystemSpecialPath

Brief description

This function returns the platform-specific absolute path matching the symbolic path (PathIdentifier).

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-119: Library assignment

Interface description

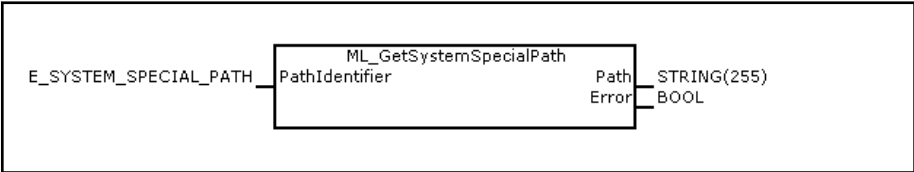


Fig. 4-51: ML_GetSystemSpecialPath interface

	Name	Type	Comment
VAR_INPUT	PathIdentifier	E_SYSTEM_SPECIAL_PATH	Symbolic path name E_SYSTEM_SPECIAL_PATH , page 159
VAR_OUTPUT	Path	STRING(255)	Platform-specific absolute path. It start and ends with the path delimiter "/"
	Error	BOOL	Error message

Tab. 4-120: *ML_GetSystemSpecialPath interface*

Functional description The function returns the platform-specific absolute path matching the symbolic path (PathIdentifier).

It is for example possible to request the absolute path of the system path from the connected device. This information can be used for an ftp file access.

The function is required, as the system path is not always on the same absolute path.

4.9.11 ML_GetTargetInfo

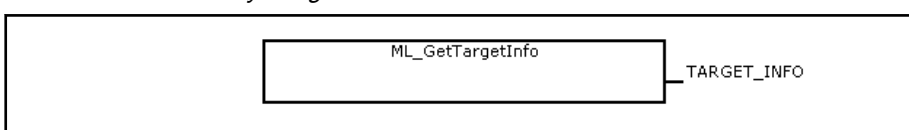
Brief description This function returns information on the target system.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-121: *Library assignment*

Interface description

Fig. 4-52: *ML_GetTargetInfo interface*

	Name	Type	Comment
RETURN		TARGET_INFO	Information on the target system TARGET_INFO on page 161

Tab. 4-122: *ML_GetTargetInfo interface*

4.9.12 ML_GetTemperature

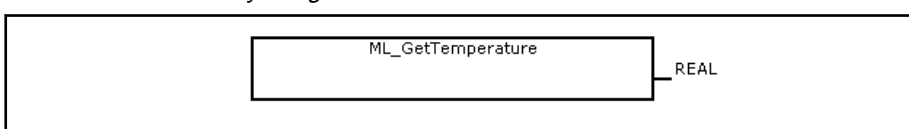
Brief description This function returns the current temperature of the control.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-123: *Library assignment*

Interface description

Fig. 4-53: *ML_GetTemperature interface*

	Name	Type	Comment
RETURN		REAL	Current temperature of the control

Tab. 4-124: *ML_GetTemperature interface***Functional description**

This function returns the current temperature of the control.

The function "ML_GetTemperature" reads the parameter C-0-0070 and returns the contained current temperature of the control.

The temperature unit can be read out with the function [ML_GetScalingType](#) on page 156.

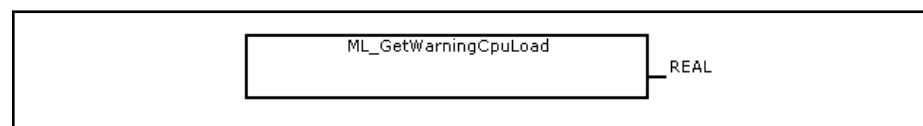
4.9.13 ML_GetWarningCpuLoad

Brief description

This function returns the maximum CPU load set in percent.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-125: *Library assignment***Interface description**Fig. 4-54: *Interface of the ML_GetWarningCpuLoad*

	Name	Type	Comment
RETURN		REAL	Maximum CPU load set in percent

Tab. 4-126: *Interface of the ML_GetWarningCpuLoad***Functional description**

This function returns the maximum CPU load set in percent.

The function "ML_GetWarningCpuLoad" reads the parameter C-0-0418 and returns the maximum CPU load set in percent.

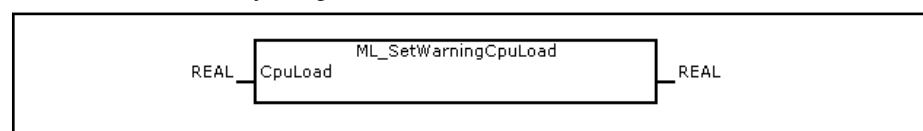
4.9.14 ML_SetWarningCpuLoad

Brief description

This function sets the maximum CPU load allowed in percent.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-127: *Library assignment***Interface description**Fig. 4-55: *Interface of the ML_SetWarningCpuLoad*

	Name	Type	Comment
VAR_INPUT	CpuLoad	REAL	Maximum CPU load allowed in percent
RETURN		REAL	

Tab. 4-128: *Interface of the ML_SetWarningCpuLoad***Functional description**

This function sets the maximum CPU load allowed in percent.

The function "ML_SetWarningCpuLoad" writes the values at the "CpuLoad" input to the parameter C-0-0418.

4.9.15 APP_INFO

The data type "APP_INFO" is used in the structure [INFO_APPLICATIONS](#) on page 160.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-129: Library assignment

The data type "APP_INFO" is a structure with the following entries:

Element	Type	Description
Name	STRING[255]	Name of the PLC application
State	STRING	State of the PLC application

Tab. 4-130: Definition of the APP_INFO structure

4.9.16 E_SCALING_TYPE

The data type "E_SCALING_TYPE" is used in the structure [ML_GetScalingType](#) on page 156.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-131: Library assignment

The data type "E_SCALING_TYPE" is an enumeration type with the following entries:

Element	Value	Description
CELSIUS	16#0	All temperatures are given in °C
FARENHEIT	16#1	All temperatures are given in F

Tab. 4-132: Definition of the enumeration type E_SCALING_TYPE

4.9.17 E_SYSTEM_SPECIAL_PATH

The data type "E_SYSTEM_SPECIAL_PATH" is used in the structure [ML_GetSystemSpecialPath](#) on page 156.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-133: Library assignment

The data type "E_SYSTEM_SPECIAL_PATH" is an enumeration type with the following entries:

Element	Data type	Value	Description
PATH_SYSTEM	DINT	16#0	System path, root
PATH_OEM	DINT	16#1	OEM path, root

Element	Data type	Value	Description
PATH_USER	DINT	16#2	User path, root
PATH_PROJECTDATAPROTECTED	DINT	16#3	Protected project data path
PATH_PROJECTDATA	DINT	16#4	Project data path
PATH_PROJECTDATA_BUNDLES	DINT	16#5	User bundle project data path
PATH_PROJECTDATA_PLC	DINT	16#6	PLC project data path
PATH_PROJECTDATA_TMP	DINT	16#7	Temporary project data path

Tab. 4-134: Definition of the enumeration type "E_SYSTEM_SPECIAL_PATH"

4.9.18 INFO_APPLICATIONS

The "INFO_APPLICATIONS" data type is used in the [ML_GetInfoApplication](#) on page 153 function block.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-135: Library assignment

The data type "INFO_APPLICATIONS" is a structure with the following entries:

Element	Type	Description
NoOfApplications	DINT	Number of PLC applications on the control
AppInfo	ARRAY[1..] OF APP_INFO	Information on the individual PLC applications APP_INFO on page 159

Tab. 4-136: Definition of the INFO_APPLICATIONS structure

4.9.19 ML_DIAGNOSIS_LOG

Brief description

This structure contains an entry of the diagnostic logbook.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-137: Library assignment

Structure element	Type	Description
number	UDINT	Error number
state	UDINT	Status of the diagnostic entry
dateTime	SYS_TIME_DATE	Time of the diagnostics
logicalAddress	DINT	Logical device address (e.g. axis, kinematics)
physicalAddress	DINT	Physical device address (e.g. device)
despatcher	UDINT	Diagnostic source

Structure element	Type	Description
index	UDINT	Diagnostic index in the diagnostic logbook
text	STRING[123]	Diagnostic text

Tab. 4-138: ML_DIAGNOSIS_LOG data structure

4.9.20 TARGET_INFO

The data type "TARGET_INFO" is used in the function [ML_GetTargetInfo](#), page 157,.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base/target

Tab. 4-139: Library assignment

The data type "TARGET_INFO" is a structure with the following entries:

Element	Type	Description
TargetType	DWORD	Target system type
TargetId	DWORD	Target system ID
TargetVersion	DWORD	Target system version
DeviceName	STRING	Control name
NodeName	STRING	Node name

Tab. 4-140: Definition of the TARGET_INFO structure

4.10 Transmission

The functions to transmit character strings (URI: Uniform Resource Identifier) are described in the following.

4.10.1 ML_SystemGetTransmissionInfo

Brief description

This function provides information on a transmission

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base / Transmission

Tab. 4-141: Library assignment

Interface description

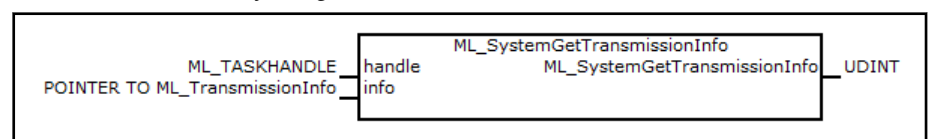


Fig. 4-56: ML_SystemGetTransmissionInfo interface

	Name	Type	Comment
VAR_INPUT	handle	POINTER TO ML_TASKHANDLE	Pointer whereto ML_TASKHANDLE is to be saved ML_TASKHANDLE, page 124
	info	POINTER TO ML_TransmissionInfo	Detailed information on the transmission ML_TransmissionInfo, page 164
Return value		UDINT	Brief note on error cause

Tab. 4-142: ML_SystemGetTransmissionInfo interface

4.10.2 ML_SystemGetTransmissionStatus

Brief description This function provides the status on a transmission

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base / Transmission

Tab. 4-143: Library assignment

Interface description

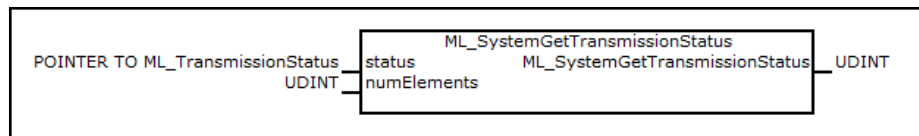


Fig. 4-57: ML_SystemGetTransmissionStatus interface

	Name	Type	Comment
VAR_INPUT	status	POINTER TO ML_Transmission-Status	Pointer whereto ML_TransmissionStatus is to be saved ML_TransmissionStatus, page 165
	numElements	UDINT	Number of elements in status
Return value		UDINT	Brief note on error cause

Tab. 4-144: ML_SystemGetTransmissionStatus interface

4.10.3 ML_SystemStartTransmission

Brief description This function starts a transmission between the control and another device.

Source and target for the transmission have to be saved in URI format (for the current state, refer to [RFC 3986](#) (Request for Comments 3986) in the internet).

The schemes "file" and "tftp" are supported. For the URI format of the "tftp" scheme, refer to [RFC 3617](#) (Request for Comments 3617).

Examples of a valid URI:

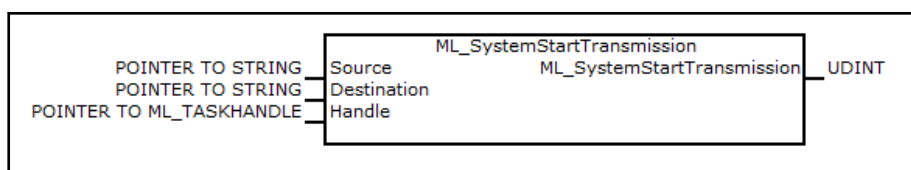
- file:///ata0b/localfile.txt
- tftp://172.31.254.1/remotefile.txt
- tftp://172.31.254.1:69/remotefile.txt

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base / Transmission

Tab. 4-145: Library assignment

Interface description

Fig. 4-58: *ML_SystemStartTransmission interface*

	Name	Type	Comment
VAR_INPUT	Source	POINTER TO STRING	Transmission source
	Destination	POINTER TO STRING	Transmission target
	Handle	POINTER TO ML_TASKHANDLE	Address in which the handle for the transmission is to be saved ML_TASKHANDLE , page 124
Return value		UDINT	Brief note on error cause

Tab. 4-146: *ML_SystemStartTransmission interface*

4.10.4 ML_SystemTftpGetRetransmitCount

Brief description

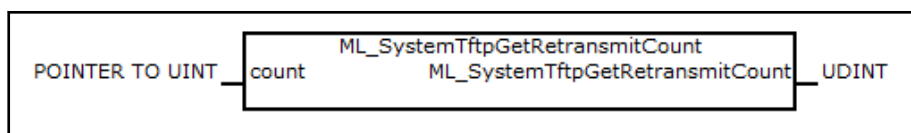
This function determines the number of permitted new TFTP transmissions for future transmissions.

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base / Transmission

Tab. 4-147: *Library assignment*

Interface description

Fig. 4-59: *ML_SystemTftpGetRetransmitCount interface*

	Name	Type	Comment
VAR_INPUT	count	POINTER TO UINT	Address in which the number of permitted new transmissions is to be saved.
Return value		UDINT	Brief note on error cause

Tab. 4-148: *ML_SystemTftpGetRetransmitCount interface*

4.10.5 ML_SystemTftpGetTimeout

Brief description

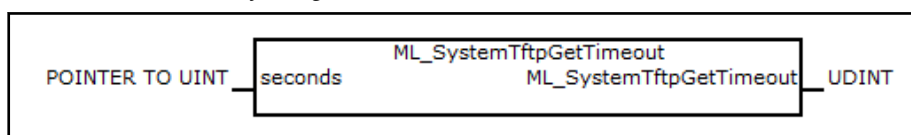
This function determines the TFTP timeout for future transmissions.

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base / Transmission

Tab. 4-149: *Library assignment*

Interface description

Fig. 4-60: *ML_SystemTftpGetTimeout interface*

	Name	Type	Comment
VAR_INPUT	seconds	POINTER TO UINT	Address in which the TFTP timeout for future transmissions is to be saved.
Return value		UDINT	Brief note on error cause

Tab. 4-150: ML_SystemTftpGetTimeout interface

4.10.6 ML_SystemTftpSetRetransmitCount

Brief description This function sets the number of permitted new TFTP transmissions for future transmissions.

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base / Transmission

Tab. 4-151: Library assignment

Interface description

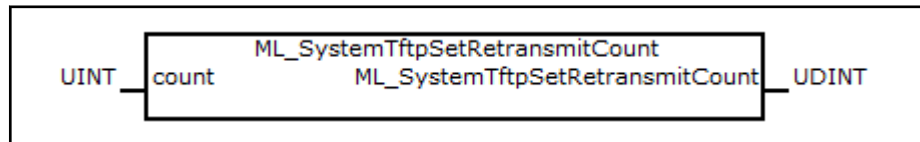


Fig. 4-61: ML_SystemTftpSetRetransmitCount interface

	Name	Type	Comment
VAR_INPUT	count	UINT	Number of permitted new transmissions.
Return value		UDINT	Brief note on error cause

Tab. 4-152: ML_SystemTftpSetRetransmitCount interface

4.10.7 ML_SystemTftpSetTimeout

Brief description This function determines the TFTP timeout for future transmissions.

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base / Transmission

Tab. 4-153: Library assignment

Interface description

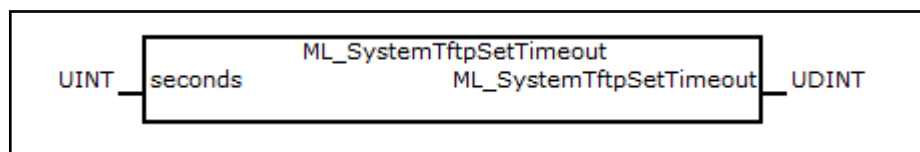


Fig. 4-62: ML_SystemTftpSetTimeout interface

	Name	Type	Comment
VAR_INPUT	seconds	UINT	TFTP timeout for future transmissions
Return value		UDINT	Brief note on error cause

Tab. 4-154: ML_SystemTftpSetTimeout interface

4.10.8 ML_TransmissionInfo

Brief description This structure defines detailed information on a transmission.

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base / Transmission

Tab. 4-155: Library assignment

Structure element	Type	Comment
status	ML_TransmissionStatus	Transmission status ML_TransmissionStatus, page 165
Source	STRING(256)	Transmission source
Destination	STRING(256)	Transmission target
ErrorMessage	STRING(128)	Error message of the transmission

Tab. 4-156: Data structure of the ML_TransmissionInfo

4.10.9 ML_TransmissionStatus

Brief description

This structure defines the status of a transmission.

Assignment target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS PLC basic package	ML_Base / Transmission

Tab. 4-157: Library assignment

Structure element	Type	Comment
Handle	ML_TASKHANDLE	ML_TASKHANDLE of the transmission ML_TASKHANDLE, page 124
Result	UDINT	Status code of the transmission
State	ML_ProcessState	Transmission state ML_ProcessState, page 125
TotalBytes	UDINT	Total number of bytes of the transmission
TransmittedBytes	UDINT	Current number of bytes of the transmission

Tab. 4-158: Data structure of the ML_TransmissionStatus

5 Function blocks according to PLCopen specification

The libraries "ML_PLCopen" and "RMB_PLCopenFieldbus" implement the PLCopen specification for the two field buses sercos and Profibus.

The library [ML_PLCopen.library, page 171](#), includes the function blocks for the sercos II and sercos III interface.

Apart from the Rexroth IndraDrives, all other sercos-capable drives are supported.

The library [RMB_PLCopenFieldbus.library, page 361](#), includes function blocks and data types to control Rexroth IndraDrives via Profibus.

Supported function blocks

The following table compares the implemented function blocks of the two libraries.

Function block	ML_PLCopen	RMB_PLCopenFieldbus
MC_Power	X	X
MC_Reset	X	X
MC_Stop	X	X
MB_Stop	-	X
MB_Home	X	X
MC_MoveAbsolute	X	X
MC_MoveAdditive	X	X
MC_MoveRelative	X	X
MC_MoveVelocity	X	X
MB_SetPositionControlMode	X	-
MB_ChangeCamData	X	-
MB_ChangeFlexEventSet	X	-
MB_ChangeFlexProfileSet	X	-
MB_ChangeProfileSet	X	-
MB_ChangeProfileStep	X	-
MB_GearInPos	X	-
MB_MotionProfile	X	-
MB_Phasing	X	-
MB_PhasingSlave	X	-
MC_CamIn	X	-
MC_CamOut	X	-
MC_GearIn	X	-
MC_GearOut	X	-
MC_TorqueControl	X	-
ML_FlexProfile	X	-
ML_OpenCyclicAnalogChannel	X	-

Function blocks according to PLCopen specification

Function block	ML_PLCOpen	RMB_PLCOpenFieldbus
ML_OpenCyclicPositionChannel	X	-
ML_OpenCyclicTorqueChannel	X	-
ML_OpenCyclicVelocityChannel	X	-
ML_WriteCyclicAnalog	X	-
ML_WriteCyclicPosition	X	-
ML_WriteCyclicTorque	X	-
ML_WriteCyclicVelocity	X	-
ML_ControlOff	X	-
ML_ControlOn	X	-
ML_ControlWriteActualData	X	-
ML_OpenCyclicAnalogChannelCtrl	X	-
ML_OpenCyclicPositionChannelCtrl	X	-
ML_OpenCyclicTorqueChannelCtrl	X	-
ML_OpenCyclicVelocityChannelCtrl	X	-
ML_WriteCyclicAnalogCtrl	X	-
ML_WriteCyclicPositionCtrl	X	-
ML_WriteCyclicTorqueCtrl	X	-
ML_WriteCyclicVelocityCtrl	X	-
MB_Command	X	-
MB_ReadLintParameter	X	-
MB_ReadListParameter	X	X
MB_ReadListSegParameterPtr	-	X
MB_ReadLrealListParameter	X	-
MB_ReadLrealParameter	X	-
MB_ReadMaxRealValue	X	X
MB_ReadMaxValue	X	X
MB_ReadMinRealValue	X	X
MB_ReadMinValue	X	X
MB_ReadName	X	-
MB_ReadParameter	X	X
MB_ReadRealParameter	X	X
MB_ReadSercosAttribute	X	X
MB_ReadSercosDataStatus	X	-
MB_ReadStringParameter	X	X

Function block	ML_PLCOpen	RMB_PLCOpenFieldbus
MB_ReadUnit	X	-
MB_WriteIntParameter	X	-
MB_WriteListParameter	X	X
MB_WriteListSegParameterPtr	-	X
MB_WriteLrealParameter	X	-
MB_WriteParameter	X	X
MB_WriteRealParameter	X	X
MB_WriteStringParameter	X	X
MC_ReadActualPosition	X	X
MC_ReadAxisError	X	X
MC_ReadStatus	X	X
ML_AbortTrigger	X	-
ML_InitTouchProbe	X	-
ML_TouchProbe	X	-
ML_TouchProbeContinuous	X	-
ML_WriteExpectWindow	X	-
MP_Drive	-	X
MP_DriveParameter	-	X
MP_FB_IndraDrv_AcyclicCommunication	-	X

Tab. 5-1: Supported function blocks in the ML_PLCOpen.library/RMB_PLCOpenFieldBus.library

6 ML_PLCopen.library

6.1 Introduction and overview



The basic function package PLC is sufficient for the library of the IndraLogic XLC/IndraMotion MLC 14VRS system.

EXCEPTIONS:

The MOT (Motion) function package is required for the function blocks from the fields "Controlled Motion", "TouchProbe" and "Motion"

The ML_PLCopen library contains essential function blocks for the implementation of IndraLogic XLC/IndraMotion MLC systems 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:

- [RIL_CommonTypes.library, page 25](#), data types for general use in all IndraWorks systems,
- [ML_Base.library, page 71](#), data types of the IndraLogic XLC/IndraMotion MLC 14VRS system for Sercos-based axes.

Library structure:

Library information and access functions, these library components are automatically created for the respective library and do not contain specific parts.

ML_PLCopen function blocks, functions and data types

Area of application (folder in library)	
POUs	Description
Global variables (Globale_Variablen)	
ProbeSelect, page 175	defines the field containing data of the touch probes 1..100.
Data types/data types of TouchProbe	
PROBECONFIG, page 175	defines the data type for data of the touch probes 1..100
POUs (Controlled Motion) - MOT (Motion) function package of the control	
ML_ControlOff, page 268	Completes the cyclic axis control of the PLC
ML_ControlOn, page 267	Provides the complete control for the cyclic command value specification to the PLC
ML_ControlWriteActualData, page 269	Writes the current values of a PLC controller to the controller axis of the MLC
ML_OpenCyclicAnalogChannelCtrl, page 270	Opens the cyclic analog command value channel for the specified axis.
ML_OpenCyclicPositionChannelCtrl, page 272	Opens the cyclic position command value channel for the specified axis
ML_OpenCyclicTorqueChannelCtrl, page 274	Opens the cyclic force/torque command value channel for the specified axis
ML_OpenCyclicVelocityChannelCtrl, page 276	Opens the cyclic velocity command value channel for the specified axis
ML_WriteCyclicAnalogCtrl, page 271	Writes a cyclic analog value for an axis if the axis is in the "Controlled Motion" state

ML_WriteCyclicPositionCtrl, page 273	Writes a cyclic position value for an axis if the axis is in the "Controlled Motion" state
ML_WriteCyclicTorqueCtrl, page 275	Writes a cyclic torque/force value for an axis if the axis is in the "Controlled Motion" state
ML_WriteCyclicVelocityCtrl, page 277	Writes a cyclic velocity value for an axis if the axis is in the "Controlled Motion" state
POUs (Motion) - MOT (Motion) function package of the control	
MB_ChangeCamData , page 225	Supports loading of cam table data under real-time conditions
MB_ChangeFlexEventSet , page 242	Used to configure events
MB_ChangeFlexProfileSet , page 239	Allows to write on a complete MotionProfile (FlexProfile) with up to 64 steps or to exchange an existing one
MB_ChangeProfileSet , page 218	Allows to write on a complete MotionProfile with up to 16 steps or to exchange an existing one
MB_ChangeProfileStep , page 222	Changes the four parameters which define one step of a MotionProfile set
MB_GearInPos, page 248	Implements an electronic gear function with fine adjustment for a phase- and velocity-synchronous (drift-free) run
MB_Home, page 199	Controls axes with incremental encoder when executing the "search home" sequence.
MB_MotionProfile, page 212	Implements an electronic cam function
MB_Phasing, page 203	Changes the offset to the master axis present as input at a synchronization function block (MC_CamIn, MB_GearInPos, MC_GearIn, MB_MotionProfile or ML_FlexProfile)
MB_PhasingSlave, page 205	Implements an adjustment of the slave axis with specified velocity and acceleration
MB_SetPositionControlMode, page 198	Sets the position controller
MC_AddAxisToGroup, page 251	Adds an axis to the axis group
MC_CamIn, page 207	Implements an electronic cam function
MC_CamOut, page 211	Completes the electronic cam function enabled by the MC_CamIn function block
MC_GearIn, page 243	Implements an electronic gear function with fine adjustment
MC_GearOut, page 247	Completes the electronic gear function with fine adjustment activated by either the function block MC_GearIn or MB_GearInPos or completes the profiles activated by either the function block MB_MotionProfile or ML_FlexProfile
MC_MoveAbsolute, page 183,	Copies the absolutely specified "position" to the "target position" and approaches it
MC_MoveAdditive, page 186	Adds the "distance" to the "target position" and approaches it
MC_MoveRelative, page 188	Determines the new "target position" by adding the "distance" and the "actual position" and approaches it
MC_MoveVelocity, page 192	Controls an "endless" motion at the respectively defined velocity
MC_Power, page 179	Controls power adding (on or off) of a real axis
MC_RemoveAxisFromGroup, page 252	Removes an axis from the axis group

MC_Reset, page 180	implements the transition from the "ErrorStop" state to the "StandStill" state by resetting all axis- and control-internal errors.
MC_Stop, page 181	implements a controlled motion stop and transfers the axis to the "Stopping" state
MC_TorqueControl, page 195	Controls a "continuously determined force with a defined force ramp in a continuous motion"
ML_FlexProfile, page 227	Implements an electronic cam table (FlexProfile)
ML_OpenCyclicAnalogChannel, page 255	Switches the axis to the mode "Cyclic Analog Command" and opens the cyclic analog command value channel
ML_OpenCyclicPositionChannel, page 257	Switches the axis to the mode "Cyclic Position Command" and opens the cyclic position command value channel
ML_OpenCyclicTorqueChannel, page 261	Switches the axis to the mode "Cyclic Torque Command" and opens the cyclic force command value channel
ML_OpenCyclicVelocityChannel, page 264	Switches the axis to the mode "Cyclic Velocity Command" and opens the cyclic velocity command value channel
ML_WriteCyclicAnalog, page 256	Writes an analog value for an axis. If the cyclic analog command value channel is open, the analog value is applied to the axis in the next cycle
ML_WriteCyclicPosition, page 259	Writes a position command for an axis. If the cyclic position command value channel is open, the position command is applied to the axis in the next cycle
ML_WriteCyclicTorque, page 263	Writes a force value for an axis. If the cyclic force command value channel is open, the force value is applied to the axis in the next cycle
ML_WriteCyclicVelocity, page 265	Writes a velocity value for an axis. If the cyclic velocity command value channel is open, the velocity value is applied to the axis in the next cycle
POUs (parameter)	
MB_Command, page 281	Allows to execute a command
MB_ReadLintParameter, page 286	Returns the value of a user-specific parameter (LINT number)
MB_ReadListParameter, page 288	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_ReadLrealListParameter, page 291	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_ReadLrealParameter, page 294	Returns the value of a user-specific parameter (LREAL number)
MB_ReadMaxRealValue, page 295	Returns the maximum value allowed of a parameter specified by the user (REAL)
MB_ReadMaxValue, page 297	Returns the maximum value allowed of a parameter specified by the user (DINT)
MB_ReadMinRealValue, page 299	Returns the minimum value allowed of a parameter specified by the user (REAL)
MB_ReadMinValue, page 301	Returns the minimum value allowed of a parameter specified by the user (DINT)
MB_ReadName, page 302	Returns the parameter name specified by the user
MB_ReadParameter, page 304	Returns the value of a specified parameter (DINT)

MB_ReadRealParameter, page 306	Returns the value of a specified parameter (REAL)
MB_ReadSercosAttribute, page 308	Allows reading the Sercos attributes of a parameter
MB_ReadSercosDataStatus, page 309	Returns the status of a parameter specified by the user
MB_ReadStringParameter, page 311	Returns the value of a user-specific parameter (STRING, character string)
MB_ReadStringParameterExt, page 313	Returns the value of a user-specific parameter (STRING, POINTER TO BYTE)
MB_ReadUnit, page 315	Returns the measuring unit of a parameter specified by the user
MB_WriteLintParameter, page 316	Writes a value to a parameter specified by the user (LINT)
MB_WriteListParameter, page 318	Writes memory content of defined length - starting from the initial address - to a specified parameter
MB_WriteRealParameter, page 321	Writes a value to a parameter specified by the user (LREAL)
MB_WriteParameter, page 322	Writes a value to a parameter specified by the user (DINT)
MB_WriteRealParameter, page 324	Writes a value to a parameter specified by the user (REAL)
MB_WriteStringParameter, page 326	Writes a character string with up to 255 characters to a user-specific parameter (STRING, character string)
MB_WriteStringParameterExt, page 327	Writes a character string of undefined length to a specified parameter (STRING, POINTER TO BYTE)
MC_ReadActualPosition, page 329	Reads the current position of the drive
MC_ReadAxisError, page 330	Retrieves the error state of the addressed axis
MC_ReadStatus, page 332	Reads the current axis state
POUs (Touch Probe) - MOT (Motion) function package of the control	
ML_AbortTrigger, page 334	Completes a single shot measurement initiated by ML_TouchProbe or a continuous measurement initiated by ML_TouchProbeContinuous
ML_InitTouchProbe, page 336	Initializes and configures the touch probe function
ML_TouchProbe, page 344	Activates the one-time position and time measurement and provides an automated time difference measurement
ML_TouchProbeContinuous, page 347	Activates the continuous position and time measurement and provides an automated time difference measurement
ML_WriteExpectWindow, page 349	Configures the expectation windows without a complete initialization run with the ML_InitTouchProbe being necessary

Tab. 6-1: *ML_TechMotion.lib*

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 (example: MC_GearIn).

If the respective function block name starts with "MB_", it means that there are comparable function blocks for different systems such as XLC, MLC and MLD (example: MB_GearInPos).

If the respective function block name starts with "ML_", it means that it has been developed for the IndraMotion MLC system to support it best possible (example: ML_FlexProfile).



As migrating from ML_ to MB_/MC_ is a long term goal for the customer, this migration is desired.

_PhaseOut



The "MC_AddAxisToGroup" and "MC_RemoveAxisFromGroup" function blocks have been moved from ML_Robot to ML_PLCopen, to make them available for IndraLogic XLC.



The function blocks "IL_SercosAttribute", "IL_SercosElementsToParaNo" and "IL_SercosStringToParaNo" are part of the ML_Base_xx_yy.libraries in 12VRS and higher.



The function blocks and data types of the motion control "Coordinated Motion" are not in this library anymore.
They were summarized in the ML_Robot.library.

6.2 Global_Variables

6.2.1 ProbeSelect

Brief description The global variable list **ProbeSelect** contains the data which is to be assigned to the touch probes 1..100.

ProbeSelect uses the data type [PROBECONFIG](#), page 175.

Library	Range
ML_PLCopen	Global_Variables

Tab. 6-2: Library assignment

Declaring the global variable list *ProbeSelect*

```
VAR_GLOBAL
_ProbeSelect:PROBECONFIG;
END_VAR
```

6.3 Data types/data types of TouchProbe

6.3.1 PROBECONFIG

Brief description The data type defines a field to record 100 touch probe data records.
The MB_PROBE_CONFIG data records are contained in the RIL_Common-Types library.

Library	Range
ML_PLCopen	Data types/data types of TouchProbe

Tab. 6-3: Library assignment

Type declaration *PROBECONFIG*

```
TYPE PROBECONFIG :
STRUCT
  arProbeConfig: ARRAY [1..100] OF MB_PROBE_CONFIG;
```

END_STRUCT
END_TYPE

6.4 State diagram of a real axis

The state diagram below defines the behavior of IndraMotion axes from the user's point of view if several IndraMotion function blocks are activated at the same time. This combination of MotionProfiles is useful to generate complicated profiles and to respond to exceptional situations in programs.

In principle, motion commands are given and processed sequentially, even if the PLC is capable of parallel processing. These motion commands follow the state diagram of the IndraMotion axes.

Each axis is in exactly one of the states defined (see the state diagram below). Each motion command represents a state transition for the axis concerned. The sequence of these transitions describes the entire behavior of the axis.

The possible status is displayed in parameter "A-0-0022, Extended axis status".

The diagram describes the behavior of a single axis.

From the user's point of view with regard to multi-axis function blocks such as MB_MotionProfile, MB_FlexProfile, MC_CamIn, MC_GearIn and MB_GearIn-Pos, each axis is in its special state in the state diagram. The cam master axis can be in the "Continuous Motion" state while the corresponding slave axis is in the "Synchronized Motion" state.

The attachment of a slave axis to a master axis does not affect the behavior of the master axis.

Function blocks with "administrative character", such as function blocks for writing and reading parameters, do not have any influence on the state diagram.

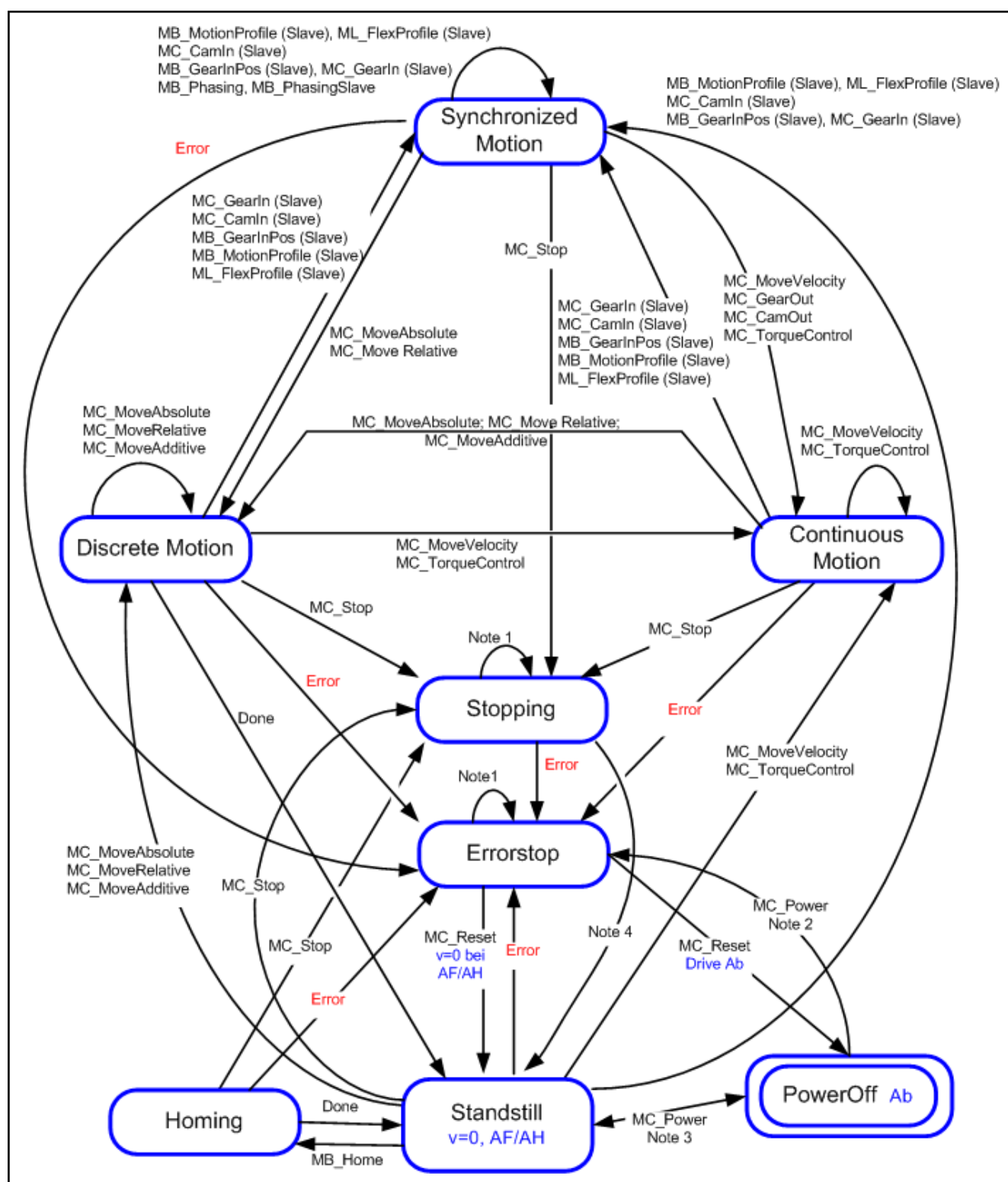


Fig. 6-1: IndraMotion state diagram of a real axis

Legend

(Note 1) - All function blocks can be called without being processed except for MC_Reset and Error. They create the respective transitions to StandStill or ErrorStop.

(Note 2) - Power.Enable = TRUE and error in the axis.

(Note 3) - Power.Enable = TRUE and error-free axis.

(Note 4) - MC_Stop.Done AND NOT MC_Stop.Execute

An error in the drive results in a power switch off after a certain period, depending on the programmed error reaction. However, the state machine remains in "ErrorStop" and when the power is switched off, "MC_Reset" goes into power-off state.

The state diagram is supplemented by

1. The state [Controlled Motion](#), page 266, for controlled motions.

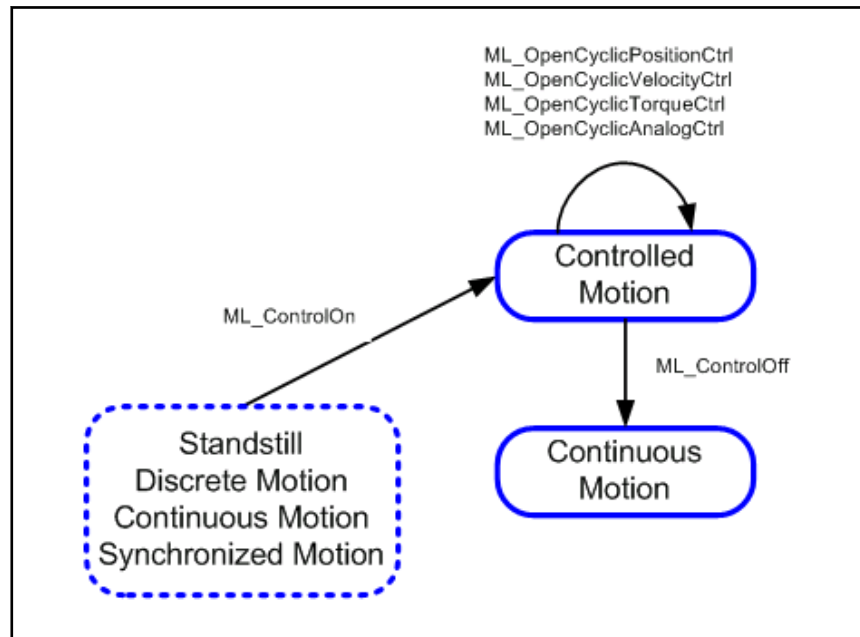


Fig. 6-2: Transition into the "Controlled Motion" status

The "Controlled Motion" status is used to assign an axis to a controller in the PLC and to prevent control by other Motion commands during this time period. The status is started with function block ML_ControlOn and ends with ML_ControlOff. In this state, only cyclical channels can be opened with the function block ML_OpenCyclic*Ctrl.

2. The state diagrams "Coordinated Motion", for axis groups (K-0-0022, Extended kinematic status).

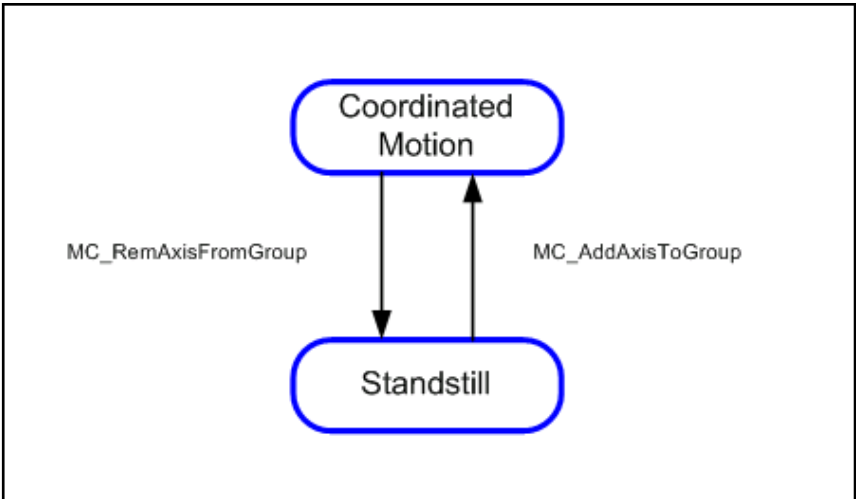


Fig. 6-3: Transition into "Coordinated Motion" state

The "Coordinated Motion" state can be started with the function block MC_AddAxisToGroup. The axis is then assigned to a kinematics. Controlling the axis with other Motion function blocks is prohibited. The assignment of an axis to a kinematics can be canceled with the function block MC_RemoveAxisFromGroup. Controlling by other Motion function blocks is now allowed again.

6.5 Function blocks for single-axis control

6.5.1 MC_Power

Brief description The function block controls the power adding (on or off).
For **each real axis, a separate instance** of this function block is required.
Enabling the function block is a prerequisite for every motion, acceleration or deceleration.
A special situation is given for the function block [MC_Stop, page 181](#). The position is kept active by the drive. Thus, the MC_Power function block cannot be switched off when MC_Stop is active.



Together with a virtual axis, the function block results in an error message.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-4: Library assignment

Interface

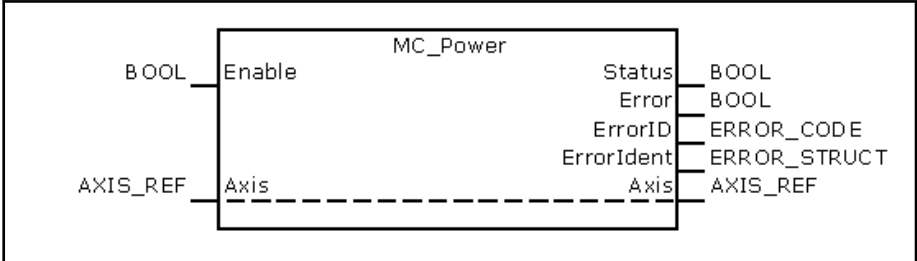


Fig. 6-4: MC_Power function block

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	When "Enable" is TRUE, power is added
VAR_OUTPUT	Status	BOOL	Power adding state Also refer to "Special features of MC_Power" on page 180.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the concrete axis

Tab. 6-5: Interface of the MC_Power function block

Special features of MC_Power

Use case "power off via MC_Power" via FALSE at the "Enable" input.

The drive status is not be evaluated via the "Status" output of the function block, but via PLC access to parameter values using AxisData[] - axis data in the shared memory.

- `AxisData[Drive1.AxisNo].Axis_Power;`
AH_AF - Drive Halt, Drive Enabled

The drive outputs power. That means that it is under torque. The "Drive Halt" function sets an axis with a defined acceleration and jerk to stand-still. "Drive enabled" means that the drive is enabled in the desired operation mode; A-0-0021.15 -RD-

- `AxisData[Drive1.AxisNo].PowerOn;`
Power added; A-0-0022.03 -RD-

Error handling:

- An error message is issued if the "Status" is still FALSE after a timeout at the set "Enable". Indicates that a hardware problem occurred when power was added
- An error message is also issued if it is tried to disconnect the power in an axis state in which this is not permitted.
- If adding on power fails or malfunctions while the axis is moving, an "ErrorStop" is generated for the axis.

The function block MC_Power generates the following error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.5.2 MC_Reset

Brief description

implements the transition from the "ErrorStop" state to the "StandStill" state by resetting all axis- and control-internal errors. Axis-internal errors are deleted via the corresponding axis number in AXIS_REF. Control errors can be reset using the ENUM NO_OBJECT or the axis reference NoAxis in AXIS_REF. The MC_Reset function block uses either the parameter "C-0-1030, Command: Clear all control errors" or "A-0-1030, Command: Clear all axis errors". In the operating mode (phase 4), MLC-internal functions are executed



The outputs of the function block instances that indicated the error(s) are not affected.

If there is no error present, triggering MC_Reset has no effect.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-6: Library assignment

Interface description

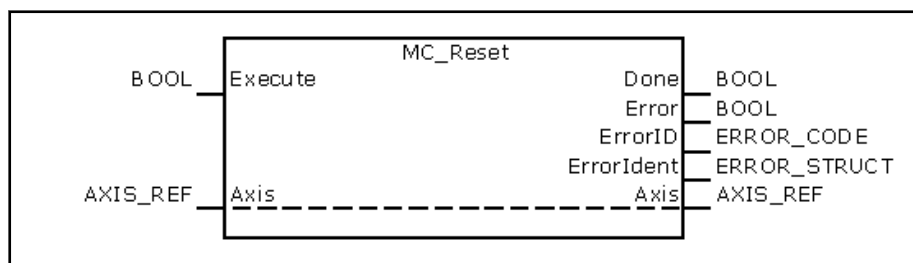


Fig. 6-5: MC_Reset function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
VAR_OUTPUT	Done	BOOL	Errors are reset
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-7: Interface of the MC_Reset function block

Signal-time diagram

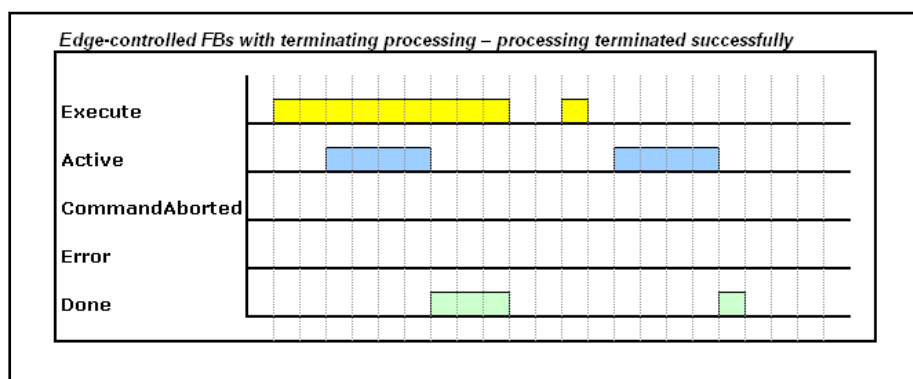


Fig. 6-6: Time diagram of MC_Reset

Troubleshooting

The MC_Reset function block generates the following error messages:

- [MLC_TABLE, page 360](#) - Error when accessing the axis/control.

6.5.3 MC_Stop

Brief description

This function block implements a controlled motion stop and transfers the axis to the "Stopping" state. As a result, any processing function block is stopped. When standstill is reached, the "Done" output is set.

The position is kept active.



When "Execute" is TRUE, the axis is in "Stopping" state and cannot be moved by another command.

In the "Stopping" state, it is not possible to switch off the axis power (MC_Power, page 179). If "Enable" of the MC_Power instance is disabled, its "status" nevertheless remains switched on until "Execute" of the MC_Stop instance is also disabled.

 MC_Stop is not supported for an IndraDrive with active operating stop (P-0-0558 Bit 0 = 1).
The function block never sets the "Done" output to TRUE.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Motion

Tab. 6-8: Library assignment

Interface description

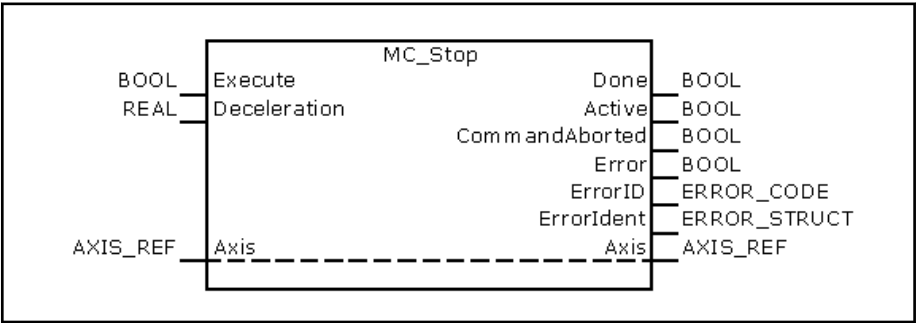


Fig. 6-7: MC_Stop function block

	Name	Type	Comment ¹⁾
VAR_INPUT	Execute	BOOL	As long as "Execute" is TRUE, the axis is stopped or kept in the "Stopping" state.
	Deceleration	REAL	Deceleration (always greater than 0, otherwise no "Stopping").
VAR_OUTPUT	Done	BOOL	Standstill reached
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command abort by following command (of another MC_Stop instance)
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the concrete axis

Tab. 6-9: Interface of the MC_Stop function block

1) The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

Signal-time diagram

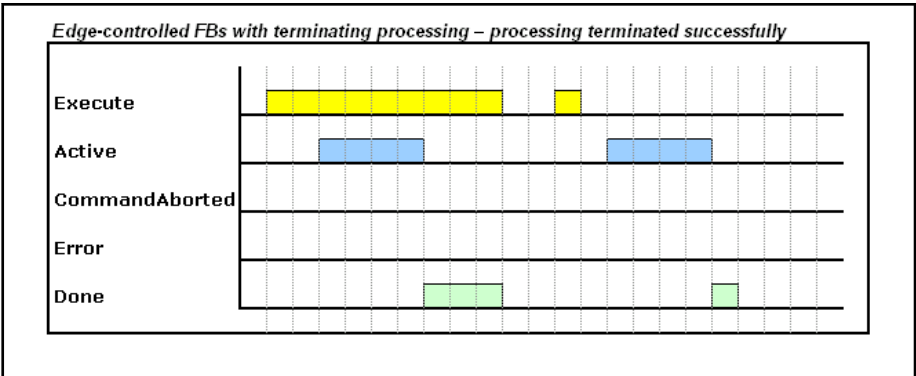


Fig. 6-8: Time diagram of MC_Stop

Troubleshooting The function block MC_Stop generates the following error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.5.4 MC_MoveAbsolute

Brief description The function block copies the absolutely specified "position" to the "target position" and approaches it.



If this function block requires power added before via an instance of the [MC_Power, page 179](#), (for real axes).



1. In addition to the function block inputs, the jerk limit value set via the parameter "A-0-0216, Jerk" is applied to this function block.
If the jerk limitation is enabled ($A-0-0216 > 0$), the behavior described in the following results
2. In the acceleration phase, the "Acceleration" input of the function block is always applied.
If the axis velocity has to be reduced during the acceleration phase (e.g. overshooting of the actual velocity), the "Acceleration" is also applied as limit value to the required deceleration.

In the following case, it is possible that the axis is decelerated with a higher deceleration than specified in the "Deceleration" input:
 - Jerk limitation is active ($A-0-0216 > 0$) AND
 - Acceleration input $>$ Deceleration input AND
 - The axis is in the acceleration phase AND
 - Motion restart with reduced positioning velocity
3. This behavior does not result without jerk limitation ($A-0-0216 = 0$)

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-10: Library assignment

Interface description

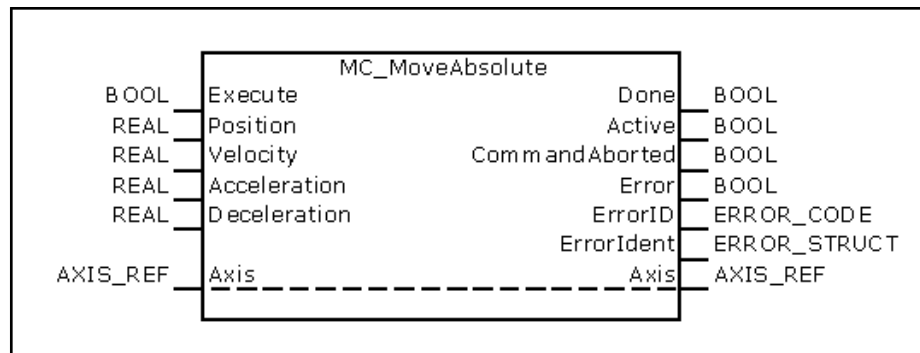


Fig. 6-9: MC_MoveAbsolute function block

	Name	Type	Comment ²⁾
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
	Position	REAL	Target position for the motion (+/-)
	Velocity	REAL	Maximum velocity value (always +) (does not necessarily have to be reached).
	Acceleration	REAL	Acceleration (always +).
	Deceleration	REAL	Deceleration (always +).
VAR_OUTPUT	Done	BOOL	Desired position reached
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-11: Interface of the MC_MoveAbsolute function block

Implementation example: The following figure shows two examples of cooperation of two "MC_MoveAbsolute" function block instances:

1. The left side of the time diagram shows the case in which "Instance2" is called after "Instance1". After "Instance1" has been implemented the desired position 6,000 (velocity 0), its "Done" output enables "Instance2" with the target position 10,000.
2. The right side of the time diagram shows how "Instance2" is already enabled while "Instance1" is still running. The motion of "Instance1" is stopped and aborted by the "Test" signal. "Instance2" directly approaches its target position 10,000, although the position 6,000 has not yet been reached.

²⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

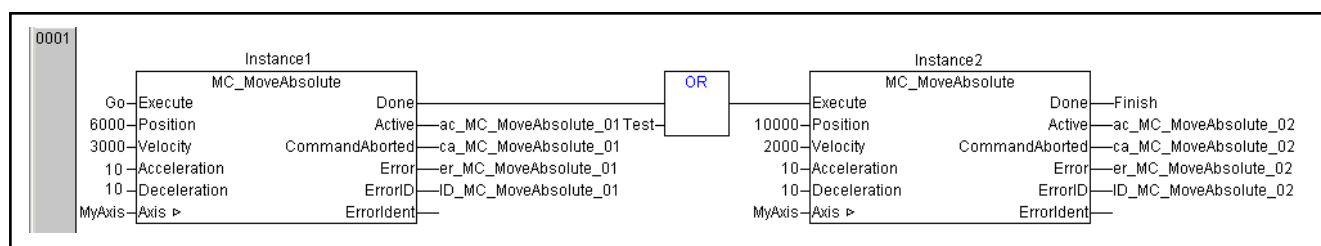


Fig. 6-10: Two instances of MC_MoveAbsolute

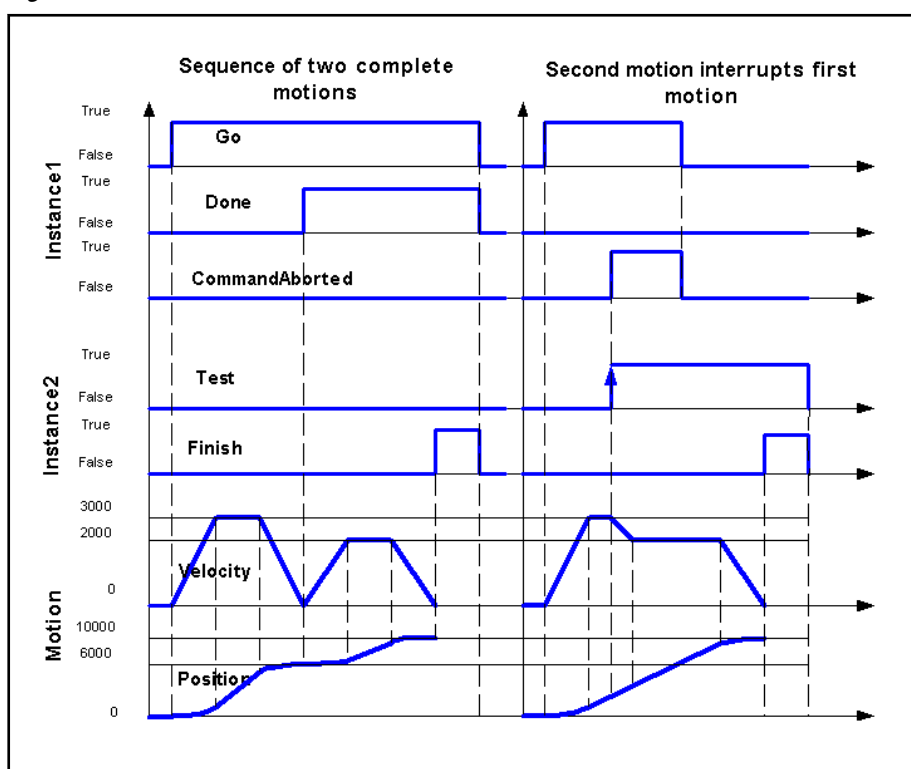


Fig. 6-11: Timing diagram: Two instances of MC_MoveAbsolute

Parameters used (Sercos-based axes)

The function block determines the new target position (A-0-2201, Target position) by applying the command position (A-0-2200, Command position) to the parameter A-0-2201 and approaching it.



In the modulo mode, reaching the target position is influenced by the parameter "A-0-0203, Direction of motion".

Special function:

The function blocks "MC_MoveAbsolute", "MC_MoveAdditive" and "MC_MoveRelative" accept a velocity specification of $V_{com} = 0$.

When these function blocks ($V_{com} = 0$) are called, the axis is stopped. Positioning is only interrupted. The target position remains in the axis.

If MC_MoveAdditive ($V_{com} \neq 0$) is called subsequently, the remaining distance is traversed. The target position is only overwritten when another Motion function block is called.

Troubleshooting:

The MC_MoveAbsolute function block generates the following error messages:

- [MLC_TABLE](#), page 360 - Error when accessing the axis/control.

6.5.5 MC_MoveAdditive

Brief description The function block adds the "distance" to the "target position" and approaches it.

 If this function block requires power added before via an instance of the [MC_Power](#), [page 179](#), (for real axes).

-  1. In addition to the function block inputs, the jerk limit value set via the parameter "A-0-0216, Jerk" is applied to this function block.
- If the jerk limitation is enabled ($A-0-0216 > 0$), the behavior described in the following results
2. In the acceleration phase, the "Acceleration" input of the function block is always applied.
- If the axis velocity has to be reduced during the acceleration phase (e.g. overshooting of the actual velocity), the "Acceleration" is also applied as limit value to the required deceleration.
- In the following case, it is possible that the axis is decelerated with a higher deceleration than specified in the "Deceleration" input:
- Jerk limitation is active ($A-0-0216 > 0$) AND
 - Acceleration input > Deceleration input AND
 - The axis is in the acceleration phase AND
 - Motion restart with reduced positioning velocity
3. This behavior does not result without jerk limitation ($A-0-0216 = 0$)

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Motion

Tab. 6-12: Library assignment

Interface description

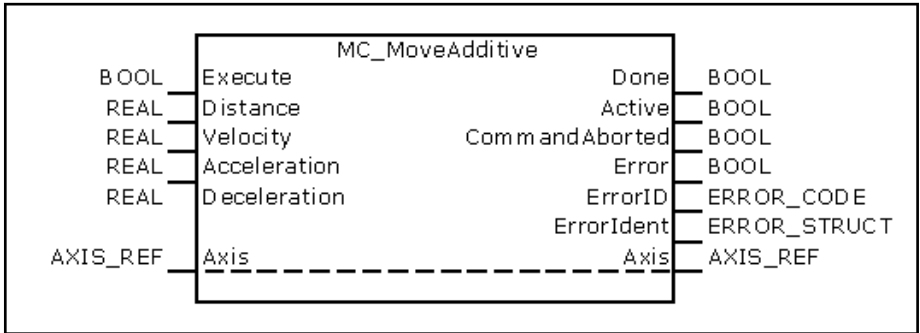


Fig. 6-12: MC_MoveAdditive function block

	Name	Type	Comment ³⁾
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
	Distance	REAL	Relative distance for the motion (+/-).
	Velocity	REAL	Maximum velocity value (always +) (does not necessarily have to be reached).
	Acceleration	REAL	Acceleration (always +).
	Deceleration	REAL	Deceleration (always +).
VAR_OUTPUT	Done	BOOL	Desired position reached
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-13: Interface of the MC_MoveAdditive function block

Implementation example:

The figure below shows two examples of two combined function block instances while the axis is in "Discrete Motion State":



Two different function block types are used:

- Instance1: [MC_MoveAbsolute on page 183](#)
- Instance2: MC_MoveAdditive.

1. The left part of the time diagram shows a case in which the second function block is started after the first one. The first function block reaches the distance 6,000 (at a velocity of 0). The "Done" output causes the second function block to travel a distance of up to 10,000.
2. The right part of the time diagram shows a case in which the second function block is already started while the first function block is still active. In this case, the first motion is stopped and aborted at constant velocity by the "Test" signal. The second function block adds its distance of 4,000 to the originally commanded position of 6,000 and moves the axis to the resulting position 10,000 at a velocity specified at the second function block.

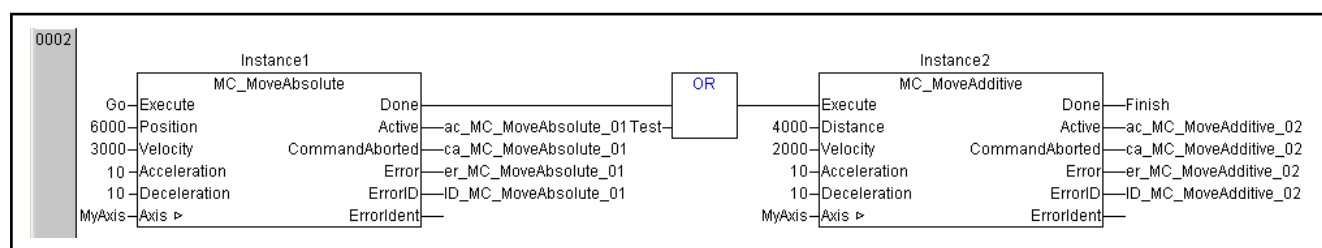


Fig. 6-13: Instances of MC_MoveAbsolute and MC_MoveAdditive

³⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

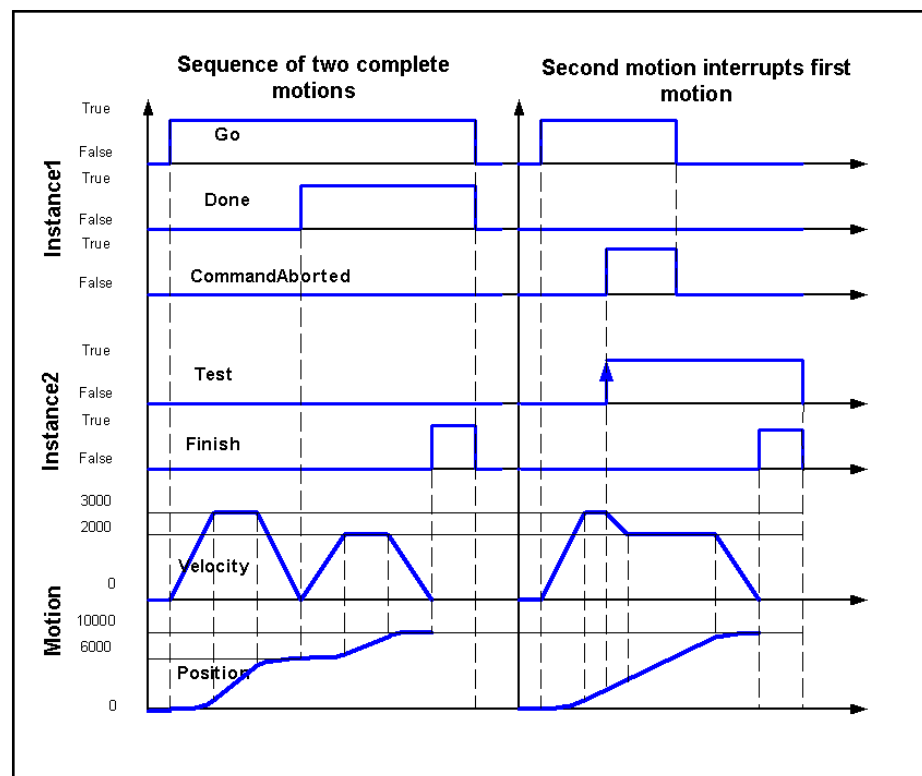


Fig. 6-14: Time diagram: Instances of MC_MoveAbsolute and MC_MoveAdditive

Parameters used (Sercos-based axes)

The function block determines the new target position (A-0-2201, Target position) by adding the distance (A-0-2200, Command position) to the previous target position and approaching it.



The distance is specified relative to the actual position. The parameter A-0-0203, Direction of motion, is not applied and does not overwrite the sign of the distance!

Special function: The function blocks "MC_MoveAbsolute", "MC_MoveAdditive" and "MC_MoveRelative" accept a velocity specification of Vcom = 0.

When these function blocks (Vcom = 0) are called, the axis is stopped. Positioning is only interrupted. The target position remains in the axis.

If MC_MoveAdditive (Vcom ≠ 0) is called subsequently, the remaining distance is traversed. The target position is only overwritten when another Motion function block is called.

Troubleshooting: The MC_MoveAdditive function block generates the following error messages:

- [MLC_TABLE, page 360](#) - Error when accessing the axis/control.

6.5.6 MC_MoveRelative

Brief description The function block determines the new "target position" by adding the "distance" to the previous "actual position" and approaching it.



If this function block requires power added before via an instance of the [MC_Power, page 179](#), (for real axes).



1. In addition to the function block inputs, the jerk limit value set via the parameter "A-0-0216, Jerk" is applied to this function block.

If the jerk limitation is enabled ($A-0-0216 > 0$), the behavior described in the following results

2. In the acceleration phase, the "Acceleration" input of the function block is always applied.

If the axis velocity has to be reduced during the acceleration phase (e.g. overshooting of the actual velocity), the "Acceleration" is also applied as limit value to the required deceleration.

In the following case, it is possible that the axis is decelerated with a higher deceleration than specified in the "Deceleration" input:

- Jerk limitation is active ($A-0-0216 > 0$) AND
- Acceleration input $>$ Deceleration input AND
- The axis is in the acceleration phase AND
- Motion restart with reduced positioning velocity

3. This behavior does not result without jerk limitation ($A-0-0216 = 0$)

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Motion

Tab. 6-14: Library assignment

Interface description

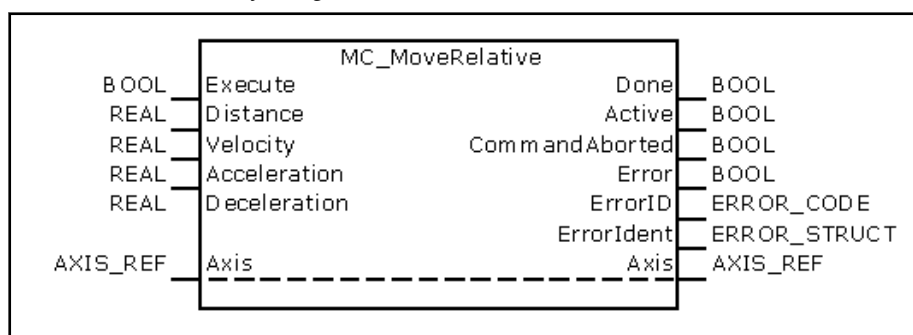


Fig. 6-15: MC_MoveRelative function block

	Name	Type	Comment ⁴⁾
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
	Distance	REAL	Relative distance for the motion (+/-).
	Velocity	REAL	Maximum velocity value (always +) (does not necessarily have to be reached).
	Acceleration	REAL	Acceleration (always +).
	Deceleration	REAL	Deceleration (always +).

⁴⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

	Name	Type	Comment ⁴⁾
VAR_OUTPUT	Done	BOOL	Desired position reached
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-15: Interface of the MC_MoveRelative function block

Signal-time diagram

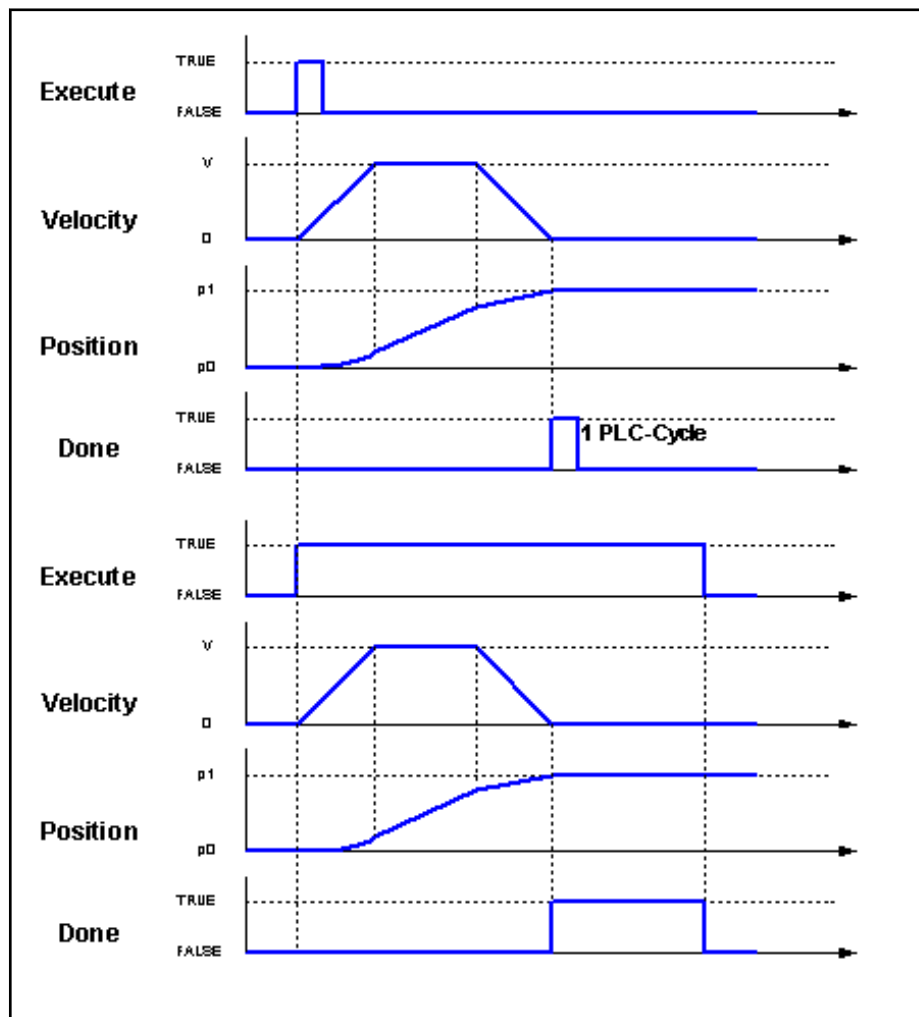


Fig. 6-16: Time diagrams for MC_MoveRelative

Implementation Example

The following example shows the combination of two MC_MoveRelative function block instances.

⁴⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

1. The left part of the time diagram shows a case in which "Instance2" is called after "Instance1" was processed. After "Instance1" has reached the desired distance 6,000 (velocity 0), its "Done" output uses "Instance2" to cause a further motion by 4,000 to reach the total distance of 10,000.
2. The right part shows how "Instance2" is already enabled by the "Test" signal while "Instance1" is still running. The first motion is aborted during the constant velocity of "Instance1". "Instance1" adds its distance of 4,000 to the current position (3,250). The axis approaches the resulting position of 7,250.

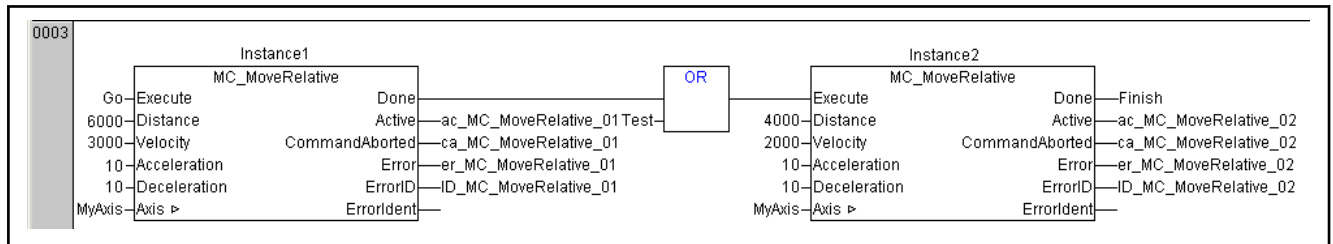


Fig. 6-17: Two instances of MC_MoveRelative

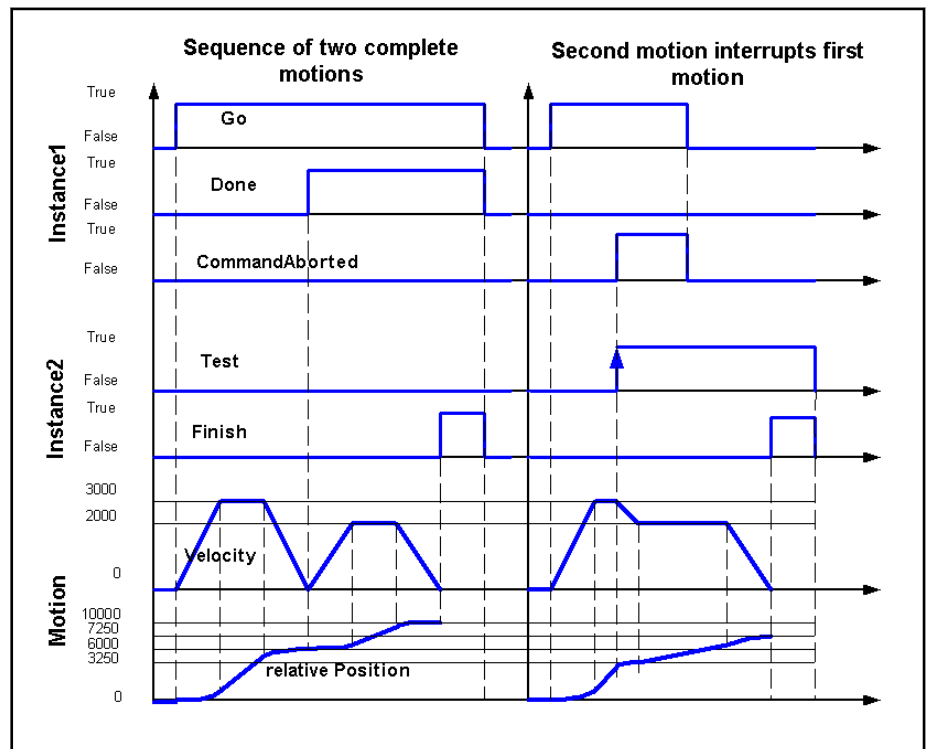


Fig. 6-18: Timing diagram: Two instances of MC_MoveRelative

Parameters used
(Sercos-based axes)

This function block determines the new target position (A-0-2201, Target position) by adding the distance (A-0-2200, Command position) and the actual position and approaching it.



The distance is specified relative to the actual position. The parameter A-0-0203, Direction of motion, is not applied and does not overwrite the sign of the distance!

If it is interpolated in the control, the distance is added to A-0-0150, Interpolated position and if it is interpolated in the drive, the distance is added to A-0-0100, Actual position value.

- Special function:** The function blocks "MC_MoveAbsolute", "MC_MoveAdditive" and "MC_MoveRelative" accept a velocity specification of $V_{com} = 0$.
When these function blocks ($V_{com} = 0$) are called, the axis is stopped. Positioning is only interrupted. The target position remains in the axis.
If MC_MoveAdditive ($V_{com} \neq 0$) is called subsequently, the remaining distance is traveled. The target position is only overwritten when another Motion function block is called.
- Troubleshooting:** The MC_MoveRelative function block generates the following error messages:
- [MLC_TABLE, page 360](#) - Error when accessing the axis/control.

6.5.7 MC_MoveVelocity

Brief description This function block controls an "endless" motion at the respectively defined velocity. To end the motion, the function block instance has to be interrupted by another function block instance which issues a new command.



If this function block requires power added before via an instance of the [MC_Power, page 179](#), (for real axes).



1. In addition to the function block inputs, the jerk limit value set via the parameter "A-0-0216, Jerk" is applied to this function block.

If the jerk limitation is enabled ($A-0-0216 > 0$), the behavior described in the following results

2. In the acceleration phase, the "Acceleration" input of the function block is always applied.

If the axis velocity has to be reduced during the acceleration phase (e.g. overshooting of the actual velocity), the "Acceleration" is also applied as limit value to the required deceleration.

In the following case, it is possible that the axis is decelerated with a higher deceleration than specified in the "Deceleration" input:

- Jerk limitation is active ($A-0-0216 > 0$) AND
 - Acceleration input > Deceleration input AND
 - The axis is in the acceleration phase AND
 - Motion restart with reduced positioning velocity
3. This behavior does not result without jerk limitation ($A-0-0216 = 0$)

The "InVelocity" signal is reset if the function block instance is interrupted by another function block instance (CommandAborted) or if the edge of its "Execute" input is falling.



If "Execute" is disabled, the drive keeps running on the same velocity.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-16: Library assignment

Interface description

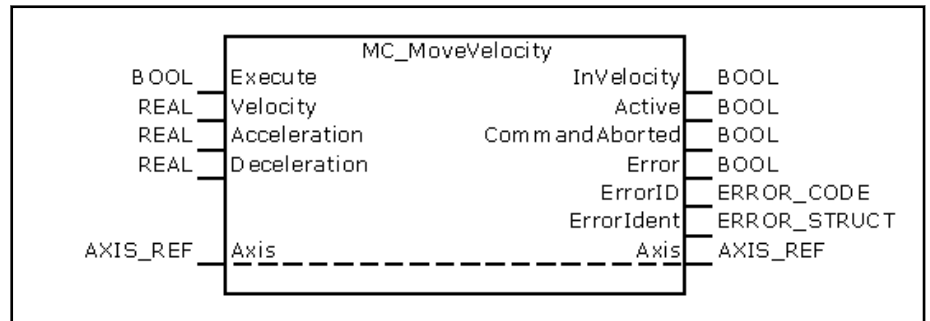


Fig. 6-19: MC_MoveVelocity function block

	Name	Type	Comment ⁵⁾
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
	Velocity	REAL	Maximum velocity value (does not necessarily have to be reached).
	Acceleration	REAL	Acceleration (always +).
	Deceleration	REAL	Deceleration (always +).
VAR_OUTPUT	InVelocity	BOOL	Velocity reached (for the first time)
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-17: Interface of the MC_MoveVelocity function block

⁵⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

Signal-time diagram

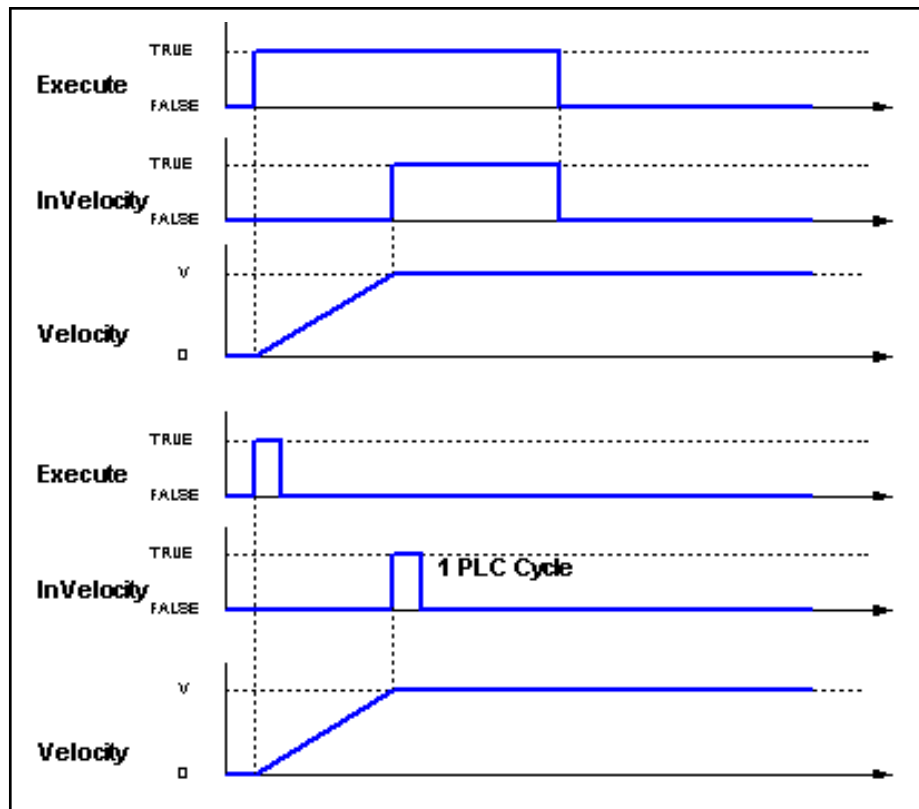


Fig. 6-20: Time diagram of MC_MoveVelocity (without aborted command)

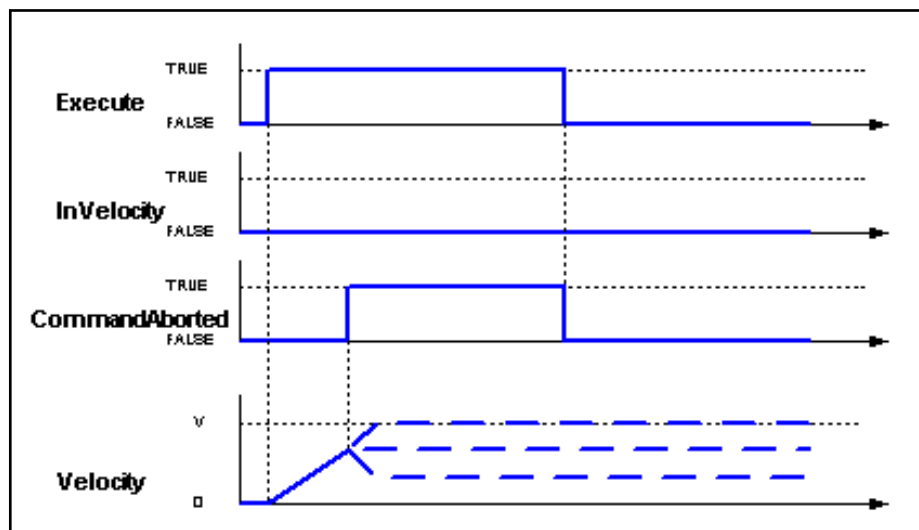


Fig. 6-21: Time diagram of MC_MoveVelocity (with aborted command)

Implementation example:

The following figure shows two examples of two combined instances of the MC_MoveVelocity function block:

1. The left part of the time diagram shows a case in which the second instance is called after the job of the first instance is complete. Once the first function block has reached the required velocity of 3,000, the "Instance1.InVelocity" output **AND** the "Next" signal cause the second function block to continue the motion at a velocity of 2,000.
2. The right side of the time diagram shows a case where the first function block has not yet reached the velocity when the second function block is started. The following is shown: The first motion is restarted by "Go" at

the "Instance1.Execute" input. While the first function block is still accelerating to reach a velocity of 3,000, it is interrupted by the "Test" signal which starts the second function block. The second function block is now running and decelerates to a velocity of 2,000.

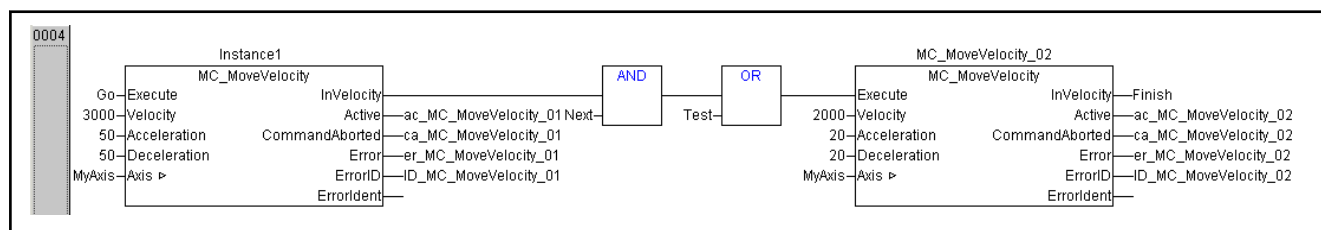


Fig. 6-22: Two instances of MC_MoveVelocity

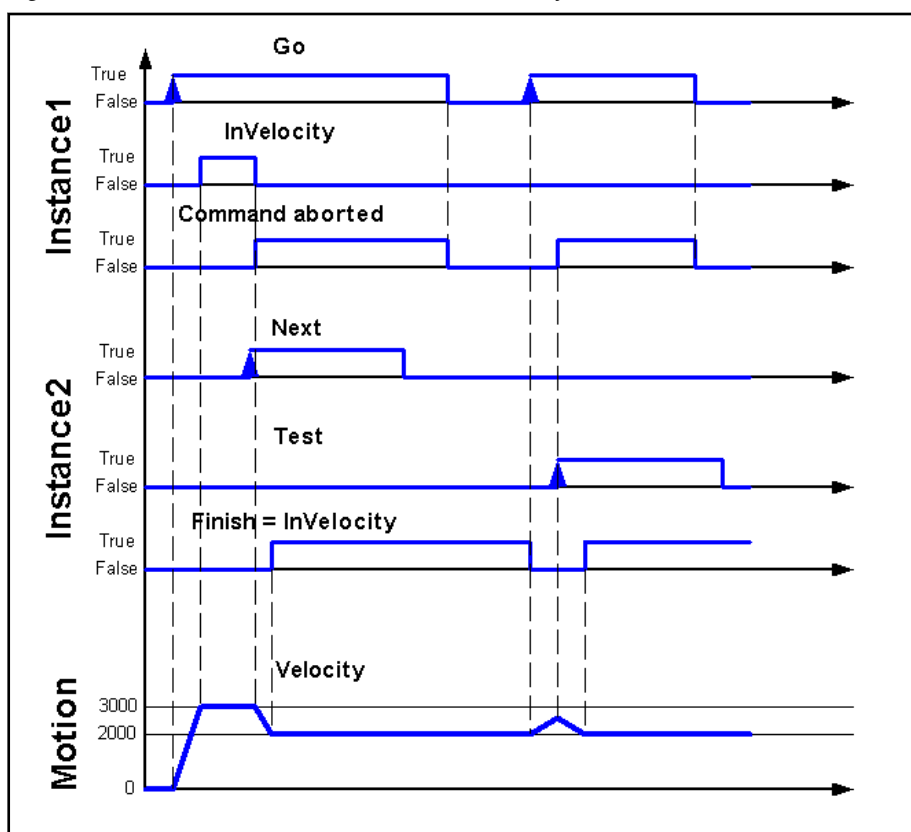


Fig. 6-23: Timing diagram: Two instances of MC_MoveVelocity

Troubleshooting: The MC_MoveVelocity function block generates the following error messages:

- [MLC_TABLE, page 360](#) - Error when accessing the axis/control.

6.5.8 MC_TorqueControl

Brief description

The function block controls a "continuously determined force with a defined force ramp in a continuous motion". Depending on the scaling, the function block is suited for power determination and torque determination (positive power/torque determination in the sense of positive velocity determination). To end the motion, the function block instance has to be interrupted by another function block instance which issues a new command.



If this function block requires power added before via an instance of the [MC_Power, page 179](#), (for real axes).



Depending on axis configuration and interpolation in the drive/control, MC_TorqueControl responses as follows:

The torque ramp in the drive is calculated in case of **interpolation in the drive**. MC_TorqueControl writes to the parameters:

- Torque input has an effect on S-0-0080, Torque, force command value
- TorqueRamp input has an effect on S-0-0822, Torque, force ramp
- S-0-0823, Torque/force ramp time The control firmware configures the parameter with a fixed value of 1000ms

The torque ramp is calculated in the control in case of **interpolation in control**.

- The control writes cyclically on "S-0-0080, Torque, force command value".
- The generation of a torque ramp in the drive is disabled (S-0-0822 = 0, S-0-0823 = 0).
- The TorqueRamp input corresponds to a change in torque within 1000ms.



The functionality is supported by the following axis types:

- IndraDrive with interpolation in the drive
- IndraDrive with interpolation in the control
- SercosDrive
- Controller axis

The "InTorque" signal is reset if the function block instance is interrupted by another function block instance (CommandAborted) or if the edge of its "Execute" input is falling.



If "Execute" is disabled, the drive keeps running on the same velocity.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Motion

Tab. 6-18: Library assignment

Interface description

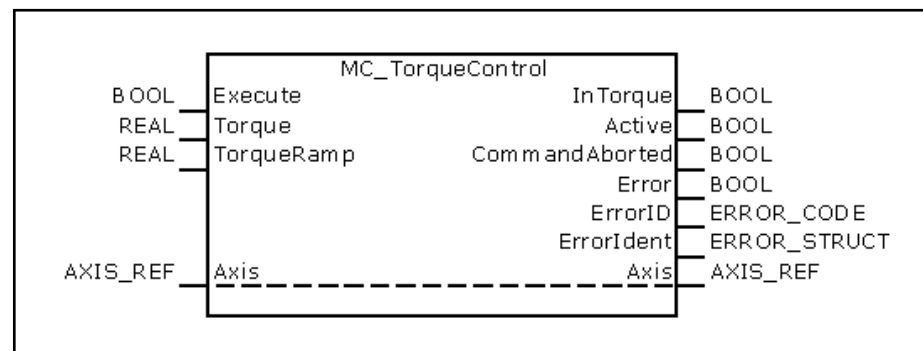


Fig. 6-24: MC_TorqueControl function block

	Name	Type	Comment ⁶⁾
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge. The input values are applied at each edge.
	Torque	REAL	Maximum value of the power/torque (always positive, value does not necessarily have to be reached).
	TorqueRamp	REAL	Power/torque difference which can be reached within the ramp time
VAR_OUTPUT	InTorque	BOOL	Power/torque reached (for the first time)
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-19: Interface of the MC_TorqueControl function block

Signal-time diagram

Edge-controlled function block with permanent processing - processing completed

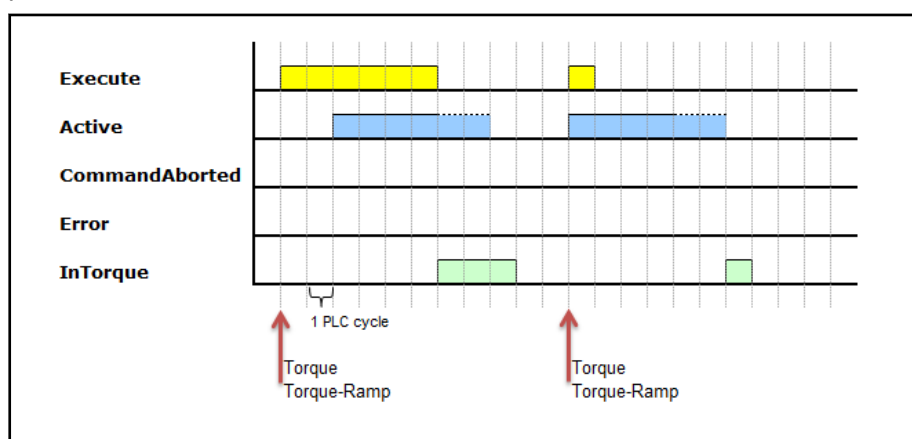


Fig. 6-25: Signal course of time of the MC_TorqueControl

Activation/deactivation of "Torque"

Applying the torque command value to the master data telegram has to be enabled in the parameter "A-0-0007, Axis configuration", bit 2.

Parameters used
(Sercos-based axes)

The function block operates together with the following Sercos parameters:

- S-0-0091, Bipolar velocity limit value (to be set before the function block call)
- S-0-0823, Ramp time (to be set before function block call)

⁶⁾ The measuring units of time, position, velocity, acceleration, deceleration and power/torque depend on the effective scaling.



Set ramp time to 1,000 ms and specify the torque/power difference to be programmed related to one second at the function block input.

- S-0-0822, Torque/force ramp, applied by the function block.



The parameters S-0-0822 and S-0-0823 are supported by the drive variants from HNC3x Sercos and MPH04.

Troubleshooting: The function block MC_TorqueControl generates the following error messages:

- [MLC_TABLE](#), page 360 - Error when accessing the axis/control.

6.5.9 MB_SetPositionControlMode

Brief description The "MB_SetPositionControlMode" function block is used to set the position controller.

The function block can be used to set the position control method (with or without lag) and to set the selection of the actual position value (motor encoder or external encoder).

The reference values for the relative positioning can be set.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-20: Library assignment

Interface

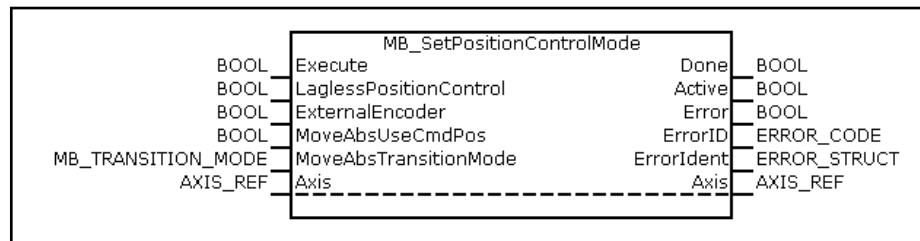


Fig. 6-26: MB_SetPositionControlMode

	Name	Data type	Description
VAR_INPUT	Execute	BOOL	Function block processing enabled (once, edge-controlled)
	LaglessPositionControl	BOOL	Affects bit 0 of "A-0-0007, Axis configuration": FALSE: Position control with lag error TRUE: Position control without lag error
	ExternalEncoder	BOOL	Affects bit 1 of "A-0-0007, Axis configuration": FALSE: Control with motor encoder (actual position value-1) TRUE: Control with external encoder (actual position value-2)
	MoveAbsUseCmdPos	BOOL	Affects bit 2 of "A-0-0201, Interpolator configuration": FALSE: Current actual position value as reference point for relative position command values TRUE: Latest effective target position as reference point for relative position command values

	Name	Data type	Description
	MoveAbsTransition-Mode	MB_TRANSITION_MODE	Default mode: "MB_ABORTING", currently, no other modes are intended
VAR_OUTPUT	Done	BOOL	"Done"=TRUE indicates that the task was completed successfully
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-21: Interface of the MB_SetPositionControlMode function block

Functional Description

The "LaglessPositionControl" input specifies whether it is to be controlled without lags by the velocity feedforward.

The "ExternalEncoder" input selects the actual position control value used to close the position control loop. The axis types "IndraDrive" and "Hydraulic-Drive" are supported.

The measuring wheel mode is switched on and off when an external encoder is configured as measuring wheel encoder.



The axis at the "Axis" input may not be used as master axis for a real axis and it may not generate any master axis data in an MLC cross link.

Troubleshooting

The function block MB_SetPositionControlMode generates the following error messages:

- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 360 - Error when accessing the axis/the control.

6.5.10 MB_Home

Brief description

This function block controls **axes with incremental encoder** when executing the "search home" sequence. The details of the sequence must be set via axis parameters. "Homing" ends at standstill.



The MB_Home function block may not be used together with virtual axes.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-22: Library assignment

Interface description

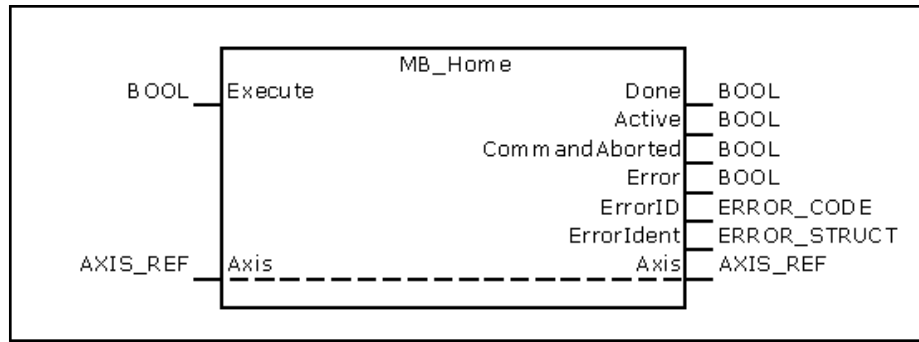


Fig. 6-27: MB_Home function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
VAR_OUTPUT	Done	BOOL	Homing completed, velocity 0
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the concrete axis

Tab. 6-23: Interface of the MB_Home function block

Signal-time diagram

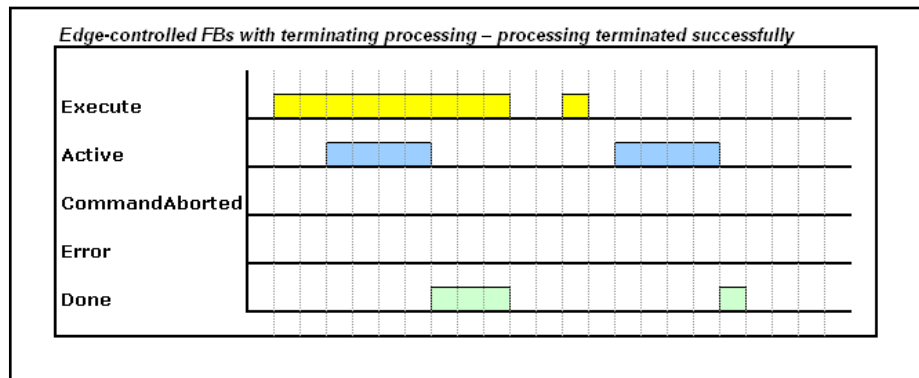


Fig. 6-28: Time diagram of MB_Home

Error handling: The function block MB_Home generates the following error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.6 Function blocks for multi-axis control

6.6.1 The MotionControl action chain

The function blocks MC_CamIn, MB_GearInPos, MC_GearIn, MB_Motion-Profile and ML_FlexProfile were each extended to MotionControl action chains.

In addition to the previously possible Sercos cycle times of 2, 4, 8, 16 ms, a **Sercos cycle time of 1ms** can be set.

For an overview on possible drives, refer to:

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Functional Description, Axes ► Real axis ► General

DOK-XLCMLC-FUNC****V14-APRS-EN-P, R911341700

This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraMotion MLC.

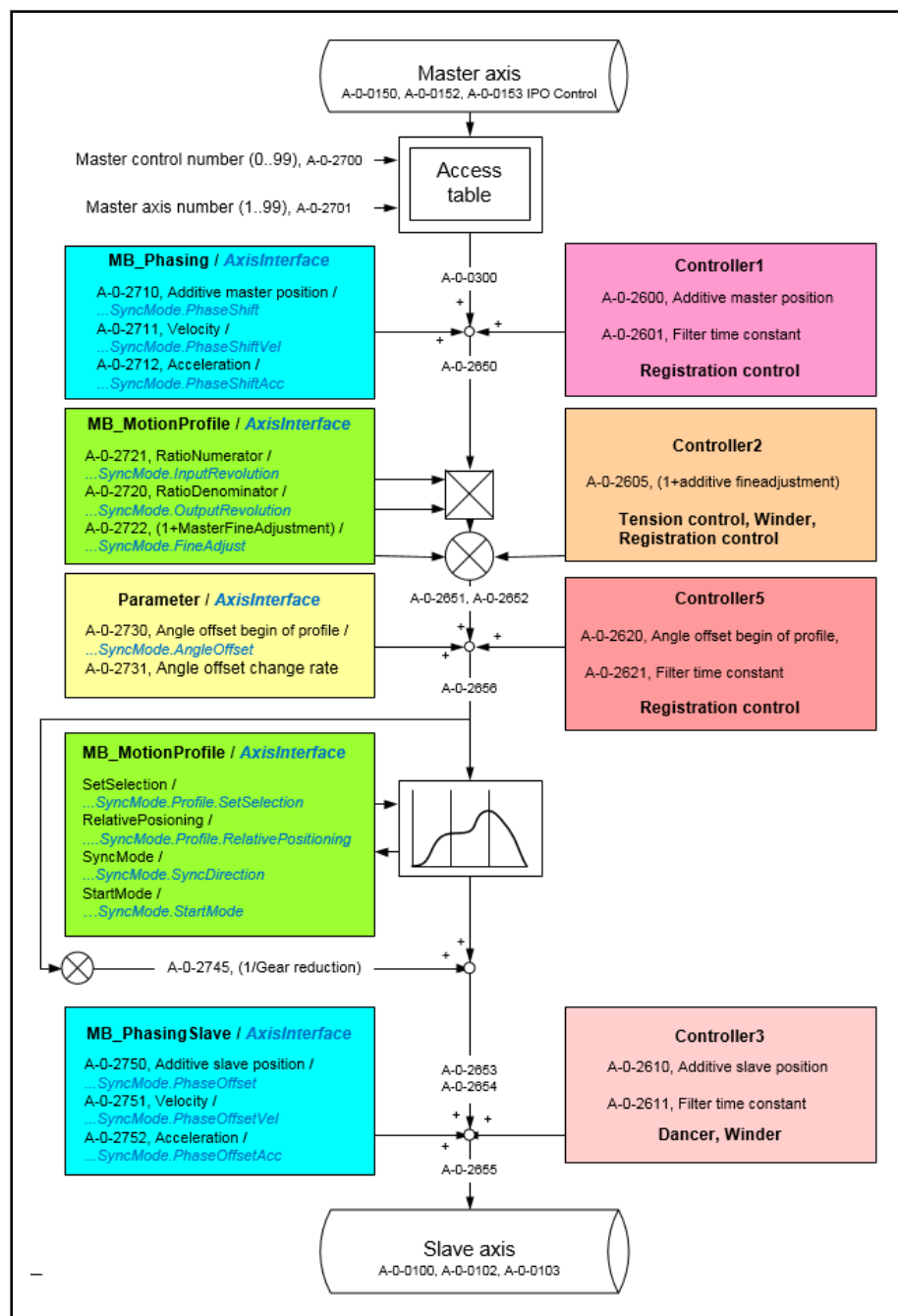


Fig. 6-29: Action chain MotionProfile as an example

With an active MB_MotionProfile, it is possible to run the function blocks MB_Phasing, MB_PhasingSlave and later also "MB_PhasingTableAngle" simultaneously (not in current version).

Motions programmed in "MB_Phasing" overlap with the motion of the main function block "MB_MotionProfile". The same applies to "MB_PhasingSlave" and later on for "MB_PhasingTableAngle" as well.

All specified function blocks are equipped with individual parameters and an individual interpolator.

For every parallel function block, there is an additional controller. The additive positions from the controllers are effective at the same place of summation as the corresponding parallel function block. The positions summarized via the controllers are not interpolated but are effective via a first order filter with programmable time constant.

"A-0-2655, ELS, position command value" is generated at the end of the action chain.

The slave axis can be a virtual axis (cascadable) or a real generic axis. The position command value of the slave axis is directly specified in the drive via the Sercos position command value interface.

For most of the function blocks, calculation is possible either in the drive or on the control (from version MLC 03VRS). This decision is made during control startup for each axis, for all function blocks of this axis together depending on the type of drive (virtual axis, generic axis, IndraDrive, EcoDrive CS).



The introduction of a new parameter can change this decision. However, the decision still applies to all the function blocks of an axis.

The function blocks "MB_MotionProfile" and "ML_FlexProfile" are only calculated on the control.

They can only be programmed if all function blocks of this axis are calculated on the control.

In case of non-compliance, an error message is generated.



The function block "MC_CamIn" is only supported in an intelligent drive (e.g. IndraDrive) and it can only be programmed if all the function blocks of this axis are calculated on the intelligent drive (e.g. IndraDrive).

The function block "MC_CamIn" is not calculated on the control.

Sercos III, C2C link

The master axis can be directly accessed in a Sercos III, C2C link, whereas this is not possible in Sercos II, MLC link, in which a link axis had to be defined in the local control. This link axis could serve as a master axis.

The axes relevant for the link have to be defined in a global variable list:

Address definition: C2C link

```
VAR_GLOBAL CONSTANT
M_RA11: AXIS_REF := (CntrlNo:=CNTRL_1, AxisNo:=AXIS_1);
M_RA13: AXIS_REF := (CntrlNo:=CNTRL_1, AxisNo:=AXIS_3);
S_RA21: AXIS_REF := (CntrlNo:=CNTRL_2, AxisNo:=AXIS_1);
S_RA22: AXIS_REF := (CntrlNo:=CNTRL_2, AxisNo:=AXIS_2);
END_VAR
```

If a master axis RA11 of control CNTRL_1 is required in control CNTRL_2 for axis RA21, the following function block wiring is required when using the automatically created global variable list of control CNTRL_2.

Excerpt from MlcVarGlobal of control CNTRL_2

```
VAR_GLOBAL CONSTANT
(*real axis "RA22" with axis number 2*)
RA22: AXIS_REF := (CntrlNo:=LOCAL_CNTRL, AxisNo:=AXIS_2);
(*real axis "RA21" with axis number 1*)
RA21: AXIS_REF := (CntrlNo:=LOCAL_CNTRL, AxisNo:=AXIS_1);
```

END_VAR (* *)

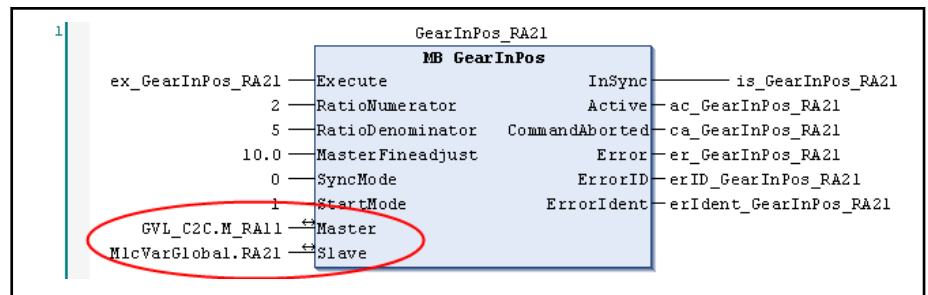


Fig. 6-30: Function blocks with master axis of removed control

6.6.2 MB_Phasing

Brief description

With the function block MB_Phasing, an offset of the master axis, positioned as input at a synchronization function block (MC_CamIn, MB_GearInPos, MC_GearIn, MB_MotionProfile or ML_FlexProfile), is changed.

Also refer to

- [MC_CamIn](#), special features of Sercos-based axes on page 209
- [MC_GearIn](#), special features of Sercos-based axes on page 245
- [MB_GearInPos](#), special features of Sercos-based axes on page 249
- [MB_MotionProfile](#), special features of Sercos-based axes on page 215
- [ML_FlexProfile](#), special features of Sercos-based axes on page 230

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-24: Library assignment

Interface description

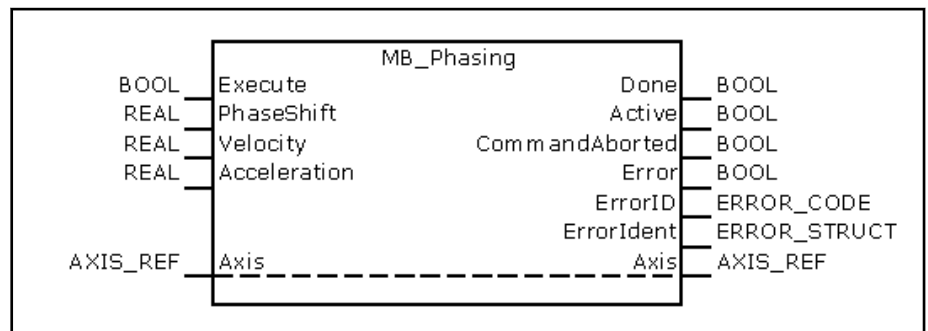


Fig. 6-31: FB MB_Phasing

	Name	Type	Comment ⁷⁾
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
	PhaseShift	REAL	Position difference for the motion (+/-)
	Velocity	REAL	Maximum velocity value in the units of the master axis (always +) (does not necessarily have to be reached).
	Acceleration	REAL	Acceleration in the units of the master axis (always +).

⁷⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling of the master axis.

	Name	Type	Comment ⁷⁾
VAR_OUTPUT	Done	BOOL	The position offset has successfully been traveled
	Active	BOOL	Processing of data after preprocessing running
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the slave axis

Tab. 6-25: Interface of the MB_Phasing FB

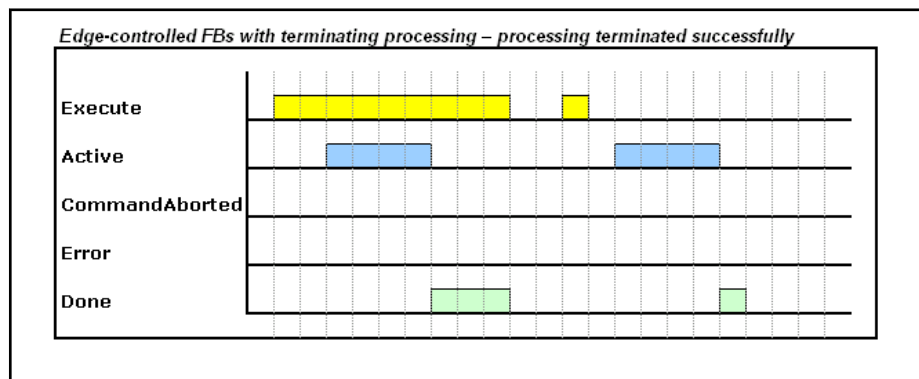
Signal-time diagram

Fig. 6-32: Signal-time diagram

Functional description

The inputs "PhaseShift", "Velocity" and "Acceleration" determine the Motion-Profile with which the path is traveled.

The traveling motion only takes place if a synchronous operation mode is active, and then related to the valid master axis.

If no synchronous mode is active, the value "PhaseShift" is immediately applied and considered during the transition to a synchronous mode. Since no master axis is valid then, the master axis referred to it that is indicated in "A-0-0200, Default master axis". If no valid master axis is indicated there either, an error is generated.

The units of the inputs defining the axis motion can be found in the scaling parameters of the master axis.

The function block works parallel to the respective main function block (MC_GearIn, MB_GearInPos, MC_CamIn, MB_MotionProfile, ML_FlexProfile).

The following control parameters are used:

- A-0-2710, ELS, phasing, additive position command value
- A-0-2711, ELS, phasing, additive position command value, velocity
- A-0-2712, ELS, phasing, additive position command value, acceleration

Processing is done

- for virtual and generic axes and, if desired, for IndraDrive, as function in the control.

⁷⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling of the master axis.

- for IndraDrive in the drive after transmission of the A-parameters to:
 - P-0-0054, Additive master axis position
 - P-0-0688, Additive master axis position, positioning velocity
 - P-0-0689, Additive master axis position, positioning acceleration.

With generic/virtual axes, the phase offset is added immediately if no synchronous mode is active.

The state of the "Done" output is also imaged to the real-time bit "Axis_In-MasterPhaseOffset" of the axis data structure.

The function block always travels the shortest distance.

The value "PhaseShift" is only traveled absolutely, i.e. if the function block is started with the value 50 and then again with the value 70, the offset after the second start is 70 and not $50 + 70 = 120$.

Only offsets smaller than the modulo value of the respective slave axis are traveled. If however, the value is higher, at first a modulo operation is carried out and then the motion is started.



For master axes to be exchanged via the ILL control link, the rotary scaling with 360° modulo value is used for calculation (scaling information is not transferred via the control link).

Troubleshooting

The function block MB_Phasing generates the following error messages:

- [MLC_TABLE](#), page 360 - error when accessing the axis/control.

6.6.3 MB_PhasingSlave

Brief description

The function block MB_PhasingSlave implements an adjustment of the slave axis with specified velocity and acceleration.

Also refer to

- [MC_CamIn](#), special features of Sercos-based axes on page 209
- [MB_GearInPos](#), special features of Sercos-based axes on page 249
- [MB_MotionProfile](#), special features of Sercos-based axes on page 215
- [ML_FlexProfile](#), special features of Sercos-based axes on page 230

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-26: Library assignment

Interface

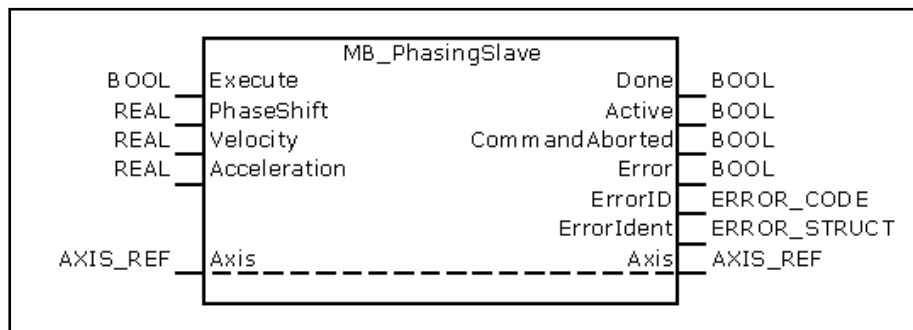


Fig. 6-33: MB_PhasingSlave FB

	Name	Type	Comment ⁸⁾
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
	PhaseShift	REAL	Position difference for the motion (+/-).
	Velocity	REAL	Maximum velocity value (always +) (does not necessarily have to be reached).
	Acceleration	REAL	Acceleration (always +).
VAR_OUTPUT	Done	BOOL	Desired difference reached
	Active	BOOL	Processing of data after preprocessing running
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-27: Interface of the MB_PhasingSlave FB

Signal-time diagram

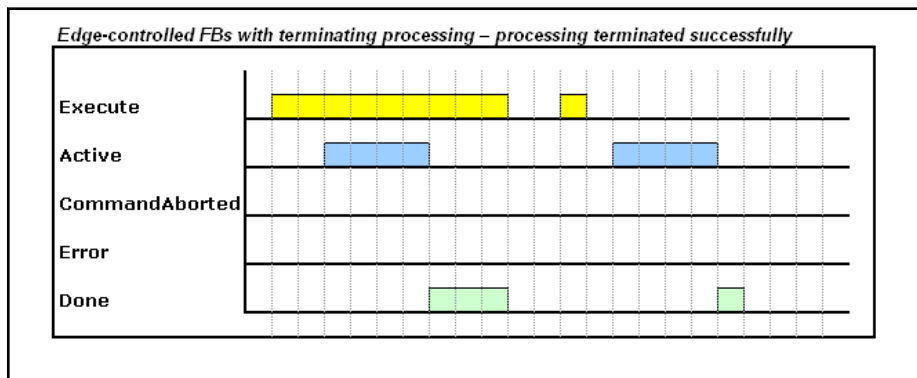


Fig. 6-34: Signal-time diagram

Functional description

The inputs "PhaseShift", "Velocity" and "Acceleration" determine the Motion-Profile with which the path is traveled.

The traveling motion only takes place if a synchronous mode is active.

If no synchronous mode is active, the value "PhaseShift" is immediately applied and considered during the transition to a synchronous mode.

The units of the inputs defining the drive motion can be found in the scaling parameters of the slave axis.

The function block runs parallel to the respective main function blocks (MC_CamIn, MB_GearInPos, MC_GearIn, MB_MotionProfile or ML_FlexProfile).



The function block is not suitable for MPx3/MC_CamIn since these do not have the necessary P-0-0686, P-0-0687.

⁸⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling of the slave axis.

The following control parameters are used:

- A-0-2750, ELS, PhasingSlave, additive position command value
- A-0-2751, ELS, PhasingSlave, additive position command value, vel.
- A-0-2752, ELS, PhasingSlave, additive position command value, acc.

Processing is done

- for virtual axes and SercosDrives as function in the control.
- for IndraDrive in the drive after transmission of the A-parameters to:
 - S-0-0048, Position command value additive
 - P-0-0686, Position command value additive, positioning velocity and
 - P-0-0687, Position command value additive, positioning acceleration.

If the slave axis is in a synchronous mode and this function block is started, the "InSync" bit of the synchronous function blocks (MB_GearInPos, MC_CamIn, MB_MotionProfile) accepts the value "FALSE" until the function block MB_PhasingSlave sets the "Done" output.

During interpolation in the drive, the "InSync" bit is always set to FALSE for one cycle, even if no position difference should be traveled.

The function block always travels the shortest distance.

The value "PhaseShift" is only traveled absolutely, i.e. if the function block is started with the value 50 and then again with the value 70, the offset after the second start is 70 and not $50 + 70 = 120$.

Only offsets smaller than the modulo value of the respective slave axis are traveled. If however, the value is higher, at first a modulo operation is carried out and then the motion is started.

Troubleshooting

The function block MB_PhasingSlave generates the following error messages:

- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 360 - Error when accessing the axis/the control.

6.6.4 MC_CamIn

Brief description

The MC_CamIn function block implements an electronic cam function.



Cam table functionality is supported by the following axis types (slave axis):

- IndraDrive with interpolation in the drive

The function block "MC_CamIn" cannot be calculated on the control.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-28: Library assignment

Interface description

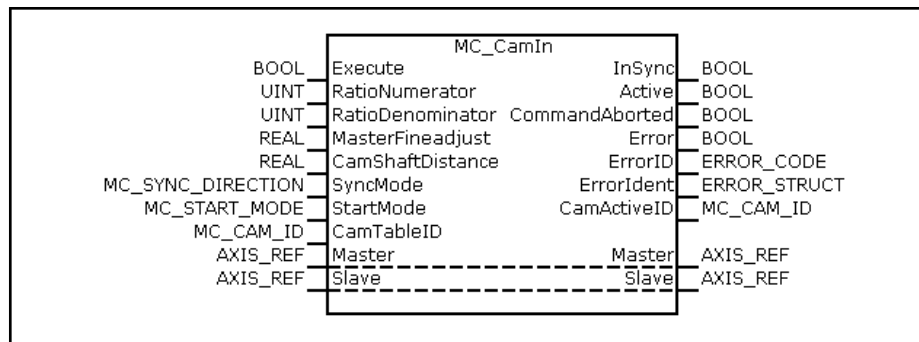
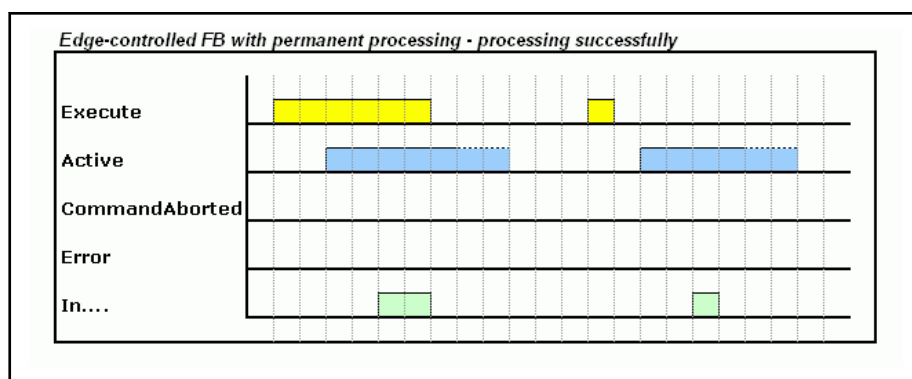


Fig. 6-35: FB MC_CamIn

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
	RatioNumerator	UINT	Gear ratio - numerator "master axis gear output revolutions" Min: 1; max: 65535; default: 1
	RatioDenominator	UINT	Gear ratio - denominator "master axis gear input revolutions" Min: 1; max: 65535; default: 1
	MasterFineadjust	REAL	"Fine adjustment of gear ratio" in %; default: 0.0, min.: -799.9999%, max.: 799.9999
	CamShaftDistance	REAL	"Cam stroke"
	SyncMode	MC_SYNC_DIRECTION	Synchronization direction: 0: = Shortest distance; 1: = Positive direction; 2: = Negative direction
	StartMode	MC_START_MODE	Synchronization type: 0:= absolute, 1:= relative
	CamTableID	MC_CAM_ID	Number of the cam to be activated 1..4(...8)
VAR_OUTPUT	InSync	BOOL	Slave axis follows master axis; coupling completed successfully
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	CamActiveID	MC_CAM_ID	Number of the cam active in the block
VAR_IN_OUT CONSTANT	Master	AXIS_REF	Contains information on the master axis
	Slave	AXIS_REF	It contains information on the slave axis which follows the master axis

Tab. 6-29: Interface of the MC_CamIn function block

Signal-time diagram



in... InSync

Fig. 6-36: Signal-time diagram

Troubleshooting

The function block MC_CamIn generates the following error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.6.5 MC_CamIn, special features of Sercos-based axes

Functional description

The MC_CamIn function block implements two jobs:

- It defines how the axis interprets or processes the cam data saved to it,
- Requests the axis to switch to the CAM editing mode and to edit one motion cycle (defined by the Sercos cycle). Once the function block is processed successfully, the axis remains in the CAM editing mode until it receives another command or an error occurs.

The objective is to implement the dependency stored in the cam table between the master axis (A-0-2700, ELS, master axis, control number, A-0-2701, ELS, master axis, axis number) and the selected slave axis.



Cam table functionality is supported by the following axis types (slave axis):

- IndraDrive with interpolation in the drive

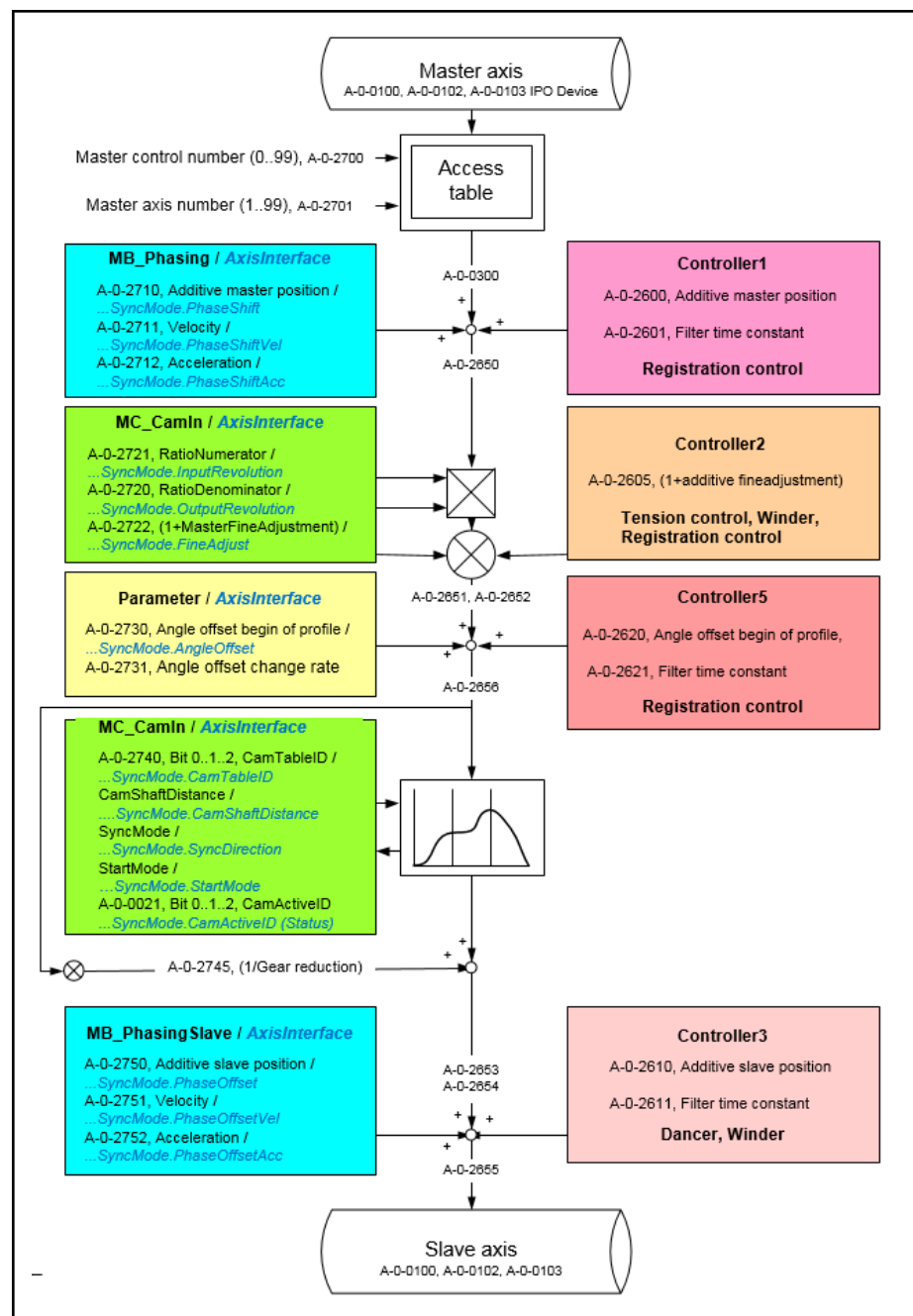


Fig. 6-37: MotionControl action chain MC_CamIn

The operation mode provides the following access points:

- The additive master axis position defined by "A-0-2710, ELS, phasing, additive position command value" can be specified via MB_Phasing or the AxisInterface
- Electronic gear defined by the ratio of "A-0-2721, ELS, master axis gear, output revolutions" and "A-0-2720, ELS, master axis gear, input revolutions" and the "A-0-2722, ELS, gear ratio fine adjust". The fine adjustment value is specified in percent and calculated as $(1 + \text{fine adjustment})$

The selection of the two possible cam tables to be previously stored in the drive: "A-0-2740, ELS, CamIn, cam table preselection".

These values are specified via MC_CamIn or the AxisInterface

- Scaling for the slave axis: A-0-2741, ELS, CamIn, cam distance
- The additive slave axis position defined by "A-0-2750, ELS, PhasingSlave, additive position command value" can be specified via MB_PhasingSlave or the AxisInterface

Based on the incoming position and velocity signal, the function block can implement the synchronization before master and slave axis engage.

6.6.6 MC_CamOut

Brief description The MC_CamOut function block completes the electronic cam function activated by the MC_CamIn function block.

This block implements two jobs:

- Ending the cam operation of the slave axis,
- It transfers the slave axis to the "Continuous Motion" mode, while maintaining the last velocity value in case there is no other command.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-30: Library assignment

Interface description

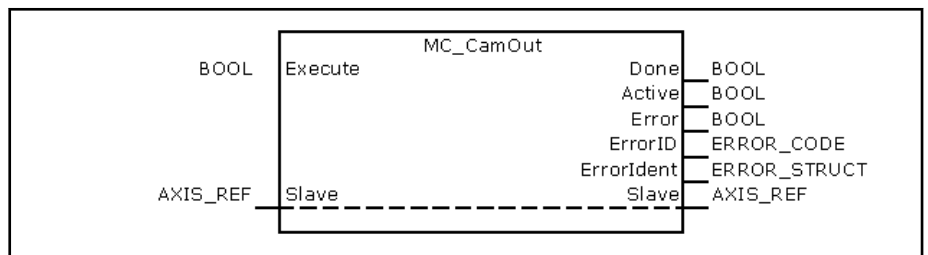


Fig. 6-38: FB MC_CamOut

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
VAR_OUTPUT	Done	BOOL	Decoupling completed successfully
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Slave	AXIS_REF	It contains information on the slave axis which follows the master axis

Tab. 6-31: Interface of the MC_CamOut function block

Signal-time diagram

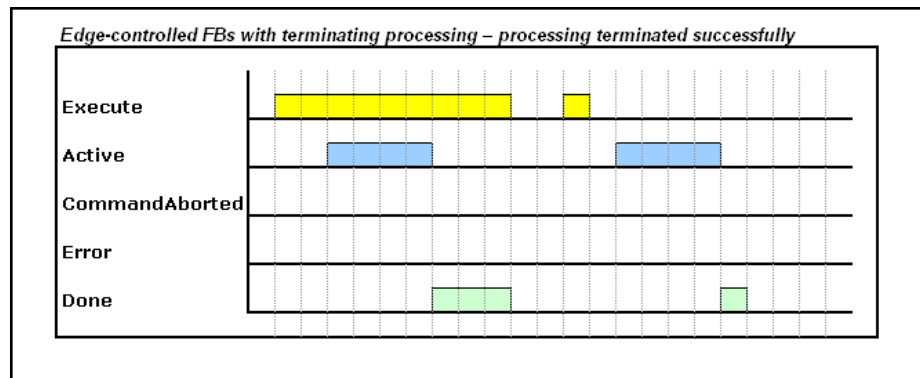


Fig. 6-39: Signal-time diagram

Troubleshooting The function block MC_CamOut generates the following error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.6.7 MB_MotionProfile

Brief description The MB_MotionProfile function block implements an electronic cam function. The cam functions are analytically defined by the parameters belonging to the operation mode "MotionProfile".



If this function block requires power added before via an instance of the [MC_Power, page 179](#), (for real axes).



The function block is deactivated by the block [MC_GearOut, page 247](#).



MotionProfile functionality is supported by the following axis types (slave axis):

- IndraDrive with interpolation in the control
- HydraulicDrive with interpolation in the control
- SercosDrive
- Virtual axis
- Controller axis

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Motion

Tab. 6-32: Library assignment

Interface description

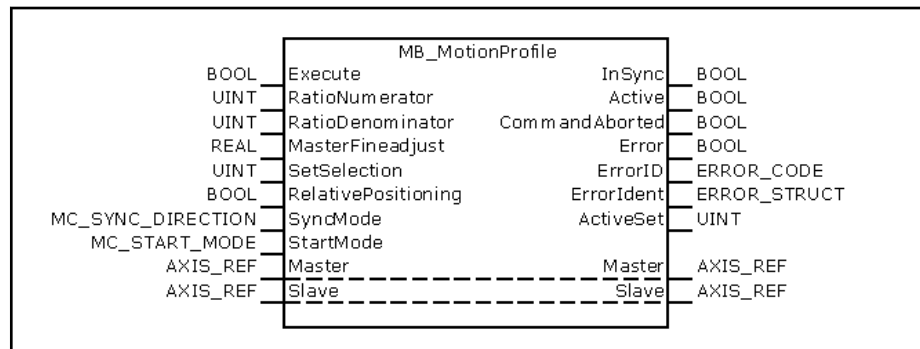


Fig. 6-40: MB_MotionProfile function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
	RatioNumerator	UINT	Gear ratio - numerator "master axis gear output revolutions" Min: 1; max: 65535; default: 1
	RatioDenominator	UINT	Gear ratio - denominator "master axis gear input revolutions" Min: 1; max: 65535; default: 1
	MasterFineadjust	REAL	"Fine adjustment of gear ratio" in %; default: 0.0, min.: -799,9999%, max.: 799,9999
	SetSelection	UINT	Selection of MotionProfile (0 or 1)
	RelativePositioning	BOOL	RelativePositioning determines whether the phase offset of the parameter A-0-2901 should be active. 0: Phase offset not active (default) 1: Offset phase active
	SyncMode	MC_SYNC_DIRECTION	Synchronization direction: 0: = Shortest distance; 1: = Positive direction; 2: = Negative direction
	StartMode	MC_START_MODE	Synchronization type: 0:= Absolute synchronization, without synchronization in case of a new block call 1:= Relative synchronization, without synchronization in case of a new block call 2:= Absolute synchronization, with synchronization in case of a new block call 3:= Relative synchronization, with synchronization in case of a new block call
VAR_OUTPUT	InSync	BOOL	Slave axis is synchronous
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	ActiveSet	UINT	Number of the MotionProfile
VAR_IN_OUT CONSTANT	Master	AXIS_REF	Master axis selection
	Slave	AXIS_REF	Selection of the slave axis following the master axis

Tab. 6-33: Interface of the MB_MotionProfile function block

Functional Description

With the "MotionProfile with relative/virtual master axis", there is a specific relationship between the master axis position (master position) and the slave axis (slave position). The master axis position is determined by a real or a virtual master axis source.

- 2 motion sequences per axis with up to 16 steps can be parameterized.

- A step is determined by a cam table or a standard profile (e.g. polynomial 5th order).
- The standard profiles rest-in-rest, rest-in-velocity, velocity-in-rest, velocity-in-velocity and constant velocity can be selected.
- The realization of the standard profiles is done with a polynomial 5th order or in case of rest-in-rest with a suitable sinus curve.
- If cam tables are required, cam tables with 3 up to a maximum of 1,024 data points can be selected.

The cam tables are stored on the control under the C-parameters C-0-2001...2099. The [MB_ChangeCamData](#), page 225, function block allows filling these parameters with PLC data.

- It is possible to select cubic spline interpolation or linear interpolation between such cam table data points.
- Each step has its own distance.
- It can be selected between an absolute or relative processing of the steps.
- An electronic gear with fine adjustment gear ratio is available.
- Additional process controller parameters such as position target value additive, master axis position additive (with 1st order filter), phase offset beginning of table (with 1st order filter) and fine adjustment gear ratio can be selected within the MotionControl functional chain.
- The synchronization mode can be selected.
- The polarity of the master axis can be set.

The following image shows a motion sequence consisting of 5 steps.

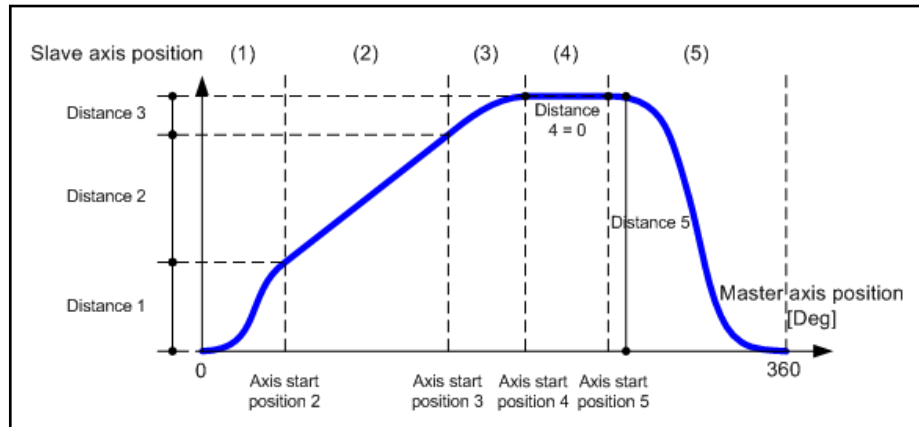


Fig. 6-41: Example of a MotionProfile

Motion laws used

- Rest-in-velocity: Motion step ①,
- Velocity-in-velocity: Motion step ②,
- Velocity-in-rest: Motion step ③,
- Velocity-in-velocity: Motion step ④,
- Rest-in-rest: Motion step ⑤.

Troubleshooting

The function block MB_MotionProfile generates the following error messages:

- [MLC_TABLE](#), page 360 - error when accessing the axis/control.

6.6.8 MB_MotionProfile, special features of Sercos-based axes

Functional description



MotionProfile functionality is supported by the following axis types (slave axis):

- IndraDrive with interpolation in the control
- HydraulicDrive with interpolation in the control
- SercosDrive
- Virtual axis
- Controller axis

The MB_MotionProfile function block implements two jobs:

- Defining how the MotionProfile saved on the control has to be interpreted or processed,
- Giving the order to start and edit the MotionProfile in the process. Once the function block is processed successfully, the axis remains in the MotionProfile editing mode until it receives another command or an error occurs.

The objective is to implement the dependency stored in the MotionProfile between the master axis (A-0-2700, ELS, master axis, control number, A-0-2701, ELS, master axis, axis number) and the selected slave axis.

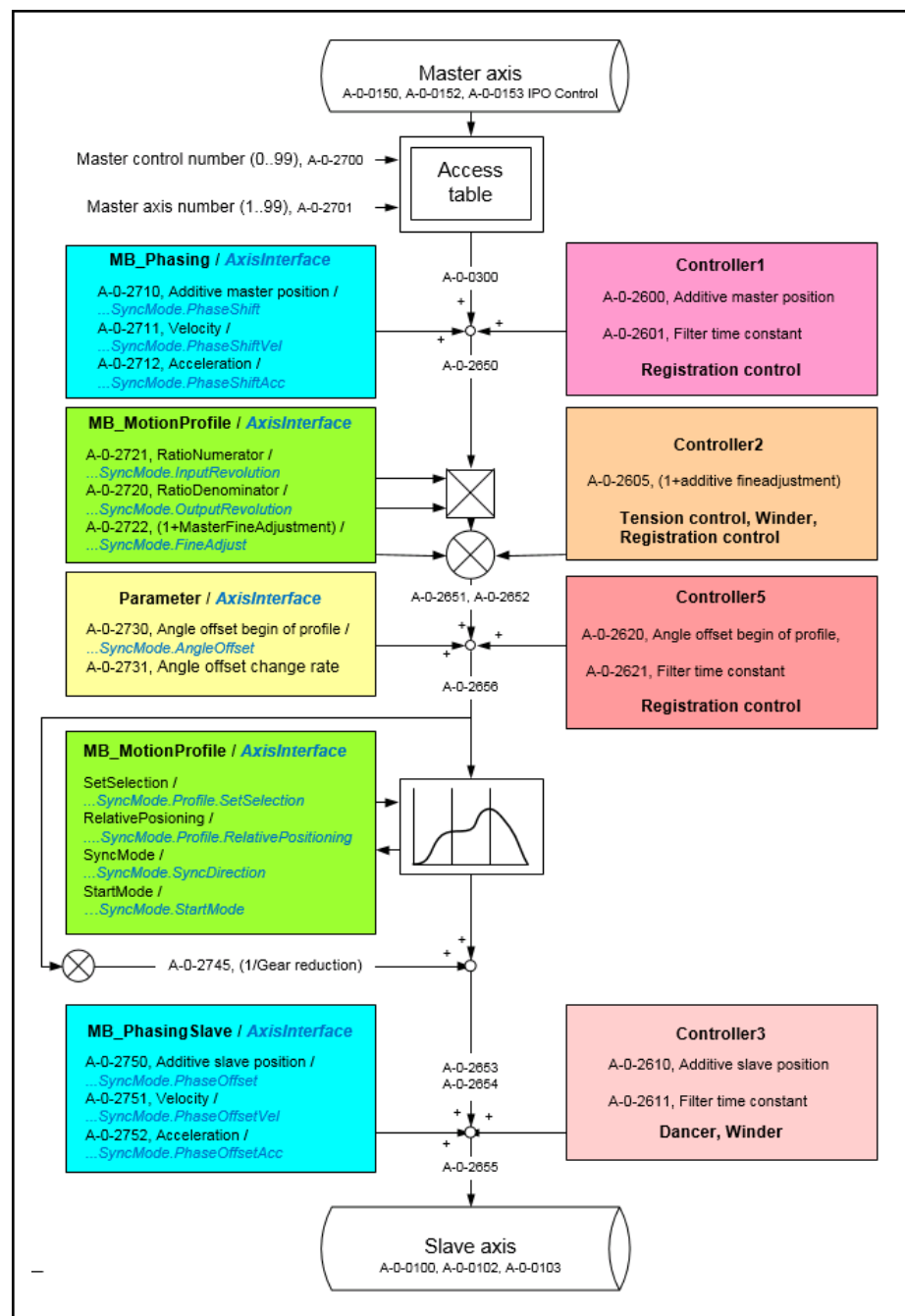


Fig. 6-42: MotionControl action chain MB_MotionProfile

The function block contains the following functional groups:

- The electronic gear which is defined by the ratio of A-0-2721, ELS, master axis gear, output revolutions and A-0-2720, ELS, master axis gear, input revolutions and
- A-0-2722, ELS, gear ratio fine adjust. The fine adjustment value is specified in percent and calculated as $(1 + \text{fine adjustment})$.
- The selection of one of the two possible MotionProfiles, which previously have to be saved in the MLC:

Parameters used together for both MotionProfiles (0 and 1) for configuration:

- A-0-2900, MotionProfile, master axis switching position
- A-0-2901, MotionProfile, motion step 1, initial slave axis position
- A-0-2904, MotionProfile, basic configuration

Axis parameters, MotionProfile 0:

- A-0-2902, MotionProfile 0, diagnostics
- A-0-2910, MotionProfile 0, number of motion steps
- A-0-2911, MotionProfile 0, master axis velocity
- A-0-2912, MotionProfile 0, list of initial master axis positions
- A-0-2913, MotionProfile 0, list of motion laws
- A-0-2914, MotionProfile 0, list of distances
- A-0-2915, MotionProfile 0, list of slave axis velocities

Axis parameters, MotionProfile 1:

- A-0-2903, MotionProfile 1, diagnostics
- A-0-2920, MotionProfile 1, number of motion steps
- A-0-2921, MotionProfile 1, master axis velocity
- A-0-2922, MotionProfile 1, list of initial master axis positions
- A-0-2923, MotionProfile 1, list of motion laws
- A-0-2924, MotionProfile 1, list of distances
- A-0-2925, MotionProfile 1, list of slave axis velocities

Parameters used together for both MotionProfiles (0 and 1) in the running operation:

- A-0-2930, MotionProfile, control word
- A-0-2931, MotionProfile, status word

Based on the incoming position and velocity signal, the function block can implement the synchronization before master and slave axis engage.

6.6.9 PLCopen function blocks to support MotionProfiles

General information

An MLC can manage a maximum of 64 axes⁹⁾ control.

Two MotionProfiles can be assigned to one axis, of which one can be activated. If necessary, it is possible to switch to the other MotionProfile.

Every MotionProfile can have up a maximum of 16 steps.

The individual motion steps can either be defined with certain motion laws (VDI 2143 and special motion laws) or with referenced cam tables.

For receiving the cam tables, the control provides storage area for a maximum of 99 cam tables with a maximum of 1,024 point each in its Compact Flash (CF). The cam tables can be addressed via the parameters C-0-2001 ... 2099.

⁹⁾ depending on the type of

The cam tables can be configured either via the PLC function block [MB_ChangeCamData](#), [page 225](#), or via the IndraWorks interface.



MotionProfile functionality is supported by the following axis types (slave axis):

- IndraDrive with interpolation in the control
- HydraulicDrive with interpolation in the control
- SercosDrive
- Virtual axis
- Controller axis

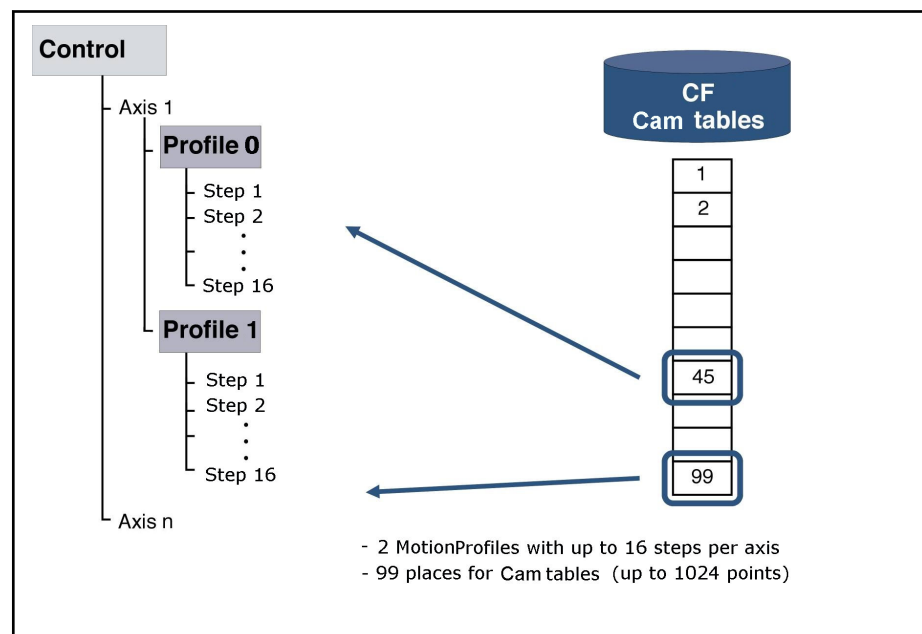


Fig. 6-43: Assigning cam tables/MotionProfile

- MC_STEP_MODE, RIL_CommonTypes.library: Enumeration data type describing the functional relationship between the slave axis and the master axis for a step of the MotionProfile or FlexProfile.
- [MB_ChangeProfileSet](#), [page 218](#): Writing a complete MotionProfile, from the PLC to the Motion
- [MB_ChangeProfileStep](#), [page 222](#): Replacing a step of the MotionProfile,
- [MB_ChangeCamData](#), [page 225](#): Transferring a cam table from the PLC memory to the Motion, C-0-2001...2099.

MB_ChangeProfileSet

Brief description

The function block MC_ChangeProfileSet allows to write on a complete MotionProfile with up to 16 steps or to exchange an existing one.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Motion

Tab. 6-34: Library assignment

Interface description

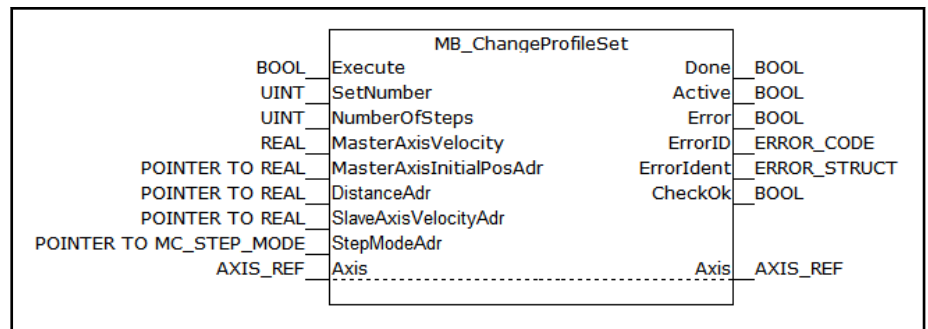


Fig. 6-44: FB MB_ChangeProfileSet

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
	SetNumber	UINT	Selection of MotionProfile (0 or 1)
	Number of Steps	UINT	Number of steps of the MotionProfile (1...16)
	MasterAxis-Ve- locity	REAL	Master axis velocity for the profile
	MasterAxis-Ini- tialPosAdr	POINTER TO REAL	Pointer to a list of master axis positions at the beginning of the re- spective step
	DistanceAdr	POINTER TO REAL	Pointer to a list of strokes at the end of the respective step
	SlaveAxis-Ve- locityAdr	POINTER TO REAL	Pointer to a list of slave axis velocities at the end of the respec- tive step
	StepModeAdr	POINTER TO MC_STEP_MODE	Pointer to a list of motion laws for the steps
VAR_OUTPUT	Done	BOOL	Data were accepted.
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	CheckOk	BOOL	Reserved
	Axis	AXIS_REF	Contains information on the axis for which the MotionProfile set is to be exchanged.

Tab. 6-35: Interface of the MB_ChangeProfileSet function block

Signal-time diagram

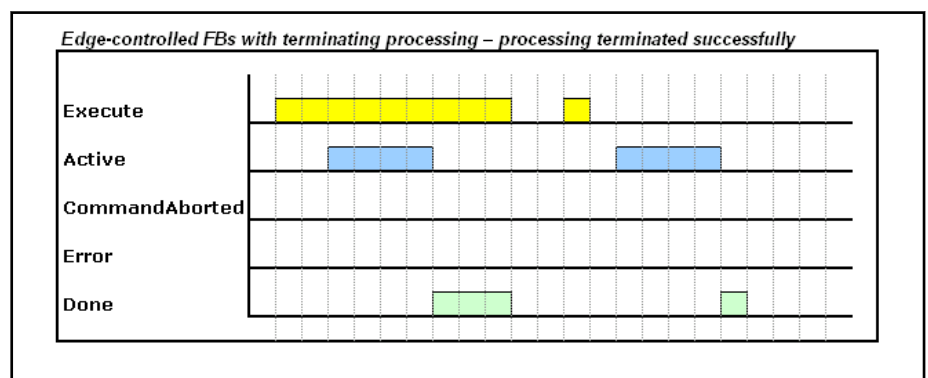


Fig. 6-45: Signal-time diagram: MB_ChangeProfileSet

Functional description The function block allows a complete MotionProfile with 1...16 steps to be transferred from the PLC to the Motion component of the control.

The data characterizing the respective steps are saved in the list the beginning of which has to be described in the PLC storage space with a pointer (of REAL).

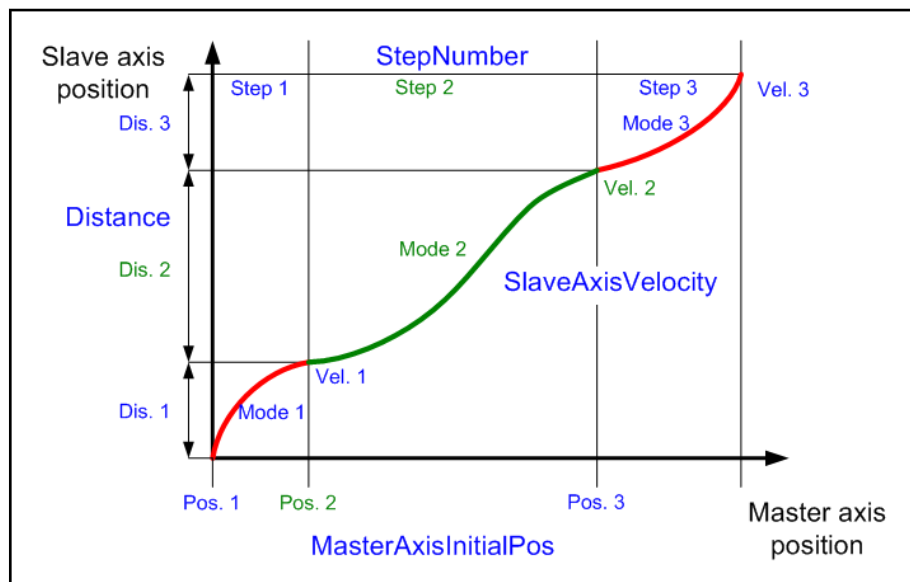


Fig. 6-46: Data sets of the individual steps of a MotionProfile

The modes of the respective steps can be taken from the enumeration type "MC_STEP_MODE" from the RIL_CommonTypes.library:



The motion mode (see MC_STEP_MODE) is an enumeration type, which in addition to functions such as

- REST_IN_REST_POLY5
- REST_IN_REST_INCLINEDSINE
- REST_IN_VELOCITY_POLY5
- VELOCITY_IN_VELOCITY_POLY5 and
- CONSTANT_VELOCITY

also allows for the selection of the CAM_TABLE_1 ... CAM_TABLE_99.

MB_ChangeProfileSet, special features of Sercos-based axes

It is possible to set presettings for the respectively changed data set in the parameter A-0-2930, MotionProfile, control word.

Bit 11:

This bit defines whether the calculation for a cam table takes place between the data points of linear or spline interpolation. The state should no longer be modified during processing.

Bit 12:

This bit defines the processing mode for a parameter modification.

- Bit 12 = 0: Parameter collection is not active. Every parameter modification is immediately tested and, if the profile is free of errors, accepted.
- Bit 12 = 1: Parameter collection is active, every parameter modification is saved. The modification is accepted in the internal work data and the modified parameters are tested only after bit 12 has been reset.

Bit 13:

This bit determines whether a cam is write-protected during processing. The write-protection is valid in the active step. Cams in inactive steps can be written. The state should no longer be modified during processing.



The test runs nearly automatically when bit 12 = 0 is preset.

The test runs after "Done" becomes active, i.e. the test result has to be taken from the parameter "A-0-2931, MotionProfile, status word".

**Setting Operation Mode "Test",
A-0-2930, bit 12 = 0**

- Reading parameter A-0-2930, MotionProfile, control word using MB_ReadParameter (value as DINT),
- DINT_TO_WORD,
- AND 16#EFFF, reset bit 12, initiation of test procedure, WORD_TO_DINT,
- Writing parameter A-0-2930, MotionProfile, control word using MB_WriteParameter (value as DINT).

**Read status to control whether the
complete profile was tested.**

- Reading parameter A-0-2931, MotionProfile, status word using MB_ReadParameter (value as DINT),
- DINT_TO_DWORD,
- **Bit 12/13:**

Bit 12 indicates whether the MotionProfile was checked in set 0.

Subsequently, the result of the test has to be taken from bit 13. For more information on a negative test result "F2229235, Error in MotionProfile", refer to A-0-2902, MotionProfile 0, diagnostics.

Bit 14/15:

Bit 14 indicates whether the MotionProfile was checked in set 1.

Subsequently, the result of the test has to be taken from bit 15. For more information on a negative test result "F2229235, Error in MotionProfile", refer to A-0-2903, MotionProfile 1, diagnostics.

Parameter assignment

Parameters used jointly for both MotionProfiles (0 and 1) for configuration:

- A-0-2900, MotionProfile, master axis switching position
- A-0-2901, MotionProfile, motion step 1, initial slave axis position
- A-0-2902, MotionProfile 0, diagnostics
- A-0-2903, MotionProfile 1, diagnostics
- A-0-2904, MotionProfile, basic configuration

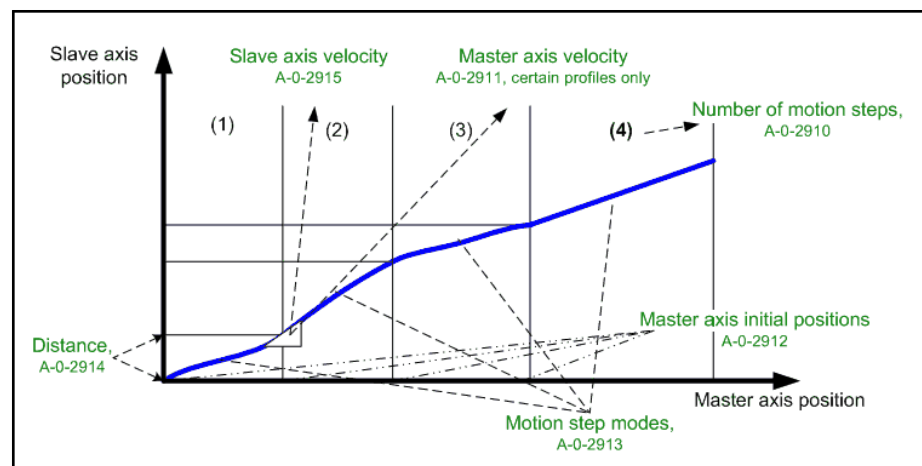


Fig. 6-47: Overview on the parameters of the MotionProfile 0
Axis parameters, MotionProfile 0

- A-0-2910, MotionProfile 0, number of motion steps
- A-0-2911, MotionProfile 0, master axis velocity
- A-0-2912, MotionProfile 0, list of initial master axis positions
- A-0-2913, MotionProfile 0, list of motion laws
- A-0-2914, MotionProfile 0, list of distances
- A-0-2915, MotionProfile 0, list of slave axis velocities

Axis parameters, MotionProfile 1

- A-0-2920, MotionProfile 1, number of motion steps
- A-0-2921, MotionProfile 1, master axis velocity
- A-0-2922, MotionProfile 1, list of initial master axis positions
- A-0-2923, MotionProfile 1, list of motion laws
- A-0-2924, MotionProfile 1, list of distances
- A-0-2925, MotionProfile 1, list of slave axis velocities

Control and status words used jointly for both MotionProfiles (0 and 1):

- A-0-2930, MotionProfile, control word
- A-0-2931, MotionProfile, status word

Troubleshooting

The function block MB_ChangeProfileSet generates the following types of error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

MB_ChangeProfileStep

Brief description

The function block MB_ChangeProfileStep changes the four parameters which define one step of a MotionProfile set.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Motion

Tab. 6-36: Library assignment

Interface description

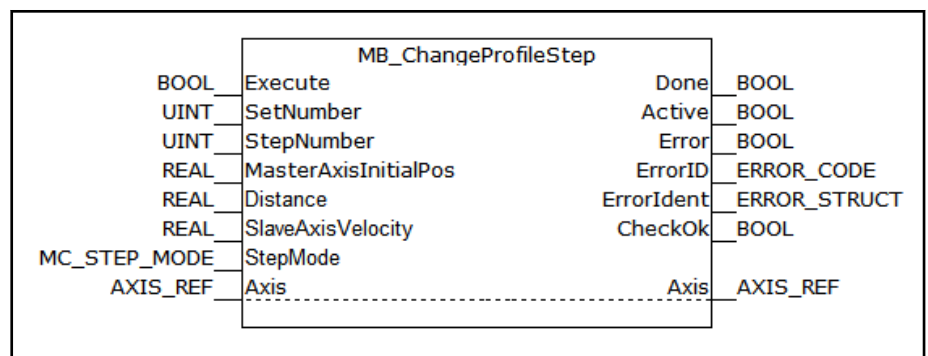


Fig. 6-48: FB MB_ChangeProfileStep

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
	SetNumber	UINT	Selection of MotionProfile (0 or 1)
	StepNumber	UINT	Selection of the motion step to be modified (1...16)
	MasterAxis-InitialPos	REAL	Initial master axis positions of the step in degrees
	Distance	REAL	Slave axis path for the step
	SlaveAxis-Velocity	REAL	Slave axis velocity within the step
	StepMode	MC_STEP_MODE	Step type
VAR_OUTPUT	Done	BOOL	Data were accepted.
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	CheckOk	BOOL	Reserved
	Axis	AXIS_REF	Contains information on the axis for which the MotionProfile set is to be changed.

Tab. 6-37: Interface of the MB_ChangeProfileStep function block

Signal-time diagram

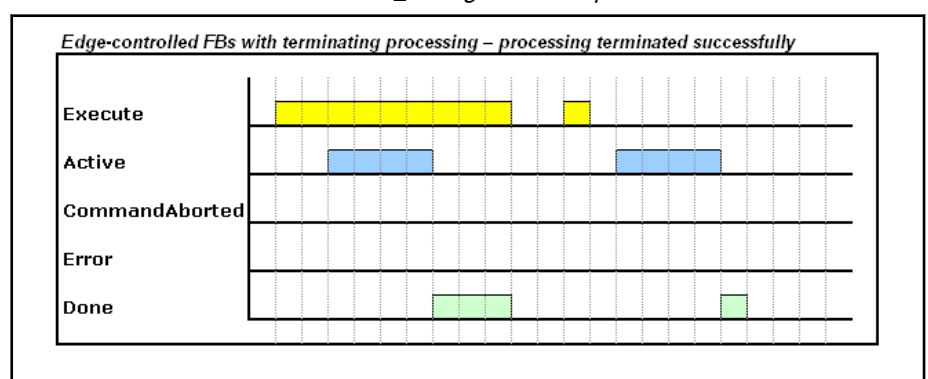


Fig. 6-49: Signal-time diagram: MB_ChangeProfileStep

Functional description

The function block replaces one step with another.

The input values of the function block are the number of the MotionProfile (SetNumber), 0 or 1, the number of the step (StepNumber), 1...16, i.e. the location of the step to be replaced, and the data of the new step master axis initial position (MasterAxisInitialPosition), distance and slave axis velocity (SlaveAxisVelocity) at the end of the step as well as the motion step mode (function or cam table) can be selected.



The motion mode (see MC_STEP_MODE) is an enumeration type, which in addition to functions such as

- REST_IN_REST_INCLINEDSINE
- REST_IN_REST_POLY5
- REST_IN_VELOCITY_POLY5
- VELOCITY_IN_VELOCITY_POLY5 and
- CONSTANT_VELOCITY

also allows for the selection of the CAM_TABLE_1 ... CAM_TABLE_99.

The units for the block inputs Distance and SlaveAxisVelocity are defined by the scaling parameters of the axis.

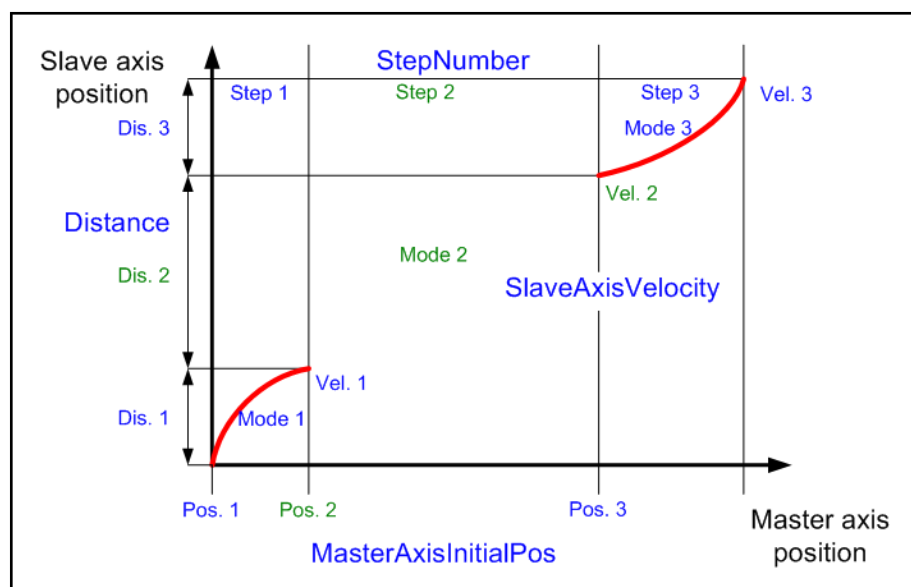


Fig. 6-50: MotionProfile with 3 steps, step 2 is modified

Troubleshooting

The function block MB_ChangeProfileStep generates the following types of error messages:

- [MLC_TABLE](#), page 360 - error when accessing the axis/control.

MB_ChangeProfileStep, special features of Sercos-based axes

Plausibility check and collection

Every writing procedure on a MotionProfile initiates a plausibility check regarding the accuracy of the new step.

Theoretically, it is possible to write a step fulfilling the transition conditions into the active MotionProfile. However, this is only possible if no motion command is carried out with this set.

If several steps are exchanged in the MotionProfile, it is possible to first block the plausibility check in order to avoid error messages and to only initiate it when all the desired steps are exchanged (collection).

Example of the use of the MB_ChangeProfileStep for two (or more) steps in the profile

The decision to collect or to test immediately can be made individually for every axis in the parameter A-0-2930, MotionProfile, control word, bit 12.

The following describes the working procedure for using "MB_ChangeProfileStep" in the example of two exchangeable steps:

1. Setting operation mode "Collection"
 - Reading parameter A-0-2930, MotionProfile, control word using MB_ReadParameter (value as DINT),
 - DINT_TO_WORD,
 - OR 16#1000, set bit 12, collect, WORD_TO_DINT,
 - Writing parameter A-0-2930, MotionProfile, control word using MB_WriteParameter (value as DINT).
2. Exchange of the first step
 - MB_ChangeProfileStep for the first data set (carried out without test).
The status change is applied to parameter A-0-2931, MotionProfile, status word.
3. Exchange of the second step
 - MB_ChangeProfileStep for the second data set (carried out without test).
4. Setting operation mode "Test"
 - Reading parameter A-0-2930, MotionProfile, control word using MB_ReadParameter (value as DINT),
 - DINT_TO_WORD,
 - AND 16#EFFF, reset bit 12, initiation of test procedure, WORD_TO_DINT,
 - Writing parameter A-0-2930, MotionProfile, control word using MB_WriteParameter (value as DINT).
5. Read status to control whether the complete profile was tested.
 - Reading parameter A-0-2931, MotionProfile, status word using MB_ReadParameter (value as DINT),
 - DINT_TO_DWORD,
 - **Bit 12/13:**
Bit 12 indicates whether the MotionProfile was checked in set 0.
Subsequently, the result of the test has to be taken from bit 13. For more information on a negative test result "F2229235, Error in MotionProfile", refer to A-0-2902, MotionProfile 0, diagnostics.
 - **Bit 14/15:**
Bit 14 indicates whether the MotionProfile was checked in set 1.
Subsequently, the result of the test has to be taken from bit 15. For more information on a negative test result "F2229235, Error in MotionProfile", refer to A-0-2903, MotionProfile 1, diagnostics.

MB_ChangeCamData

Brief description

The function block MB_ChangeCamData supports loading cams under real-time conditions saved or edited on the PLC to the cam table parameters C-0-2001 ... 2099 (Motion).

The function block transports "NumberOfElements * 4 byte" data, beginning with the location in the PLC memory to which the "CamTableAdr" pointer points, to the cam table selected with "CamTableID" (RAM, C-parameters).

Done indicates that the copy procedure is successfully completed.



The individual data to be transferred may be in the range -799,999,999 to +799,999,999.



The function block does not carry out a check of the content of the data to be transferred, so that the user has to ensure that the information to be transferred are of a correct quantity of data and the right type of data (DINT).

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Motion

Tab. 6-38: Library assignment

Interface description

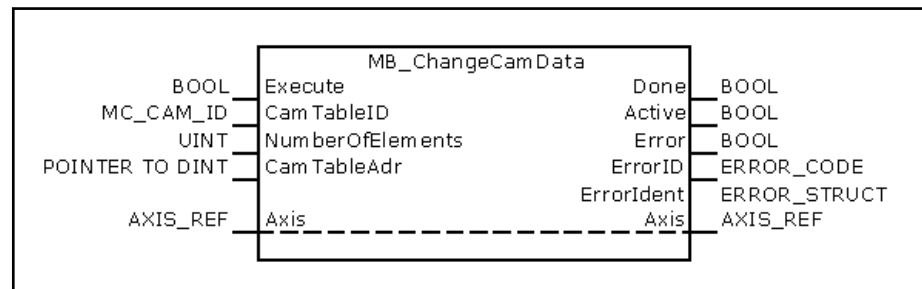


Fig. 6-51: Function block MB_ChangeCamData

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
	CamTableID	MC_CAM_ID	Number of the cam table to be written 1..99 (C-0-2001...2099).
	NumberOfElements	UINT	Number of data points 3...1,024 in the desired cam table
	CamTableAdr	POINTER TO DINT	Pointer points to the PLC storage area containing the data to be transported.
VAR_OUTPUT	Done	BOOL	Cam table data has been transferred
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the slave axis

Tab. 6-39: Interface of the MB_ChangeCamData function block

Signal-time diagram

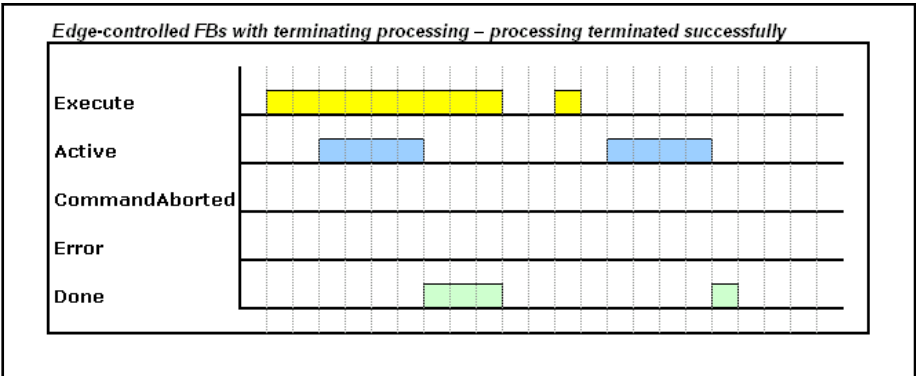


Fig. 6-52: Signal-time diagram: MB_ChangeCamData

- Troubleshooting
- The function block MB_ChangeCamData generates the following types of error messages:
- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.6.10 ML_FlexProfile

Brief description The "ML_FlexProfile" function block implements an electronic cam table (FlexProfile).
The FlexProfiles are analytically defined by the parameters belonging to the operation mode "Electronic MotionProfile".

If this function block requires power added before via an instance of the [MC_Power, page 179](#), (for real axes).

The function block is deactivated by the block [MC_GearOut, page 247](#),.

- The FlexProfile functionality is supported by the following axis types (slave axis):*
- IndraDrive with interpolation in the control
 - HydraulicDrive with interpolation in the control
 - SercosDrive
 - Virtual axis
 - Controller axis

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-40: Library assignment

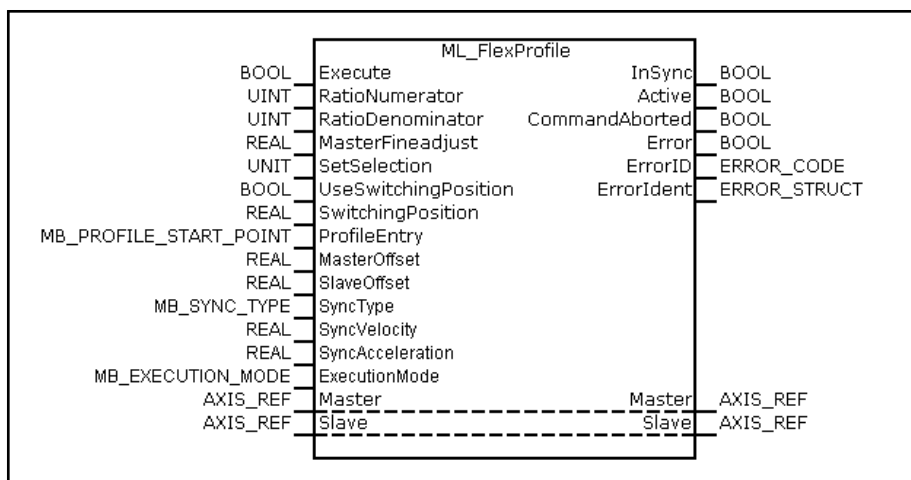


Fig. 6-53: ML_FlexProfile function block

Interface description

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
	RatioNumerator	UINT	Gear ratio - numerator "master axis gear output revolutions" Min: 1; max: 65535; default: 1
	RatioDenominator	UINT	Gear ratio - denominator "master axis gear input revolutions" Min: 1; max: 65535; default: 1
	MasterFineadjust	REAL	"Fine adjustment of gear ratio" in %; default: 0.0, min.: -799.9999%, max.: 799.9999
	SetSelection	UINT	Selecting the FlexProfile (0 ...3)
	UseSwitchingPosition	BOOL	UseSwitchingPosition determines whether the switching position or the initial position has to be activated (taken into consideration). If TRUE, the motion start and/or the switching to another profile is only carried out when the master axis reaches the position applied at the input "SwitchingPosition".
	SwitchingPosition	REAL	Switching position or initial position where the switching to another profile or the profile start should take place.
	ProfileEntry	MB_PROFILE_START_POINT	Describes how the origin of the profile has to be interpreted with a profile start or change of profile. 0 - Absolute access to the profile 1 - Profile starts at the current master and slave axis position 2 - Profile starts at the current master axis position and at the absolute slave axis position 3 - Profile starts at the current slave axis position and at the absolute master axis position 4 - Profile starts at the current master axis position and the absolute slave axis position
	SlaveOffset	REAL	Offset of the slave axis

	Name	Type	Comment
	MasterOffset	REAL	Offset of the master axis
	SyncType	MB_SYNC_TYPE	Synchronization type 0 - Soft switching shortest distance 1 - Soft switching clockwise 2 - Soft switching counterclockwise 3 - Hard switching
	SyncVelocity	REAL	Synchronization velocity
	SyncAcceleration	REAL	Synchronization acceleration
	ExecutionMode	MB_EXECUTION_MODE	Processing type 0 - Cyclic FlexProfile processing 1 - One-time FlexProfile processing
VAR_OUTPUT	InSync	BOOL	Slave axis is synchronous
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by the following command.
	Error	BOOL	Indicates that an error has occurred in the function block instance.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Master	AXIS_REF	Master axis selection
	Slave	AXIS_REF	Selection of the slave axis following the master axis

Tab. 6-41: Interface of the ML_FlexProfile function block

Functional description

With the FlexProfiles with relative/virtual master axis, there is a predetermined relation between the master axis position (master position) and the slave axis position (slave position). The master axis position is determined by a real or a virtual master axis source.

- Four FlexProfiles per axis with up to 64 steps can be parameterized.
- A motion step is specified by a cam table or a standard profile (e.g. polynomial 5th order, polynomial 7th order, sinusoid, etc.).
- In a FlexProfile, up to 16 conditions (response to the position, time, control bit ...) can be defined.
- The profile starting point can be defined with the profile start or the change of profile.
- The master axis can be configured either as a time axis or a position axis.
- The standard profiles rest-in-rest, rest-in-velocity, velocity-in-rest, velocity-in-velocity and constant velocity can be selected.
- If cams are required, cam tables with 3 and up to a maximum of 1,024 data points can be selected.

The cam tables are stored on the control under the C-parameters C-0-2001...2099. The [MB_ChangeCamData](#) , on page 225 function block allows filling these parameters with PLC data.

- Cam tables with up to 100 points are interpolated with cubic splines. Cam tables with more than 100 points are linearly interpolated.

- Each step has its own distance.
- The switching type (SyncType) can be selected, i.e. the profiles can be changed hard or with an automatic synchronization procedure.
- Synchronization velocity and synchronization acceleration are available.
- An electronic gear with fine adjustment gear ratio is available.
- The switching position or the beginning of the FlexProfile can be configured and activated.
- Additional process controller parameters such as position command value additive, master axis position additive (with 1st order filter), phase offset beginning of table (with 1st order filter) and fine adjustment gear ratio can be selected within the MotionControl action chain.

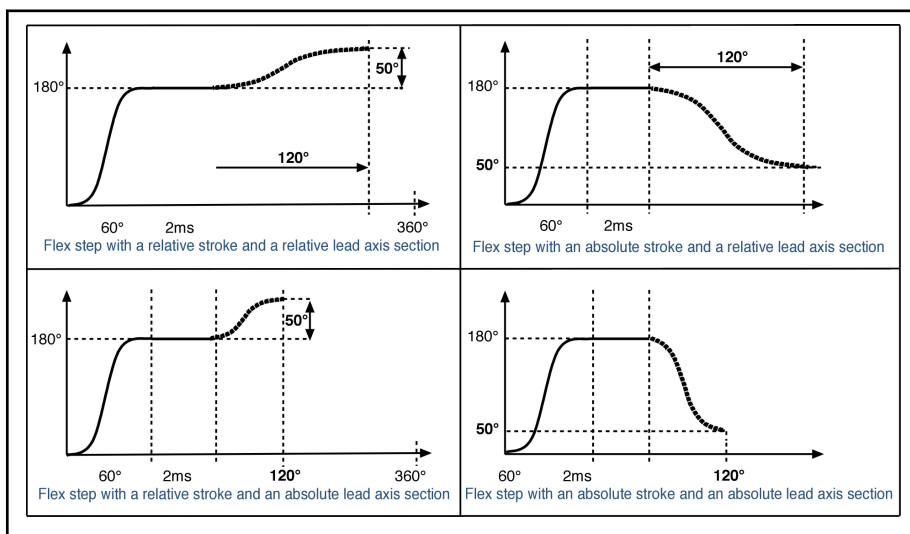


Fig. 6-54: Example of a FlexProfile

Troubleshooting

The function block ML_FlexProfile generates the following error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.6.11 ML_FlexProfile, special features of Sercos-based axes

Functional description



The FlexProfile functionality is supported by the following axis types (slave axis):

- IndraDrive with interpolation in the control
- HydraulicDrive with interpolation in the control
- SercosDrive
- Virtual axis
- Controller axis

The ML_FlexProfile function block implements two jobs:

- Defining how the FlexProfile saved on the control has to be interpreted or processed,
- Giving the order to start and process the FlexProfile in the processing. Once the block is processed successfully, the axis remains in the FlexProfile processing mode until it receives another command or an error occurs.

The objective is to implement the dependency stored in the FlexProfile between the master axis (A-0-2700, ELS, master axis, control number, A-0-2701, ELS, master axis, axis number) and the selected slave axis.

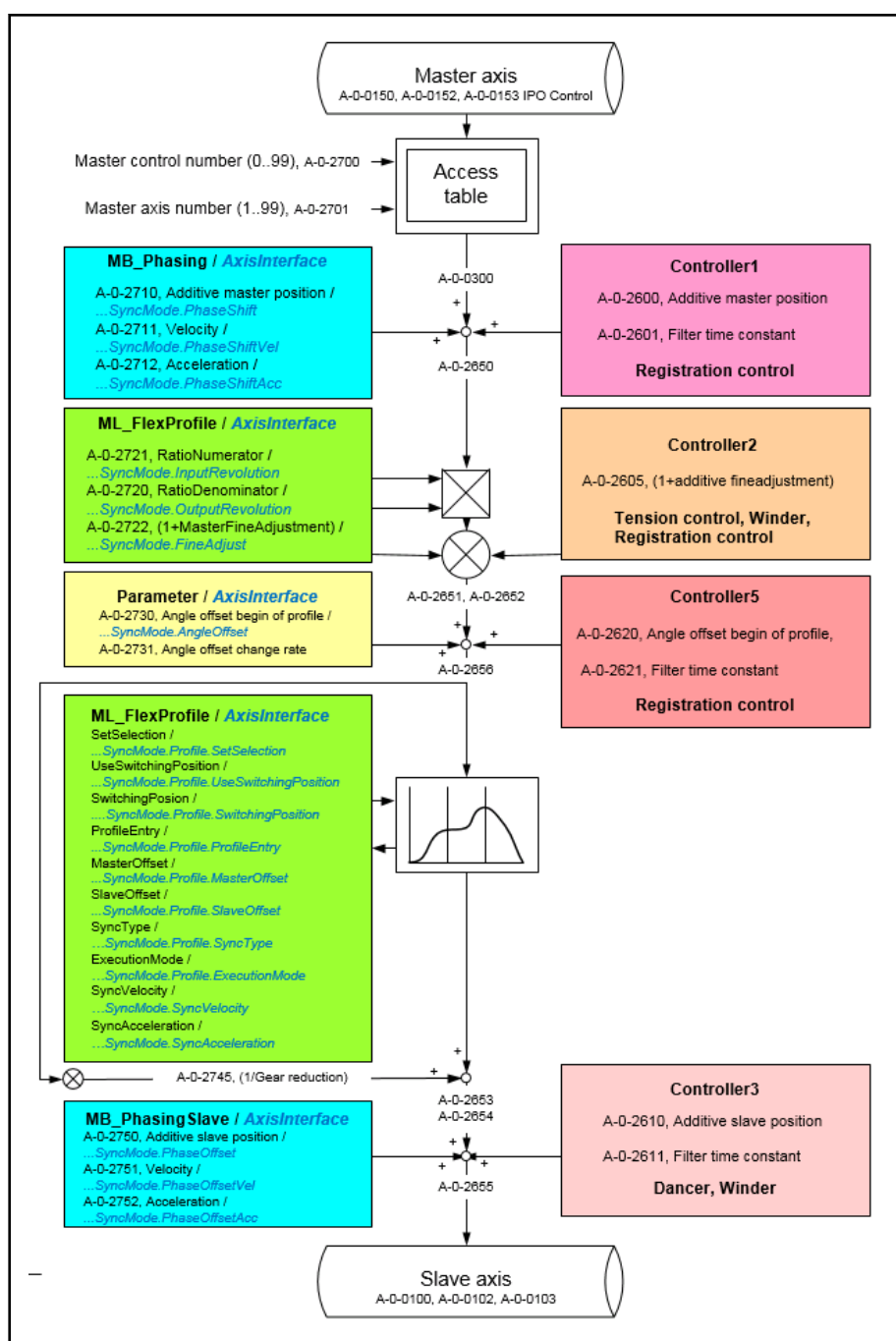


Fig. 6-55: MotionControl action chain ML_FlexProfile

The operation mode provides the following access points:

- The additive master axis position defined by "A-0-2710, ELS, phasing, additive position command value" can be specified via MB_Phasing or the AxisInterface
- Electronic gear defined by the ratio of "A-0-2721, ELS, master axis gear, output revolutions" and "A-0-2720, ELS, master axis gear, input revolutions" and the "A-0-2722, ELS, gear ratio fine adjust". The fine adjustment value is specified in percent and calculated as (1 + fine adjustment)

The selection of one of the four possible FlexProfiles previously saved in the PLC:

These values are specified via ML_Flexprofile or the AxisInterface

- The additive slave axis position defined by "A-0-2750, ELS, PhasingSlave, additive position command value" can be specified via MB_PhasingSlave or the AxisInterface

The selection of one of the four possible FlexProfiles previously saved in the PLC:

- *Common parameters for all four sets*
 - A-0-3000, FlexProfile, current set
 - A-0-3001, FlexProfile, current step
 - A-0-3002, FlexProfile, master axis switching position
 - A-0-3003, FlexProfile, slave axis, additive command position value
 - A-0-3004, FlexProfile, master axis, additive command position value
 - A-0-3009, FlexProfile, control word
 - A-0-3010, FlexProfile, status word
 - A-0-3011, FlexProfile, conditions: control variables
 - A-0-3012, FlexProfile, conditions: status variables
 - A-0-3013, FlexProfile, value, current time-master.

FlexProfile 0-parameters with parameter names	FlexProfile 1	FlexProfile 2	FlexProfile 3
A-0-3020, FlexProfile 0, master axis velocity	A-0-3050	A-0-3080	A-0-3110
A-0-3021, FlexProfile 0, list of master axis sources	A-0-3051	A-0-3081	A-0-3111
A-0-3022, FlexProfile 0, list of slave axis distances	A-0-3052	A-0-3082	A-0-3112
A-0-3023, FlexProfile 0, list of master axis ranges	A-0-3053	A-0-3083	A-0-3113
A-0-3024, FlexProfile 0, list of master axis ranges, units	A-0-3054	A-0-3084	A-0-3114
A-0-3025, FlexProfile 0, list of motion laws	A-0-3055	A-0-3085	A-0-3115
A-0-3026, FlexProfile 0, list of motion step types	A-0-3056	A-0-3086	A-0-3116
A-0-3027, FlexProfile 0, list of slave axis starting velocities	A-0-3057	A-0-3087	A-0-3117
A-0-3028, FlexProfile 0, list of slave axis starting accelerations	A-0-3058	A-0-3088	A-0-3118
A-0-3029, FlexProfile 0, list of slave axis starting jerks	A-0-3059	A-0-3089	A-0-3119
A-0-3030, FlexProfile 0, list of slave axis end velocities	A-0-3060	A-0-3090	A-0-3120
A-0-3031, FlexProfile 0, list of slave axis end accelerations	A-0-3061	A-0-3091	A-0-3121
A-0-3032, FlexProfile 0, list of slave axis end jerks	A-0-3062	A-0-3092	A-0-3122
A-0-3033, FlexProfile 0, list of slave axis travel velocities	A-0-3063	A-0-3093	A-0-3123
A-0-3034, FlexProfile 0, list of slave axis travel accelerations	A-0-3064	A-0-3094	A-0-3124
A-0-3035, FlexProfile 0, list of slave axis travel jerks	A-0-3065	A-0-3095	A-0-3125
A-0-3036, FlexProfile 0, list of turning point displacements	A-0-3066	A-0-3096	A-0-3126
A-0-3040, FlexProfile 0, events, list of motion step assignments	A-0-3070	A-0-3100	A-0-3130
A-0-3041, FlexProfile 0, events, list of trigger modes	A-0-3071	A-0-3101	A-0-3131
A-0-3042, FlexProfile 0, events, list of action modes	A-0-3072	A-0-3102	A-0-3132

FlexProfile 0-parameters with parameter names	FlexProfile 1	FlexProfile 2	FlexProfile 3
A-0-3043, FlexProfile 0, events, list of trigger values	A-0-3073	A-0-3103	A-0-3133
A-0-3044, FlexProfile 0, events, list of trigger values, units	A-0-3074	A-0-3104	A-0-3134
A-0-3046, FlexProfile 0, command: Check and apply event	A-0-3076	A-0-3106	A-0-3136
A-0-3047, FlexProfile 0, diagnostics	A-0-3077	A-0-3107	A-0-3137
A-0-3048, FlexProfile 0, command: Check and apply profile	A-0-3078	A-0-3108	A-0-3138
A-0-3049, FlexProfile 0, checksum	A-0-3079	A-0-3109	A-0-3139

Based on the incoming position and velocity signal, the function block can implement the synchronization before master and slave axis engage.

6.6.12 Switching and Synchronizing the FlexProfile

Selecting the Coordinate System

The classification of the different profiles in the coordinate system can be relative or absolute with regard to the respective master axis and slave axis. This results in the option to combine the admissible setting possibilities arbitrarily (also refer to MB_PROFILE_START_POINT, ML_Base.library,).

The red line symbolizes the current profile and the blue line symbolizes the profile to which it should be switched.

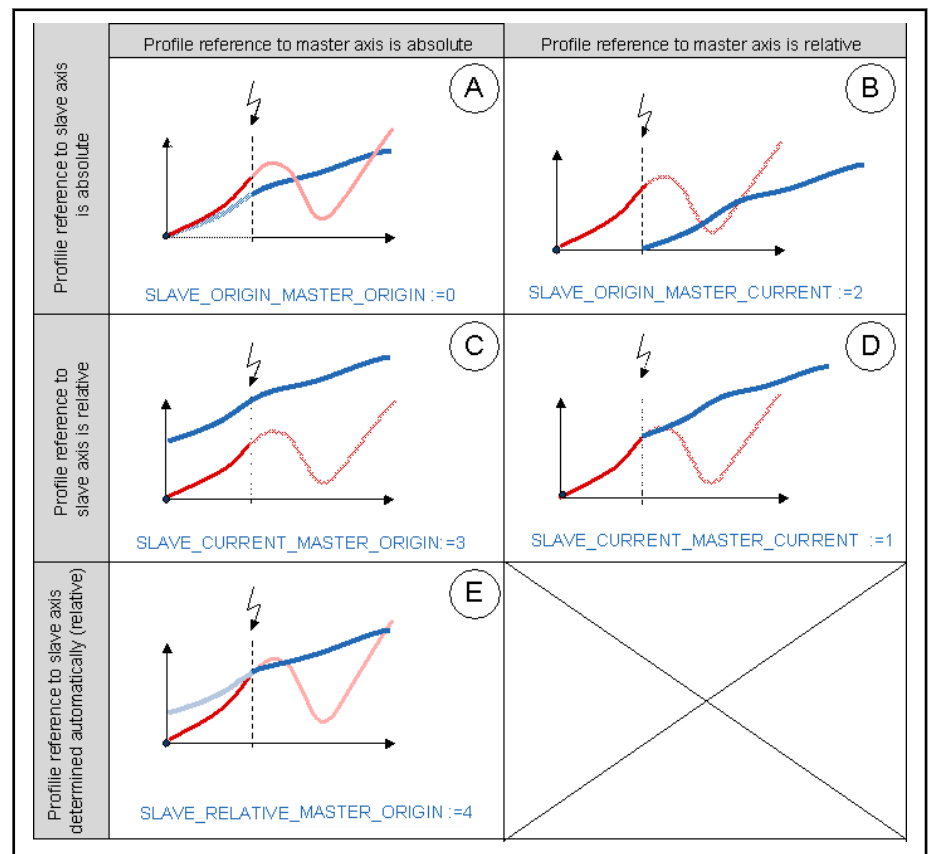


Fig. 6-56: Selecting the coordinate system

Switching and Synchronizing

When the assignment of profiles is defined, the switching type (switching to phase or immediately) as well as the synchronization type (with or without

synchronization) can be selected (also refer to MB_EXECUTION_MODE, ML_Base.library).

When switching to phase, wait after the enable signal till the specified phase is reached and at exactly this point, switching to another state is executed. In case of a decoupled state, the new profile is started instead. Immediate switching is executed simultaneously with the input of a positive edge. When using this functionality, note the congruence of the FlexProfiles. If this is ignored, command value jumps can occur in case of some configurations.

The following figures present some possible combinations for switching and synchronizing together with arbitrary coordinate systems.

The figure below shows the switching to another MotionProfile. The origin of the first and second MotionProfile is congruent and is located in the zero point of the coordinate system (absolute master axis position and absolute slave axis position - case A). The MotionProfiles are not moved.

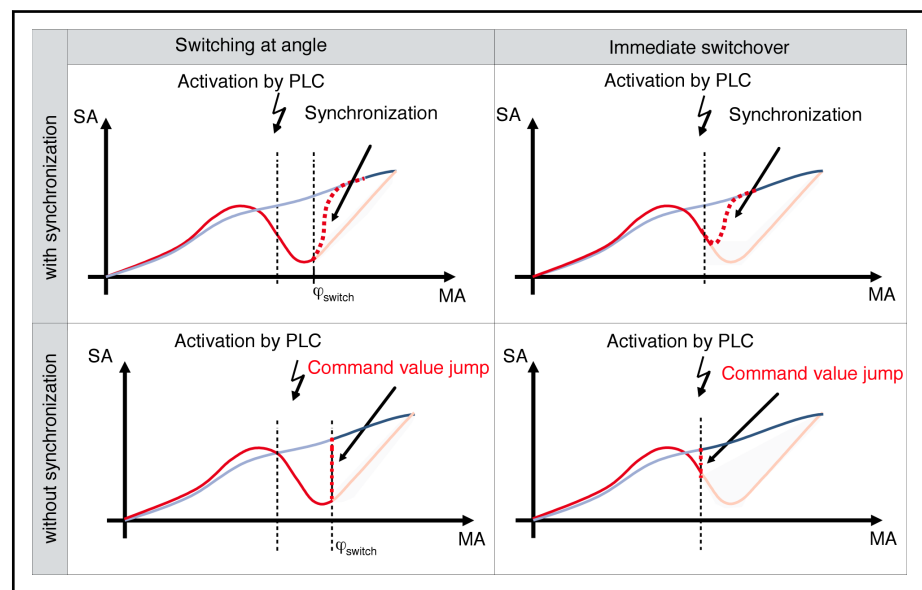


Fig. 6-57: Case A - Switching and synchronizing with absolute master axis position and absolute slave axis position

The figure below shows the switching to another MotionProfile. The origin of the second MotionProfile is moved to the current master and slave axis position of the first MotionProfile (relative master axis position and relative slave axis position - case D).

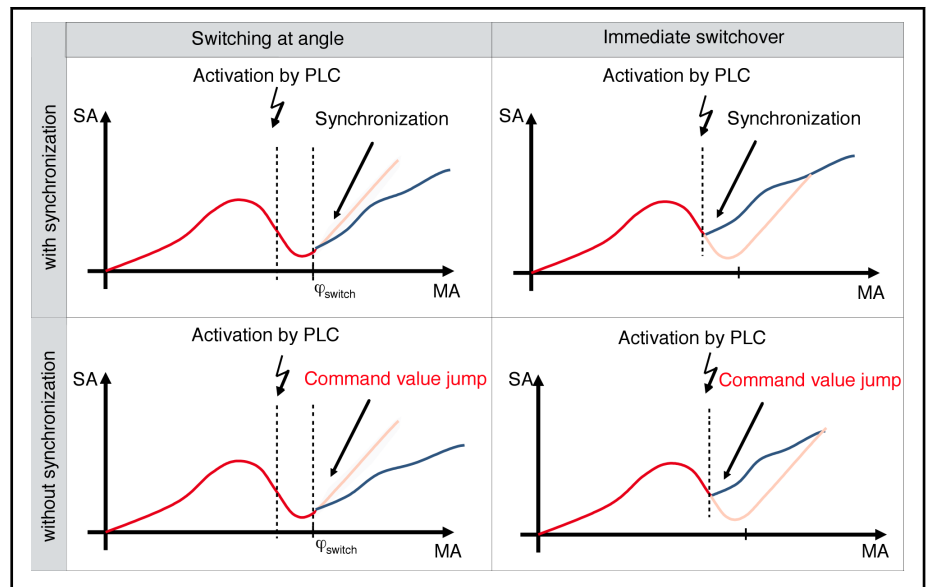


Fig. 6-58: Case D - Switching and synchronizing with relative master axis position and relative slave axis position -

The figure below shows the switching to another MotionProfile. The origin of the second MotionProfile is moved to the current master axis position of the first MotionProfile (relative master axis position and absolute slave axis position - case B). The MotionProfile is not moved towards the slave axis.

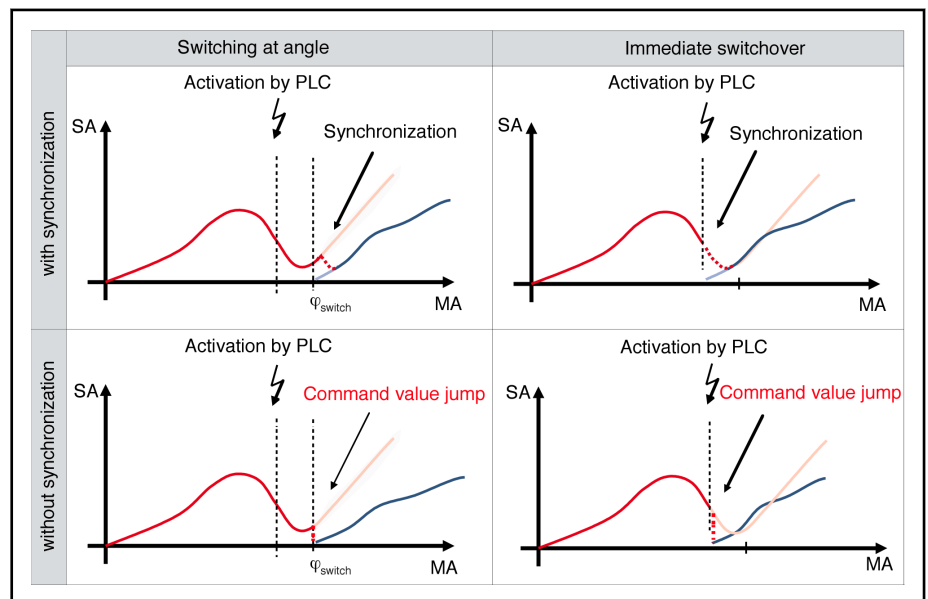


Fig. 6-59: Case B - Switching and synchronizing with relative master axis position and absolute slave axis position

The figure below shows the switching to another MotionProfile. The origin of the second MotionProfile is displaced to the current slave axis position of the first MotionProfile (absolute master axis position and relative slave axis position - case C). The MotionProfile is not moved towards the master axis.

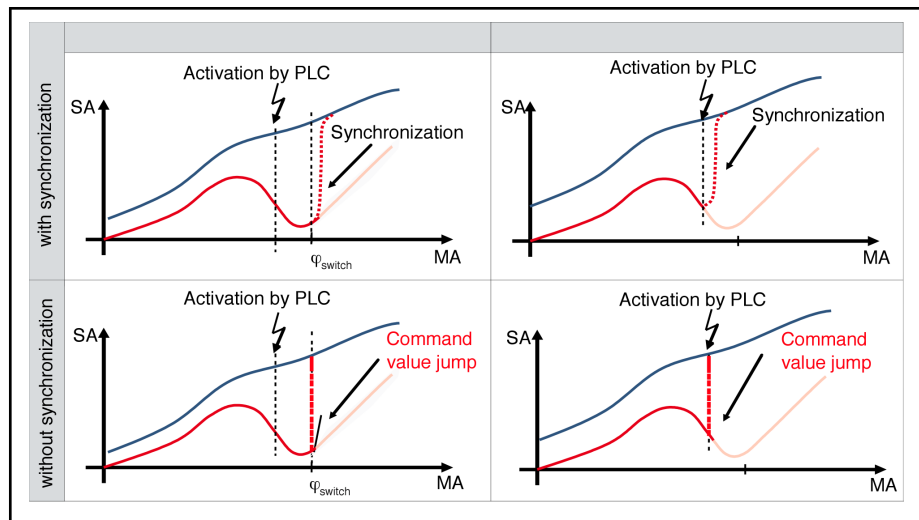


Fig. 6-60: Case C - Switching and synchronizing with absolute master axis position and relative slave axis position

The figure below shows the switching to another MotionProfile. If a slave axis phase offset is not significant and if the slave axis is not supposed to move during synchronization with a master axis at standstill, mode E is appropriate. The profile entry is relative. In case of a master axis at standstill, the slave axis would not carry out any synchronizing motion. When the master axis starts, the slave axis follows the master axis as specified in the MotionProfile.

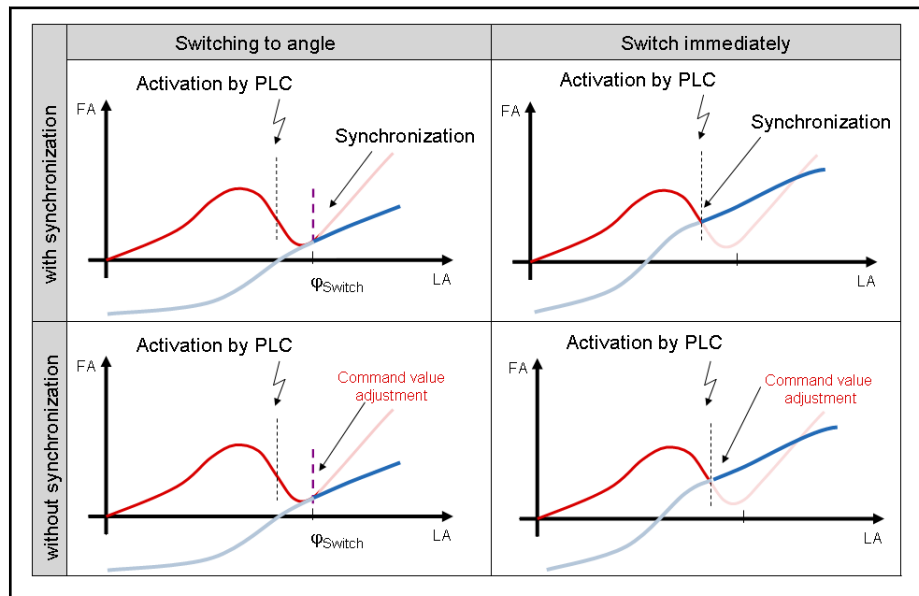


Fig. 6-61: Case E - Profile reference to slave axis is determined automatically (relative)

Processing Types

There are two processing types (see also MB_EXECUTION_MODE, ML_Base.library):

- Continuous processing
- One-time processing (one shot)

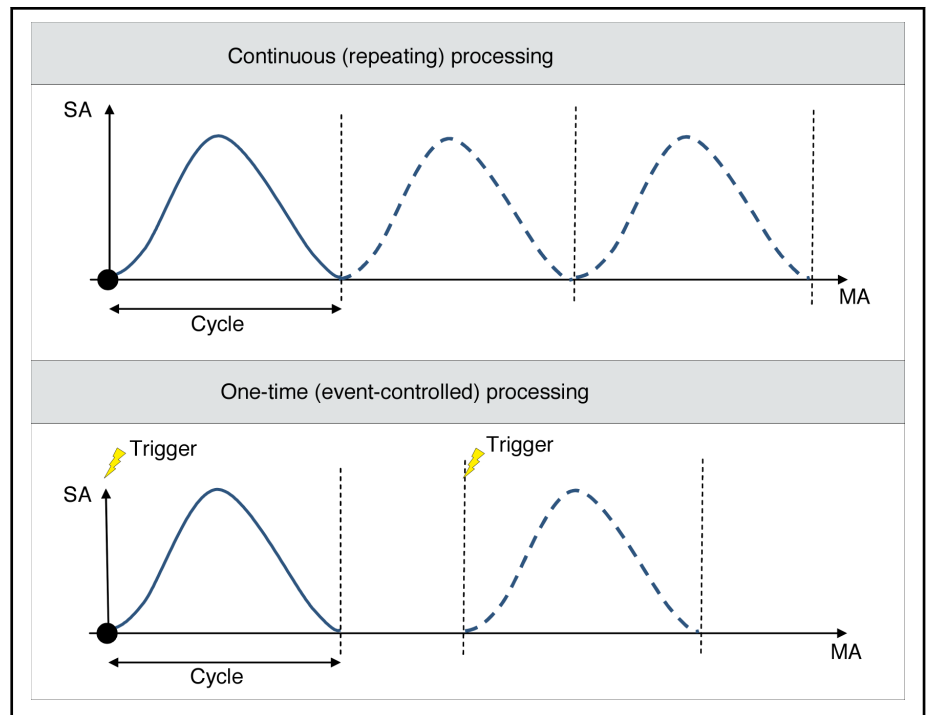


Fig. 6-62: Continuous processing and one-time processing

During the continuous processing, the active FlexProfile is executed cyclically. Ensure that the final profile velocity of the last motion step is identical to the initial velocity of the entire profile. If necessary, an error message is output. With the one-time processing, the selected profile is only executed once. The same profile can be started several times with a new call for the "MC_FlexProfile" function block.

6.6.13 PLCopen function blocks for supporting FlexProfiles

General information

The IndraLogic XLC/IndraMotion MLC system can manage up to 99 axes ¹⁰⁾ with synchronized and coordinated movements.

Four profiles can be respectively assigned to one axis, of which one can be activated. If necessary, it is possible to switch to the other profiles.

Each profile can take up a maximum of 64 steps.

The individual motion steps can either be defined with certain motion laws (VDI 2143 and special motion laws) or with referenced cam tables.

For registration of the cam tables, the control provides storage space for a maximum of 99 point tables with three to a maximum of 1,024 points each in its Compact Flash (CF). The point tables can be addressed via the parameters C-0-2001 ... 2099.

The cam tables can be configured either via the PLC function block [MB_ChangeCamData](#), [page 225](#), or via the IndraWorks interface.

¹⁰⁾ depending on the type of control



The FlexProfile functionality is supported by the following axis types (slave axis):

- IndraDrive with interpolation in the control
- HydraulicDrive with interpolation in the control
- SercosDrive
- Virtual axis
- Controller axis

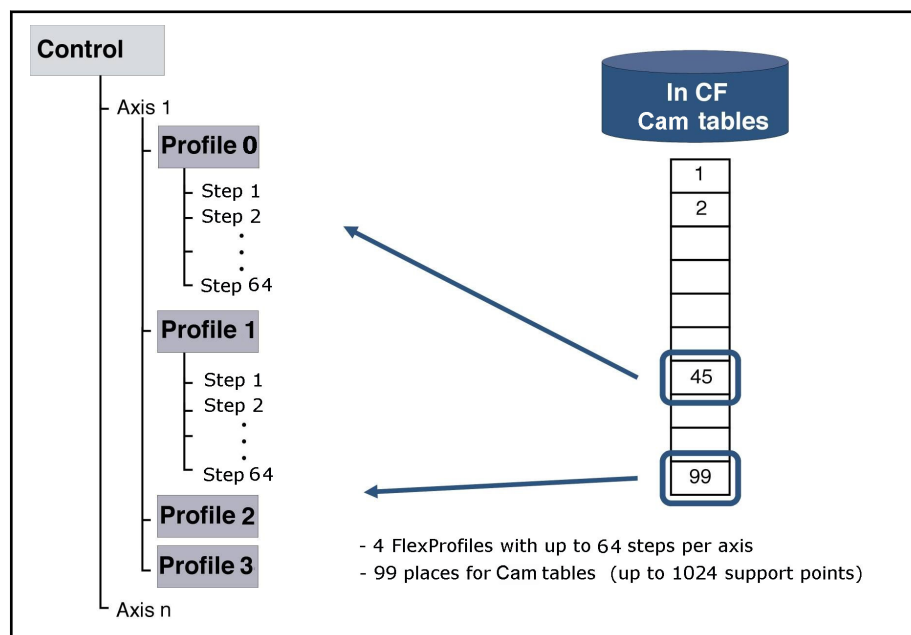


Fig. 6-63: Assigning the cam table/FlexProfile

For processing FlexProfiles and cam tables on the MLC, the following blocks are available:

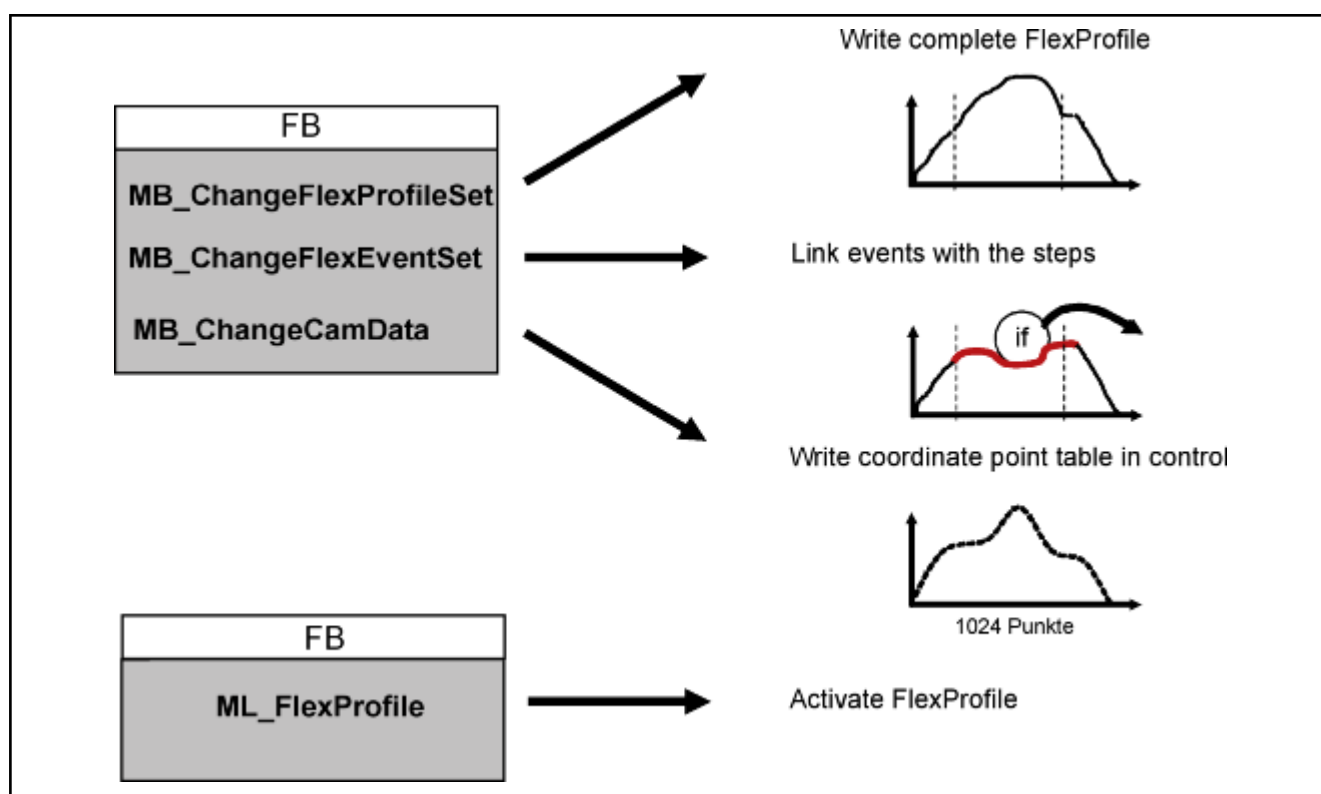


Fig. 6-64: Overview of the PLC blocks

- [MB_ChangeFlexProfileSet](#), page 239: Writing a complete FlexProfile, from the PLC to the Motion
- [MB_ChangeFlexEventSet](#), page 242: Configuration of events.
- [MB_ChangeCamData](#), page 225: Transferring a cam table from the PLC memory to the Motion, C-0-2001...2099.

MB_ChangeFlexProfileSet

Brief description

The function block "MB_ChangeFlexProfileSet" allows to write on a complete MotionProfile (FlexProfile) with up to 64 steps or to exchange an existing one.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-42: Library assignment

Interface description

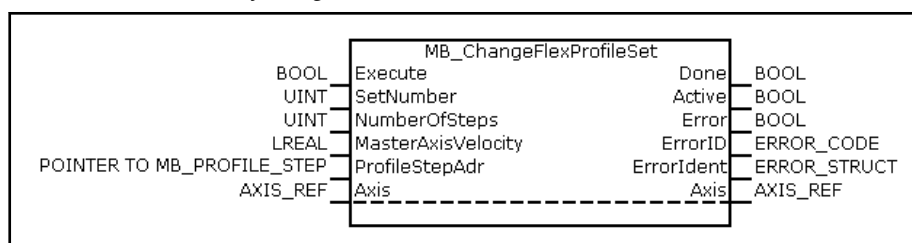


Fig. 6-65: MB_ChangeFlexProfileSet function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge.
	SetNumber	UINT	Selecting the FlexProfile (0 ... 3)
	NumberOfSteps	UINT	Number of FlexProfile steps (1...64)
	MasterAxisVelocity	LREAL	Standardized master axis velocity the specified velocities relate to
	ProfileStepAdr	POINTER TO MB_PRO-FILE_STEP	Pointer to structure data array containing the definition of the FlexProfile.
VAR_OUTPUT	Done	BOOL	Data were accepted.
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	Indicates that an error has occurred in the function block instance.
	ErrorID	ERROR_CODE	Brief indication of the error cause.
	ErrorIdent	ERROR_STRUCT	Detailed information on the error.
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the axis whose FlexProfile set is to be exchanged.

Tab. 6-43: Interface of the MB_ChangeFlexProfileSet function block

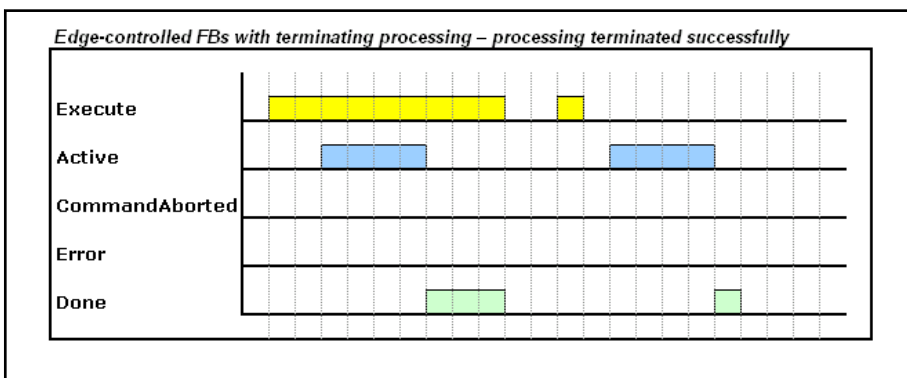
Signal-time diagram

Fig. 6-66: Signal-time diagram: MB_ChangeFlexProfileSet

Functional description

The function block allows a complete FlexProfile with 1...64 steps to be transferred from the PLC to the Motion component of the control.

The data characterizing the respective steps are saved in the list, the beginning of which has to be described in the PLC storage space with a pointer (TO LREAL).

Troubleshooting

The function block MB_ChangeProfileSet generates the following types of error messages:

- [MLC_TABLE](#), [ERROR_TABLE 16#0030](#), [page 360](#) - error when accessing the axis/the control.

Characteristics of the Event Control

In almost all the processing machines, several axes participate in the process. The motion tasks to be carried out by the individual axes mostly depend on one another and result from the application task.

Therefore, the motion of one axis often has to start at exactly the place where another axis has reached a certain position or where another axis remained in a motion step for a certain time. This is shown in the following picture.

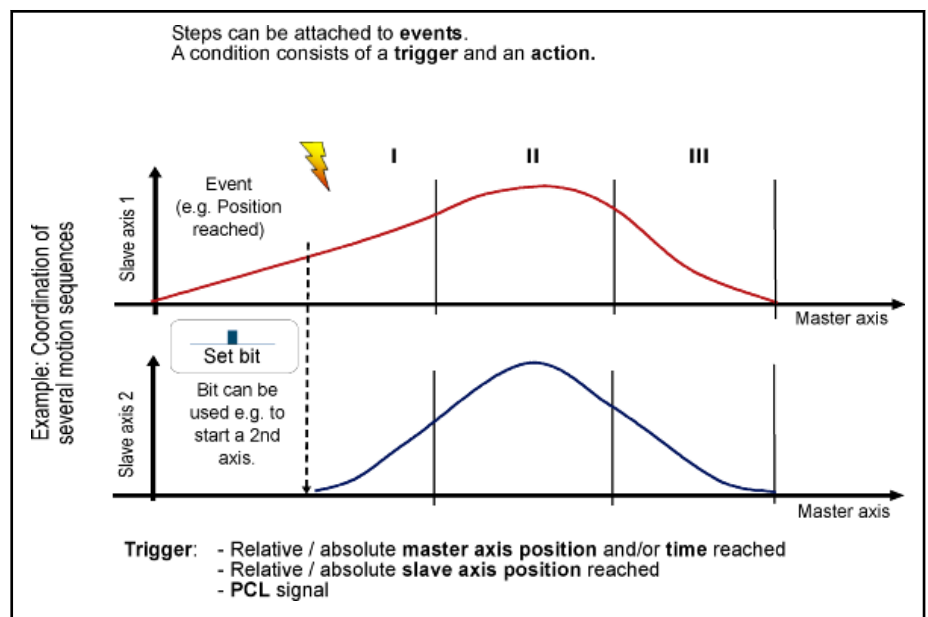


Fig. 6-67: Example of an event control

Events are very suitable to mark these points in the MotionProfile. It is very easy to react in a PLC via the AxisData structure. It is possible to start follow-up motions of other axes or to decide on the further processing. A resulting dynamic "motion chain" is realized in a PLC reflecting the machine dependencies.

Events can be attached to the motion steps. One condition consists of a trigger and an action.

The trigger identifies the control variable states. These are:

- **Relative/absolute master axis position reached**
Describes the reaching of a relative/absolute position of the master axis. Similar to a programmable limit switch, the current position of a master source can be monitored. If it crosses a specified position in the current motion step, the event occurs.
- **Relative/absolute slave axis position reached**
Describes the reaching of a relative/absolute position of the slave axis. The same concept can also be applied to the position of the slave axis.
- **PLC signal**
Actions can be triggered by a PLC. Events are reported to the PLC or directly start an action.
- **Time-controlled abort**
The step is not time-controlled. That means that it has no time master, but a master axis to which traveling should be synchronous. Connect an event which monitors the travel time since the beginning of the motion step with an action triggering an abort of the current motion step. The resulting MotionProfile can be used generically for different master axis velocities.
- **Monitoring for time segments**
With regard to the time-master functionality, the events can also be used for the monitoring of different master sources. Depending on the motion task, an event can be set in such a way that the travel distance of the master axis is monitored during a motion step coupled to the time master. Thus, execute error reactions if the master axis traveled too far

during this time. Different process states can be easily and comfortably buffered by the user with the help of the events. It is possible to react on this in a PLC.

The action is an active element responsible for the activation of certain variables.

Possible actions are:

- Set bit immediately
One of the available 16 bits is immediately set in AxisData.
- Set bit at the end of the motion step
One of the available 16 bits is set in the AxisData at the end of the motion step.
- Set bit at the end of the profile
One of the available 16 bits is set in the AxisData at the end of the profile.
- Set bit immediately and continue with the next motion step
Causes the immediate abortion of the current motion step. This step is not processed completely, instead, the next motion step begins directly from the ongoing motion. If this is a Flex motion step, this change can take place smooth and jerk-free.

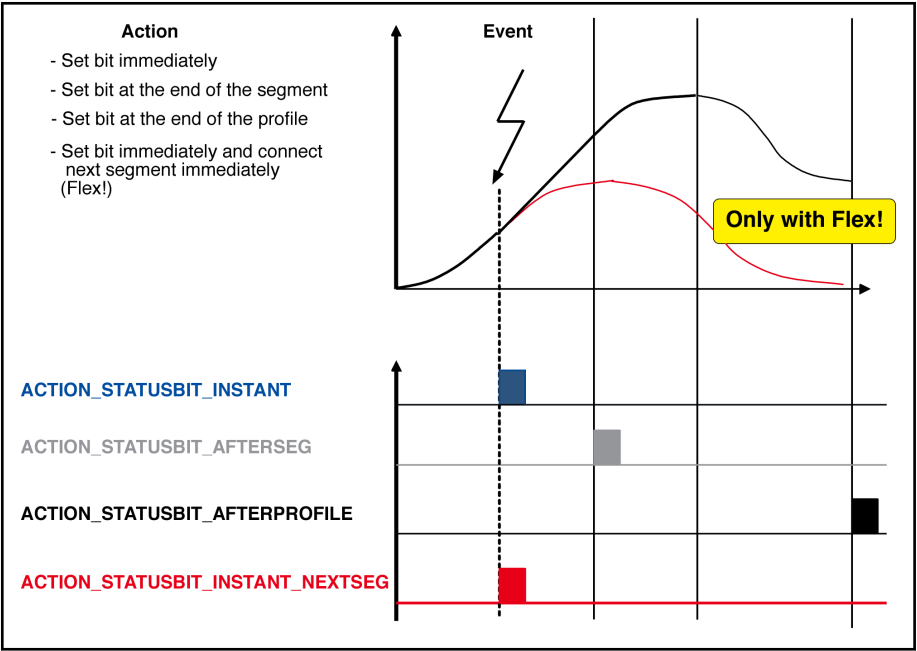


Fig. 6-68: Possible actions

The events can be configured either via the PLC function block "MB_ChangeFlexEventSet" or the IndraWorks interface.

MB_ChangeFlexEventSet

Brief description

The function block "MB_ChangeFlexEventSet" is used for the configuration of events.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-44: Library assignment

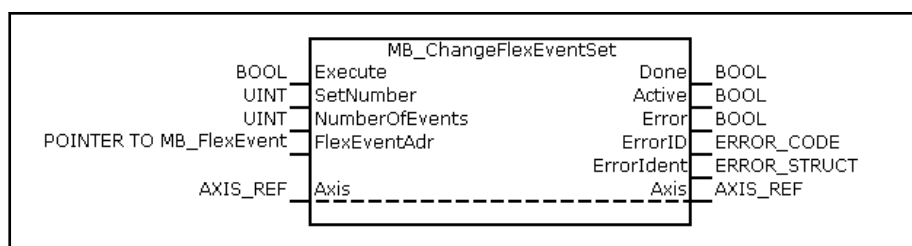


Fig. 6-69: MB_ChangeFlexEventSet function block

Interface description

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge.
	SetNumber	UINT	Selection of FlexProfile (0 ...3)
	NumberOfEvents	UINT	Number of events to be defined (1..16)
	FlexEventAdr	POINTER TO MB_FLEX_EVENT	Definition list of the events (1..16) The max. number of events per FlexProfile set is 16. The events can be distributed as desired across the FlexProfile set.
VAR_OUTPUT	Done	BOOL	Data was applied
	Active	BOOL	Processing of data runs after preprocessing is completed.
	Error	BOOL	Indicates that an error has occurred in the function block instance.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the axis for which the MotionProfile set is to be exchanged.

Tab. 6-45: Interface of the MB_ChangeFlexEventSet FB

- A maximum of 16 events can be defined as per FlexProfile.
- All the defined events can be distributed as desired across the FlexProfile set, i.e. all the 16 events can also be connected with a joint motion step.
- An event can only be referenced once.
- If similar events are referenced on several motion steps, these have to be duplicated.

Troubleshooting The function block MB_ChangeFlexEventSet produces the following error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.6.14 MC_GearIn

Brief description The function block MC_GearIn implements an electronic gear function with fine adjustment.



The functionality is supported by the following axis types (slave axis):

- IndraDrive with interpolation in the drive and in the control
- HydraulicDrive with interpolation in the control
- SercosDrive
- Virtual axis
- Controller axis

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Motion

Tab. 6-46: Library assignment

Interface description

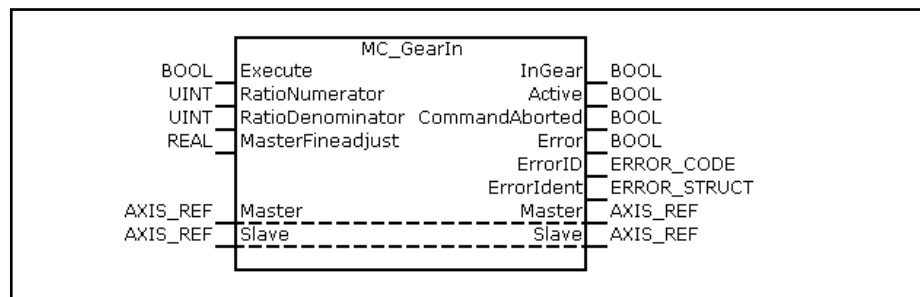
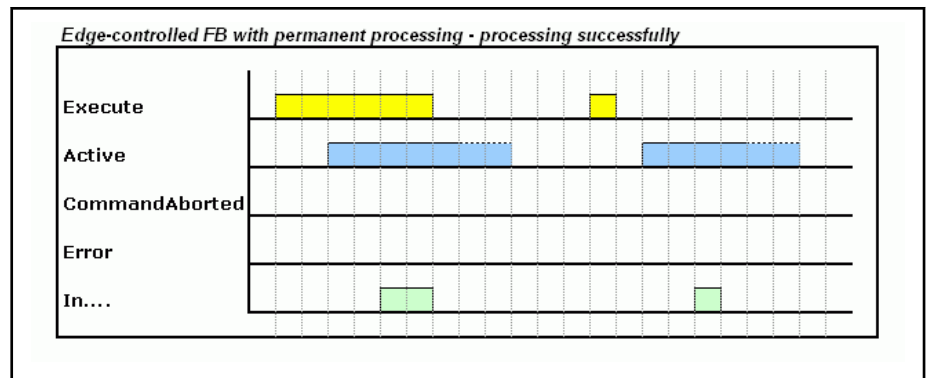


Fig. 6-70: MC_GearIn function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
	RatioNumerator	UINT	Gear ratio - numerator "Master axis gear output revolutions" Min: 1; max: 65535; default: 1
	RatioDenominator	UINT	Gear ratio - denominator "Master axis gear input revolutions" Min: 1; max: 65535; default: 1
	Master Fineadjust	REAL	"Fine adjustment of gear ratio" in %; default: 0.0, min.: -799,9999%, max.: 799.9999
VAR_OUTPUT	InGear	BOOL	Slave axis follows master axis, coupling was successful (depends on the parameter "A-0-0623, Synchronization run window, velocity")
	Active	BOOL	Processing of data after preprocessing running
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Master	AXIS_REF	Contains information on the master axis
	Slave	AXIS_REF	Contains information on the slave axis following the master axis

Tab. 6-47: Interface of the MC_GearIn function block

Signal-time diagram



In... InGear

Fig. 6-71: Signal-time diagram

Troubleshooting

The function block MC_GearIn generates the following error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.6.15 MC_GearIn, special features of Sercos-based axes

Functional description

The gear function is realized with a drive internal gear functionality or calculated on the control. The objective is a velocity-synchronous (drift-free) run between the master axis (A-0-2700, ELS, master axis, control number, A-0-2701, ELS, master axis, axis number) and the selected slave axis.



The functionality is supported by the following axis types (slave axis):

- IndraDrive with interpolation in the drive and in the control
- HydraulicDrive with interpolation in the control
- SercosDrive
- Virtual axis
- Controller axis

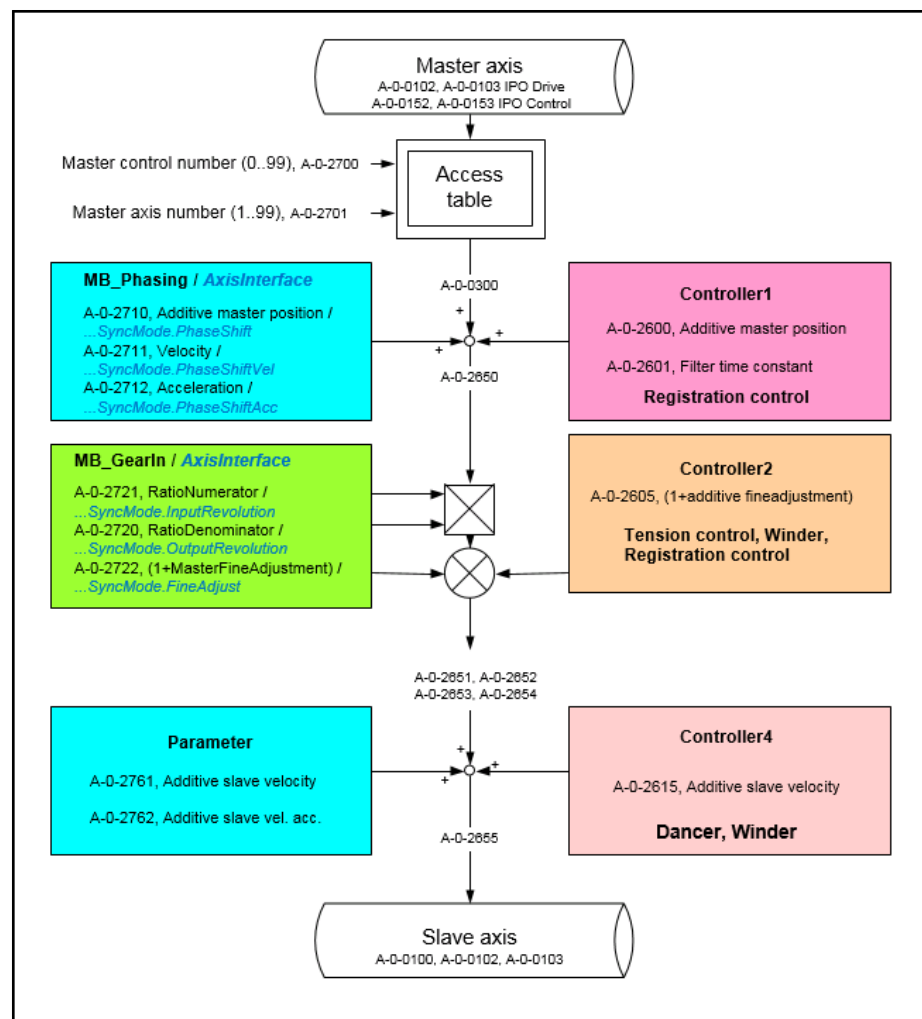


Fig. 6-72: MotionControl functional chain MC_GearIn

The operation mode provides the following access points:

- The additive master axis position defined by "A-0-2710, ELS, phasing, additive position command value" can be specified via MB_Phasing or the AxisInterface
- Electronic gear defined by the ratio of "A-0-2721, ELS, master axis gear, output revolutions" and "A-0-2720, ELS, master axis gear, input revolutions" and the "A-0-2722, ELS, gear ratio fine adjust". The fine adjustment value is specified in percent and calculated as (1 + fine adjustment).

These values are specified via "MC_GearIn" or the AxisInterface

- The additive command velocity of the slave axis defined by "A-0-2761, ELS, PhasingSlaveVel, additive command velocity"

If the block is calculated in the drive (IndraDrive), the A-parameters are transferred to the following S- and P-parameters:

- A-0-2721 to P-0-0157,
- A-0-2720 to P-0-0156,
- A-0-2722 to P-0-0083.

Based on the incoming velocity signal, the block is able to implement the synchronization before master and slave axis engage.

6.6.16 MC_GearOut

Brief description

The function block MC_GearOut stops the gear function with fine adjustment enabled by either the function block MC_GearIn or MB_GearInPos or the profiles activated by either the function block MB_MotionProfile or ML_FlexProfile.

This block implements two jobs:

- Stops the velocity-synchronous run of the slave axis
- Transfers the slave axis to the "Continuous Motion" mode, while maintaining the last velocity value in case there is no other command.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Motion

Tab. 6-48: Library assignment

Interface description

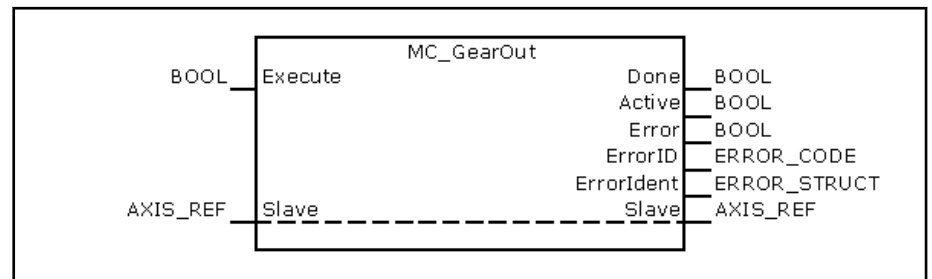


Fig. 6-73: FB MC_GearOut

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
VAR_OUTPUT	Done	BOOL	Decoupling completed successfully
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Slave	AXIS_REF	It contains information on the slave axis which follows the master axis

Tab. 6-49: Interface of the MC_GearOut function block

Signal-time diagram

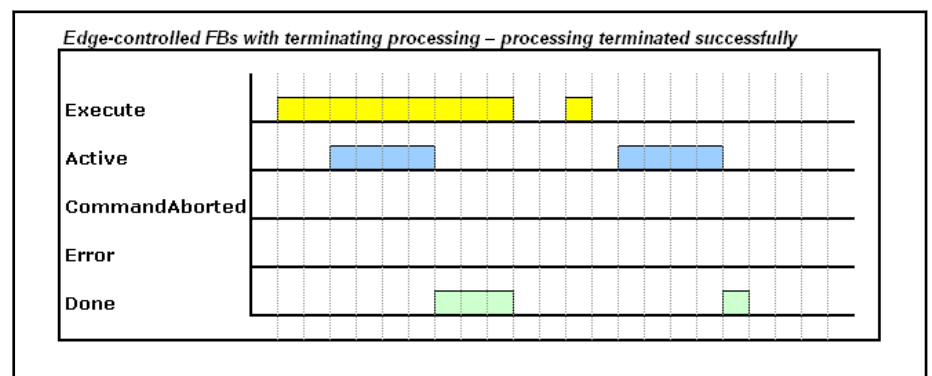


Fig. 6-74: Signal-time diagram

Troubleshooting The function block MC_GearOut generates the following error messages:

- [MLC_TABLE](#), [page 360](#) - error when accessing the axis/control.

6.6.17 MB_GearInPos

Brief description The function block MB_GearInPos implements an electronic gear function with fine adjustment.

The objective is a phase-synchronous and velocity-synchronous (drift-free) run between the master axis and the selected slave axis.



The functionality is supported by the following axis types (slave axis):

- IndraDrive with interpolation in the drive and in the control
- HydraulicDrive with interpolation in the control
- SercosDrive
- Virtual axis
- Controller axis

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-50: Library assignment

Interface description

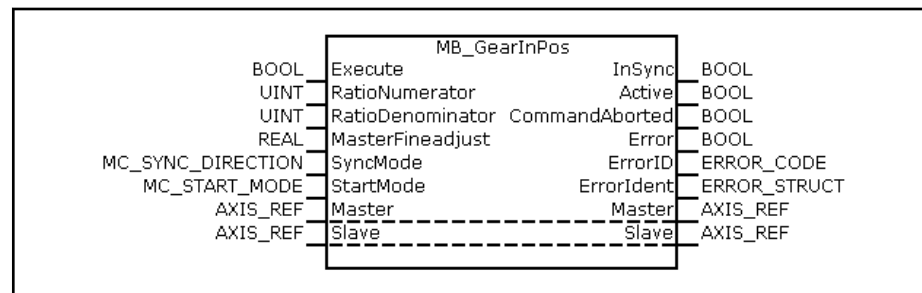


Fig. 6-75: MB_GearInPos function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
	RatioNumerator	UINT	Gear ratio - numerator "master axis gear output revolutions" Min: 1; max: 65535; default: 1
	RatioDenominator	UINT	Gear ratio - denominator "master axis gear input revolutions" Min: 1; max: 65535; default: 1
	MasterFineadjust	REAL	"Fine adjustment of gear ratio" in %; default: 0.0, min.: -799,9999%, max.: 799.9999
	SyncMode	MC_SYNC_DIRECTION	Synchronization direction MC_SYNC_DIRECTION , page 59
	StartMode	MC_START_MODE	Synchronization type MC_START_MODE , page 57
VAR_OUTPUT	InSync	BOOL	Synchronous run reached (depends on the parameter "A-0-0622 Synchronization run window, position")

	Name	Type	Comment
	Active	BOOL	Processing of data after preprocessing running
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Master	AXIS_REF	Contains information on the master axis
	Slave	AXIS_REF	Contains information on the slave axis following the master axis

Tab. 6-51: Interface of the MC_GearInPos function block

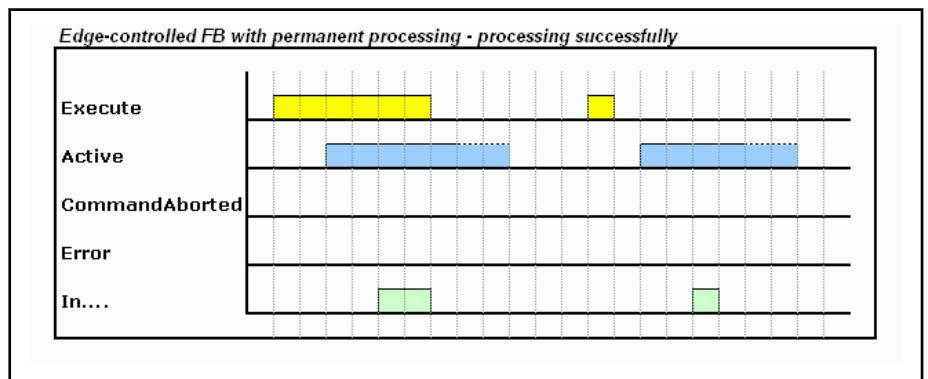


Only the two start modes

- ABSOLUTE = 0
- RELATIVE = 1

of the enumeration table MC_START_MODE are to be used for command MB_GearInPos. If a different value is used at the "StartMode" input, synchronization automatically takes place in mode RELATIVE.

Signal-time diagram



In... InSync

Fig. 6-76: Signal-time diagram

Troubleshooting

The function block MB_GearInPos generates the following error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.6.18 MB_GearInPos, special features of Sercos-based axes

Functional description

The gear function is realized with a drive internal gear functionality or calculated on the control. The objective is a phase- and velocity-synchronous (drift-free) run between the master axis (A-0-2700, ELS, master axis, control number, A-0-2701, ELS, master axis, axis number) and the selected slave axis.



The functionality is supported by the following axis types (slave axis):

- IndraDrive with interpolation in the drive and in the control
- HydraulicDrive with interpolation in the control
- SercosDrive
- Virtual axis
- Controller axis

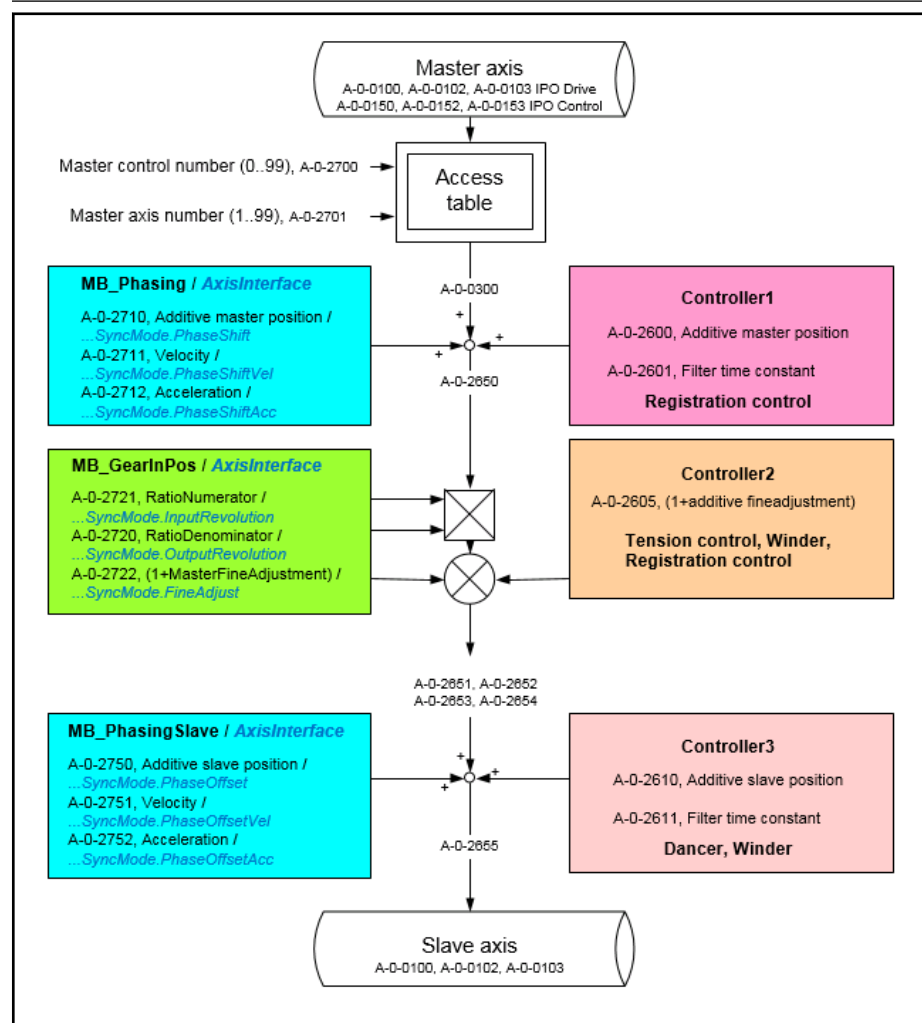


Fig. 6-77: MotionControl functional chain MC_GearInPos

The operation mode provides the following access points:

- The additive master axis position defined by "A-0-2710, ELS, phasing, additive position command value" can be specified via MB_Phasing or the AxisInterface
- Electronic gear defined by the ratio of "A-0-2721, ELS, master axis gear, output revolutions" and "A-0-2720, ELS, master axis gear, input revolutions" and the "A-0-2722, ELS, gear ratio fine adjust". The fine adjustment value is specified in percent and calculated as (1 + fine adjustment).

These values are specified via MB_GearInPos or the AxisInterface

- The additive slave axis position defined by "A-0-2750, ELS, PhasingSlave, additive position command value" can be specified via MB_PhasingSlave or the AxisInterface

If the block is calculated in the drive (IndraDrive), the A-parameters are transferred to the following S- and P-parameters:

- A-0-2721 to P-0-0157,
- A-0-2720 to P-0-0156,
- A-0-2722 to P-0-0083.

Based on the incoming position and velocity signals, the block is able to implement the synchronization before master and slave axis engage.

6.7 Function block for the axis group control

6.7.1 MC_AddAxisToGroup

Brief description The function block MC_AddAxisToGroup adds an axis to an axis group. This function block does not trigger a motion. The "Group State Machine" changes to the state "GroupStandStill, no IRDProgramSelected" when all axes necessary for the group are added.



The "MC_AddAxisToGroup" and "MC_RemoveAxisFromGroup" function blocks have been moved from ML_Robot.library to ML_PLCopen.library, to make them available for IndraLogic XLC.



The necessary axes of the group have to be configured in the IndraWorks. The results of the configuration are filed in parameter "K-0-0008, Axis configuration list".

The ID of an axis within a group is also specified in the parameter K-0-0008.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-52: Library assignment

Interface description

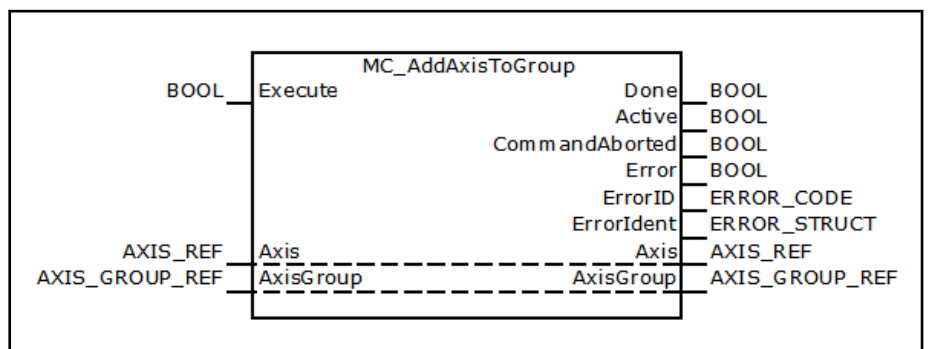


Fig. 6-78: MC_AddAxisToGroup

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
VAR_OUTPUT	Done	BOOL	The axis has been successfully included in the axis group

	Name	Type	Comment
	Active	BOOL	Processing of data after preprocessing running
	CommandAborted	BOOL	Command aborted due to a subsequent axis command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	AxisGroup	AXIS_GROUP_REF	Control number and object number of the axis group
	Axis	AXIS_REF	Control number and object number of the axis

Tab. 6-53: Interface of the MC_AddAxisToGroup FB

Signal-time diagram

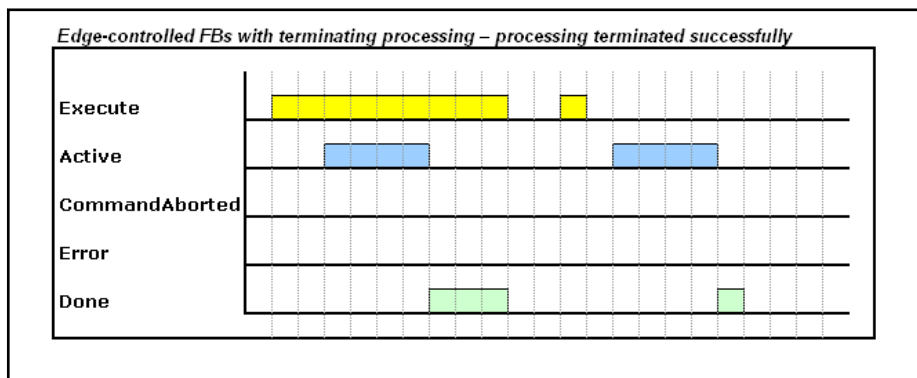


Fig. 6-79: Signal-time diagram: MC_AddAxisToGroup

6.7.2 MC_RemoveAxisFromGroup

Brief description The function block MC_RemoveAxisFromGroup removes an axis from the axis group. This function block does not trigger a motion. The "Group State Machine" switches to state "GroupUndefined".



The "MC_AddAxisToGroup" and "MC_RemoveAxisFromGroup" function blocks have been moved from ML_Robot.library to ML_PLCOpen.library, to make them available for IndraLogic XLC.



The required axes of the group are filed in the parameter "K-0-0008, Axis configuration list". The ID of an axis within a group is also specified in the parameter K-0-0008.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Motion

Tab. 6-54: Library assignment

Interface description

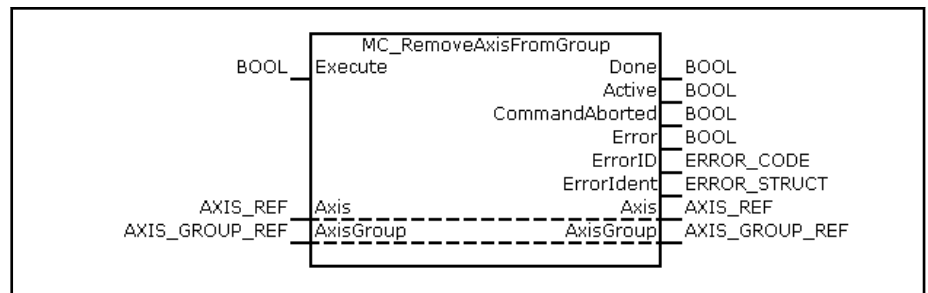


Fig. 6-80: MC_RemoveAxisFromGroup

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
VAR_OUTPUT	Done	BOOL	The axis has been successfully removed from the axis group
	Active	BOOL	Processing of data after preprocessing running
	CommandAborted	BOOL	Command aborted by a subsequent axis command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Control number and object number of the axis
	AxisGroup	AXIS_GROUP_REF	Control number and object number of the axis group

Tab. 6-55: Interface of the MC_RemoveAxisFromGroup function block

Signal-time diagram

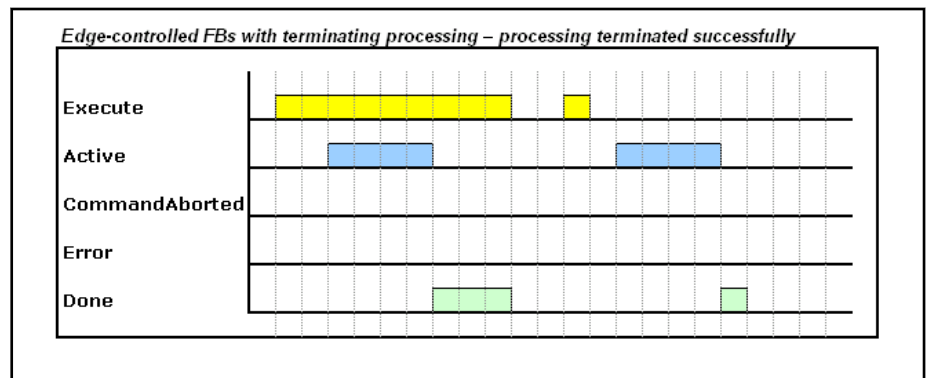


Fig. 6-81: Signal-time diagram: MC_RemoveAxisFromGroup

6.8 Function blocks to use the cyclic data channel

6.8.1 General information on using the cyclic data channel

With the sercos bus, a cyclic data channel for transmitting information from the control to an addressed axis can be used.

To do this, the block "ML_OpenCyclic***Channel" is used to switch the addressed axis to the operating mode "Cyclic *** Command" and the respective cyclic command value channel is opened.

With the command value channel open, it is possible to use the block "ML_WriteCyclic***" to write a command value for the axis during each cycle which will take effect on the axis in the next cycle.



The function blocks to use the cyclic data channel require the **function package MOT** of the control

The following blocks are available:

Analog command value channel, operating mode "Cyclic Analog Command"

- [ML_OpenCyclicAnalogChannel](#), page 255, with
- [ML_WriteCyclicAnalog](#), page 256,

Position command value channel, operation mode "Cyclic Position Command"

- [ML_OpenCyclicPositionChannel](#), page 257, with
- [ML_WriteCyclicPosition](#), page 259,

Power command value channel, operation mode "Cyclic Torque Command"

- [ML_OpenCyclicTorqueChannel](#), page 261, with
- [ML_WriteCyclicTorque](#), page 263,

Velocity command value channel, operation mode "Cyclic Velocity Command"

- [ML_OpenCyclicVelocityChannel](#), page 257, with
- [ML_WriteCyclicVelocity](#), page 265,

Two blocks have to be used in combination with each other. The following example describes how the function blocks [ML_OpenCyclicPositionChannel](#) and [ML_WriteCyclicPosition](#) have to be used.

The target is to command position command values from the PLC to an axis. Initially, it is not ensured if the cycle position command value channel is already open. The function blocks [ML_OpenCyclicPositionChannel](#) and [ML_WriteCyclicPosition](#) are used in a PLC task which runs synchronously to the Motion calculation. The function blocks should be interconnected as follows:

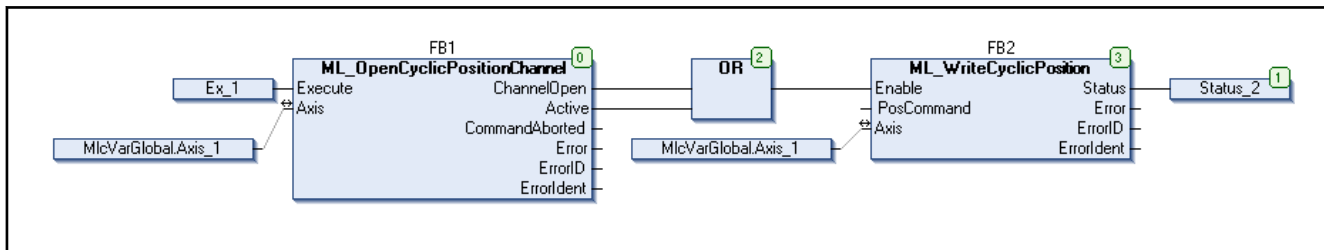


Fig. 6-82: Interconnecting the function blocks [ML_OpenCyclicPositionChannel](#) and [ML_WriteCyclicPosition](#)

First cycle: The "Execute" input of the [OpenCyclicTorqueChannel](#) function block has to be set to TRUE if the individual command value specification should be started. If subsequently the error output is set to TRUE, the cycle position command value channel has already been opened by another user. In this case, the function block [ML_WriteCyclicPosition](#) should not be overwritten with a position command value. The command value of the other user of the cyclic channel would be overwritten. If Active is TRUE, the first command value has to be written with the [ML_WriteCyclicPosition](#) block. The outputs CommandAborted and ChannelOpen cannot be TRUE during the first clock cycle.

Enabling cycles: The output "ChannelOpen" is FALSE when opening the cyclic command value channel. If another user tries to simultaneously open the

cyclic command value channel, the output is Error TRUE. If the output CommandAborted is TRUE, opening of the cyclic channel was canceled by another Motion command. The command value which is to be activated should be written during this process and if the channel was opened.

Work cycles: The output "ChannelOpen" is TRUE when opening the cyclic command value channel. A cyclic position command value has to be written over the ML_WriteCyclicPosition function block of the axis.

Closing the command value channel: A cyclic channel cannot be closed explicitly. It can only be canceled and closed by another Motion function block (e.g. MC_Stop). When the output CommandAborted is TRUE at the ML_OpenCyclicPositionChannel function block, no cyclic position command value may be written via the ML_WriteCyclicPosition function block. The execute input can be set to FALSE at the ML_OpenCyclicPositionChannel function block.

Summary:

Output	Description/action
Active	The cyclic command value channel is opened. The cyclic position command value has to be written via the ML_WriteCyclicPosition function block.
ChannelOpen	The cyclic position command value channel is opened. The cyclic position command value has to be written via the ML_WriteCyclicPosition function block.
Error	The cyclic command value channel is used by another user. No command value can be written via the ML_WriteCyclicPosition function block.
CommandAborted	The cyclic position command value channel was closed. No command value can be written via the ML_WriteCyclicPosition function block.

Tab. 6-56: Summary of ML_OpenCyclicPositionChannel outputs

The function blocks for the cyclic velocity/torque and analog channel can be used analogously.

6.8.2 ML_OpenCyclicAnalogChannel

Brief description The function block switches the axis to the mode 'Cyclic Analog Command' and opens the cyclic velocity analog value channel.



When ML_OpenCyclicAnalogChannel is activated and the cyclic analog command value channel is already open, an error is generated.

The function block has to be used in MotionTask (synchronous to the Sercos cycle). Otherwise, jerky axis motions can result.

When the "Execute" input is deactivated, all outputs are set to FALSE. The cyclic analog command value channel remains open if it has been open before.

If the function block was interrupted by another Motion function block, the function block is aborted and the cyclic analog command value channel is closed.

The function block is only available for a real axis of type HydraulicDrive.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Motion

Tab. 6-57: Library assignment

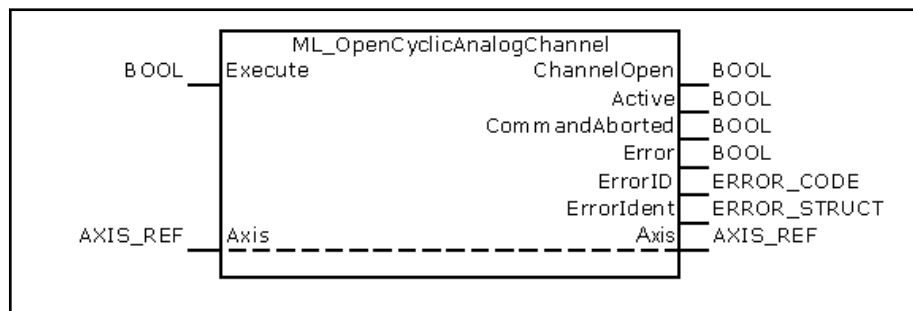
Interface description

Fig. 6-83: ML_OpenCyclicAnalogChannel function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge.
VAR_OUTPUT	ChannelOpen	BOOL	Indicates that the cyclic analog command value channel is open and that the axis follows the analog value of the last cycle.
	Active	BOOL	Indicates that the function block set the axis to the "Cyclic Analog Command" mode.
	CommandAborted	BOOL	Command was aborted by another command.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the concrete axis

Tab. 6-58: Interface of the ML_OpenCyclicAnalogChannel function block

Troubleshooting

The function block ML_OpenCyclicAnalogChannel generates the following error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.8.3 ML_WriteCyclicAnalog

Brief description

The function block ML_WriteCyclicAnalog writes an analog value for an axis. If the cyclic analog command value channel is open, the analog value is applied to the axis in the next cycle.



To open the cyclic analog command value channel, use the function block [ML_OpenCyclicAnalogChannelCtrl, page 270](#).

The function block has to be used in a PLC task (synchronous to the Sercos cycle). Otherwise, jerky axis motions can result.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Motion

Tab. 6-59: Library assignment

Interface description

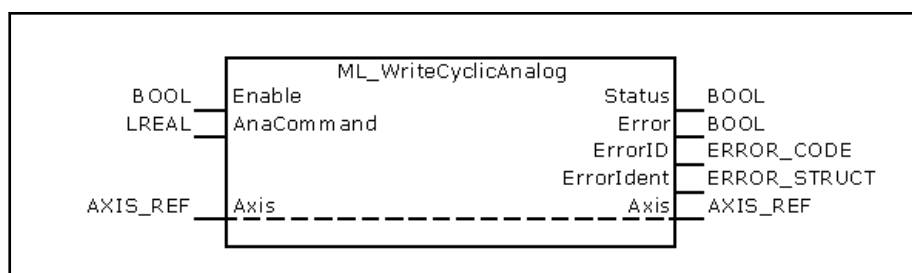


Fig. 6-84: ML_WriteCyclicAnalog

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	As long as the "Enable" output is TRUE, "AnaCommand" is written on the axis. The analog value only becomes effective when the cyclic analog command value channel is open.
	AnaCommand	LREAL	Indicates the analog value to be directed at by the axis in the next cycle. The analog value only becomes effective when the cyclic analog command value channel is open.
VAR_OUTPUT	Status	BOOL	Indicates that the analog value has been written on the axis. The analog value only becomes effective when the cyclic analog command value channel is open.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-60: Interface of the ML_WriteCyclicAnalog function block

Special features of the AnaCommand input

1. The values of the "AnaCommand" input become effective in the next cycle if the cyclic analog command value channel is open.
2. The value of the "AnaCommand" input does not have the scaling of the assigned axis but is active without unit.

Troubleshooting

The function block ML_WriteCyclicAnalog generates the following error messages in:

- [MLC_TABLE, page 360](#) - Error when accessing the axis/control.

6.8.4 ML_OpenCyclicPositionChannel

Brief description

The function block switches the axis to the mode "Cyclic Position Command" and opens the cyclic position command value channel



When ML_OpenCyclicPositionChannel is activated and the cyclic position command value channel is already open, an error is generated.

The function block has to be used in MotionTask (synchronous to the Sercos cycle). Otherwise, jerky axis motions can result.

When the "Execute" input is deactivated, all outputs are set to FALSE. The cyclic position command value channel remains open if it has been open before.

If the function block was interrupted by another Motion function block, the function block is aborted and the cyclic position command value channel is closed (also see signal-time diagram).

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Motion

Tab. 6-61: Library assignment

Interface description

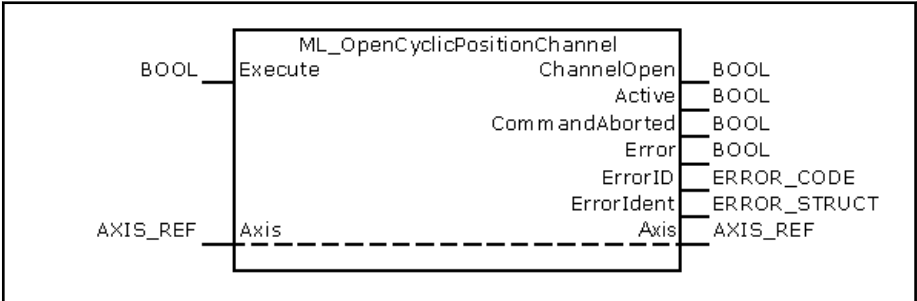


Fig. 6-85: ML_OpenCyclicPositionChannel function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge.
VAR_OUTPUT	ChannelOpen	BOOL	Indicates that the cyclic position command value channel is open and that the axis follows the position value of the last cycle.
	Active	BOOL	Indicates that the function block sets the axis to the "Cyclic Position Command" mode.
	CommandAborted	BOOL	Command was aborted by another command.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the concrete axis

Tab. 6-62: Interface of ML_OpenCyclicPositionChannel function block

Signal-time diagram

The functioning of the block becomes clear when observing the following pulse diagram. The function block operates in competition with an instance of a [MC_MoveAbsolute](#), page 183

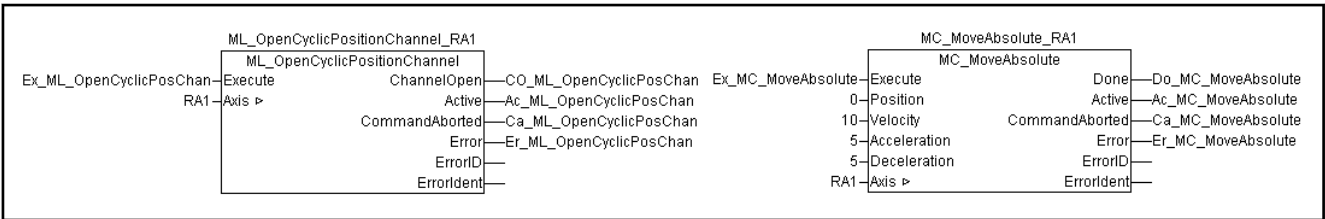


Fig. 6-86: Time behavior ML_OpenCyclicPositionChannel

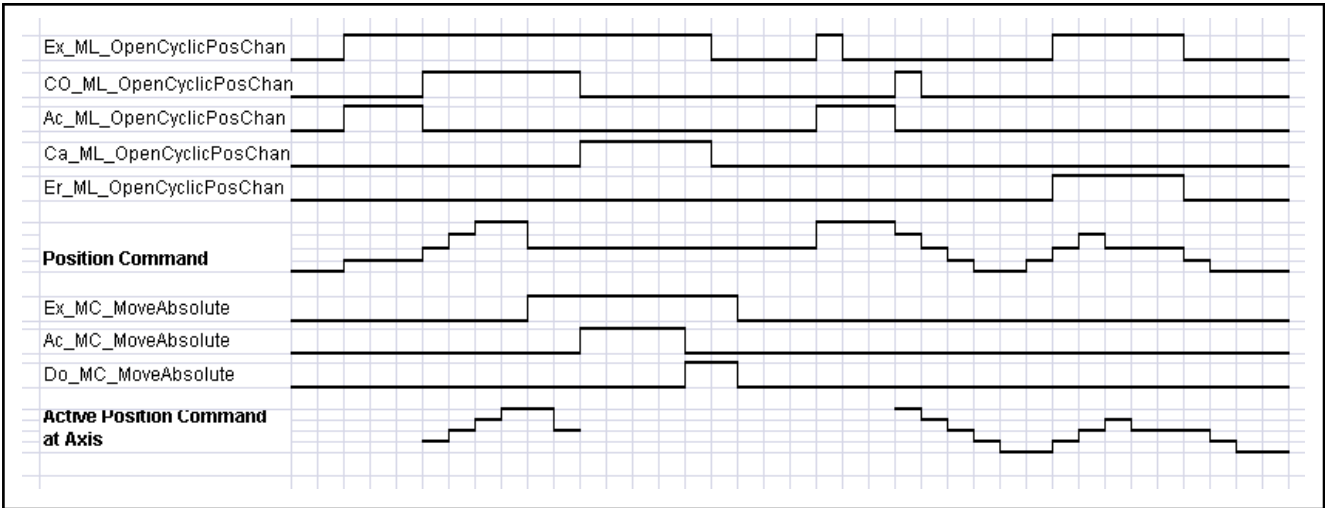


Fig. 6-87: Time behavior ML_OpenCyclicPositionChannel

- Troubleshooting** The function block ML_OpenCyclicPositionChannel generates the following error messages:
- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.8.5 ML_WriteCyclicPosition

Brief description The function block ML_WriteCyclicPosition writes a position command for an axis. If the cyclic position command value channel is open, the position command is applied to the axis in the next cycle.

 To open the cyclic position command value channel, use the function block [ML_OpenCyclicPositionChannel, page 257](#). The function block has to be used in a PLC task (synchronous to the Sercos cycle). Otherwise, jerky axis motions can result.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-63: Library assignment

Interface description

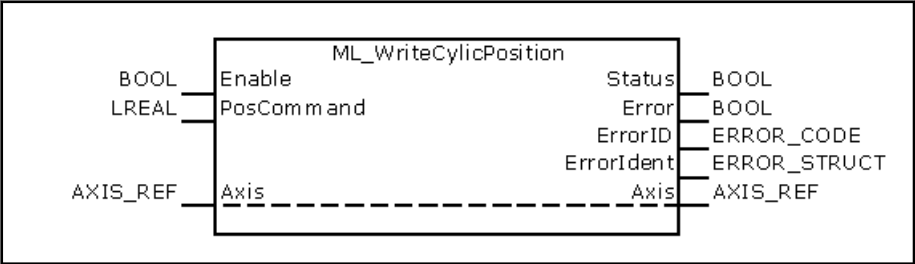


Fig. 6-88: ML_WriteCyclicPosition

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	As long as the "Enable" output is TRUE, "PosCommand" is written on the axis. The position command only becomes effective when the cyclic position command value channel is open.
	PosCommand	LREAL	Indicates the position to be directed at by the axis in the next cycle. The position command only becomes effective when the cyclic position command value channel is open.
VAR_OUTPUT	Status	BOOL	Indicates that the position command has been written on the axis. The position command only becomes effective when the cyclic position command value channel is open.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the concrete axis

Tab. 6-64: Interface of the ML_WriteCyclicPosition function block

Special features of the PosCommand input

1. The values of the "PosCommand" input become effective in the next cycle, if the cyclic position command value channel is open.
2. The values of the "PosCommand" input should be close to the position of the motion vector.
3. The value of the "PosCommand" input has the scaling of the assigned axis (linear, rotary, modulo).

Troubleshooting

The function block ML_WriteCyclicPosition generates the following error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

Implementation example:**Common use of ML_OpenCyclicPositionChannel and ML_WriteCyclicPosition**

The "ChannelOpen" output OR-linked with the "Active" output of the FB1 can be used as "Enable" input by the FB2. The "PosCommand" input of the FB2 has to be assigned valid values from the position table: The first value from the position table has to be written on the "PosCommand" input as long as the "Active" output is set by the FB1. The following entries from the position table have to be written on the "PosCommand" input as long as the "ChannelOpen" output is set by the FB1.

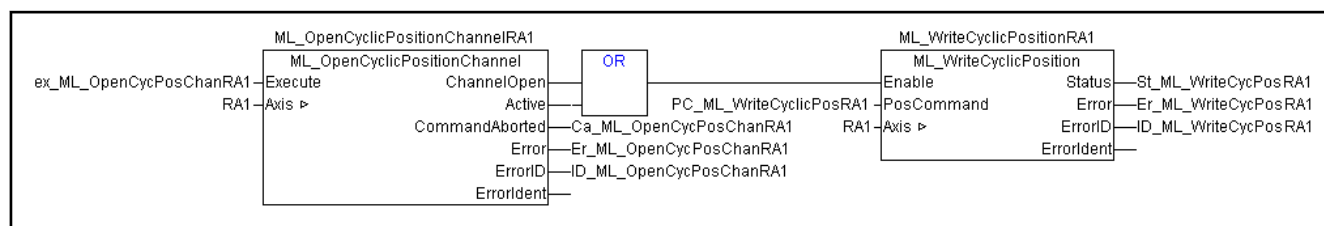


Fig. 6-89: Example: ML_OpenCyclicPositionChannel and ML_WriteCyclicPosition

First cycle:

- The execute input is set at the ML_OpenCyclicPositionChannel function block.
- If afterwards, the "Error" output is set, the cyclic position command value channel has already been open and is used by another block instance.

- If the "Active" output is set afterwards, the "Enable" input has to be set at the ML_WriteCyclicPosition function block and the "PosCommand" has to be set to the first entry of the position table.

The "CommandAborted" and "ChannelOpen" output cannot be set in the first cycle.

Activation cycles ("ChannelOpen" is not set)

- If the "Error" output is set, the cyclic position command value channel has been opened by another person during the initialization phase. The "Enable" input at the ML_WriteCyclicPosition function block can be disabled.
- If the "CommandAborted" output is set, the ML_OpenCyclicPosition-Command has been interrupted by another motion function block. The "Enable" input at the function block ML_OpenCyclicPositionCommand can be deactivated.
- If the "Active" output is set, the "Enable" input has to be set at the ML_WriteCyclicPosition function block and the "PosCommand" input has to be set to the first entry of the position table.

Work cycles

- "ChannelOpen" output is set. The value of the "PosCommand" input at the ML_OpenCyclicPositionCommand function block of the last cycle is effective at the axis. The "Enable" input has to be set at the ML_WriteCyclicPosition function block and the PosCommand input has to be set to the next entry of the position table.

Abort

- The cyclic position command value channel is closed when another motion function block (e.g. MC_Stop) becomes active on the axis.
- If the "CommandAborted" output is active, the cyclic position command value has been closed. In this cycle, no position command becomes effective on the drive. The "Enable" input at the ML_WriteCyclicPosition function block can be disabled. The "Execute" input at the ML_OpenCyclicPositionCommand function block can be disabled.

Summary

Output	Description/action
Active	The cyclic position command value is initialized. The "Enable" input has to be set at the ML_WriteCyclicPosition function block and the "PosCommand" input has to be set to the first entry of the position table.
ChannelOpen	The cyclic position command value channel is open. The "Enable" input has to be activated at the ML_WriteCyclicPosition function block and the "PosCommand" input has to be set to the next entry of the position table.
Error	The cyclic position command value channel has already been opened and is being used by somebody else. The "Enable" input at the ML_WriteCyclicPosition function block has to be disabled.
CommandAborted	The cyclic position command value channel has been closed. The "Enable" input at the ML_WriteCyclicPosition function block can be disabled.

Tab. 6-65: Summary

6.8.6 ML_OpenCyclicTorqueChannel

Brief description The function block switches the axis to the mode 'Cyclic Torque Command' and opens the cyclic power command value channel.



When ML_OpenCyclicTorqueChannel is activated and the cyclic power command value channel is already open, an error is generated.

The function block has to be used in MotionTask (synchronous to the Sercos cycle). Otherwise, jerky axis motions can result.

When the "Execute" input is deactivated, all outputs are set to FALSE. The cyclic power command value channel remains open if it has been open before.

If the function block was interrupted by another Motion function block, the function block is aborted and the cyclic force command value channel is closed.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Motion

Tab. 6-66: Library assignment

Interface description

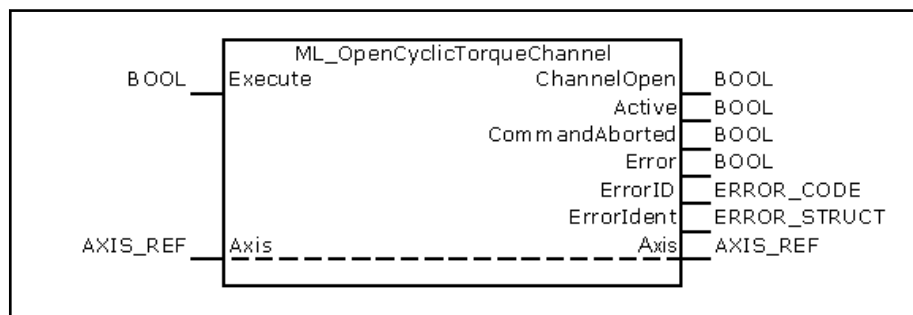


Fig. 6-90: ML_OpenCyclicTorqueChannel function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge.
VAR_OUTPUT	ChannelOpen	BOOL	Indicates that the cyclic force command value channel is open and that the axis follows the force value of the last cycle.
	Active	BOOL	Indicates that the function block sets the axis to the "Cyclic Torque Command" mode.
	CommandAborted	BOOL	Command was aborted by another command.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the concrete axis

Tab. 6-67: Interface of the ML_OpenCyclicPoswerChannel function block

Troubleshooting

The function block ML_OpenCyclicTorqueChannel generates the following error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.8.7 ML_WriteCyclicTorque

Brief description The function block ML_WriteCyclicTorque writes a force value for an axis. If the cyclic force command value channel is open, the force value is applied to the axis in the next cycle.



To open the cyclic force command value channel, use the function block [ML_OpenCyclicPositionChannel](#), page 257,.

The function block has to be used in a PLC task (synchronous to the Sercos cycle). Otherwise, jerky axis motions can result.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-68: Library assignment

Interface description

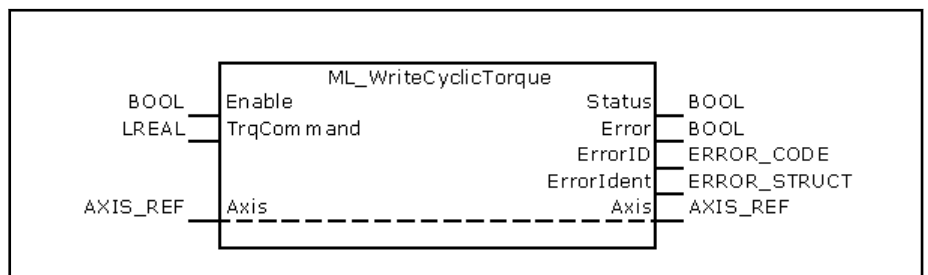


Fig. 6-91: ML_WriteCyclicTorque

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	As long as the "Enable" output is TRUE, "TrqCommand" is written on the axis. The force value only becomes effective when the cyclic power command value channel is open.
	TrqCommand	LREAL	Indicates the force value that the axis should approach in the next cycle. The force value only becomes effective when the cyclic power command value channel is open.
VAR_OUTPUT	Status	BOOL	Indicates that the force value has been written on the axis. The force value only becomes effective when the cyclic power command value channel is open.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the concrete axis

Tab. 6-69: Interface of the ML_WriteCyclicTorque function block

Special features of the TrqCommand input

1. The values of the "TrqCommand" input become effective in the next cycle if the cyclic power command value channel is open.
2. The value of the "TrqCommand" input has the scaling of the assigned axis.

Troubleshooting

The function block ML_WriteCyclicTorque generates the following error messages:

- [MLC_TABLE](#), page 360 - error when accessing the axis/control.

6.8.8 ML_OpenCyclicVelocityChannel

Brief description The function block switches the axis to the mode 'Cyclic Velocity Command' and opens the cyclic velocity command value channel.



When ML_OpenCyclicVelocityChannel is activated and the cyclic velocity command value channel is already open, an error is generated.

The function block has to be used in MotionTask (synchronous to the Sercos cycle). Otherwise, jerky axis motions can result.

When the "Execute" input is deactivated, all outputs are set to FALSE. The cyclic velocity command value channel remains open if it has been open before.

If the function block was interrupted by another Motion function block, the function block is aborted and the cyclic velocity command value channel is closed.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-70: Library assignment

Interface description

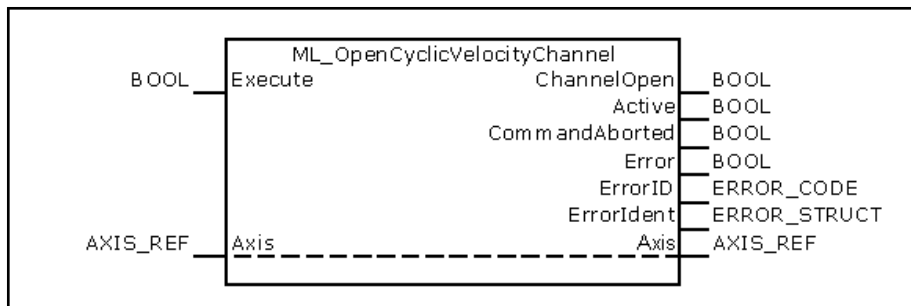


Fig. 6-92: ML_OpenCyclicVelocityChannel function block

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge.
VAR_OUTPUT	ChannelOpen	BOOL	Indicates that the cyclic velocity command value channel is open and that the axis follows the velocity value of the last cycle.
	Active	BOOL	Indicates that the function block set the axis to the "Cyclic Velocity Command" mode.
	CommandAborted	BOOL	Command was aborted by another command.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the concrete axis

Tab. 6-71: Interface of the ML_OpenCyclicVelocityChannel function block

Troubleshooting The function block ML_OpenCyclicVelocityChannel generates the following error messages:

- [MLC_TABLE](#), page 360 - error when accessing the axis/control.

6.8.9 ML_WriteCyclicVelocity

Brief description The function block ML_WriteCyclicVelocity writes a velocity value for an axis. If the cyclic velocity command value channel is open, the velocity value is applied to the axis in the next cycle.



To open the cyclic velocity command value channel, use the function block [ML_OpenCyclicPositionChannel](#), page 257,.

The function block has to be used in a PLC task (synchronous to the Sercos cycle). Otherwise, jerky axis motions can result.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Motion

Tab. 6-72: Library assignment

Interface description

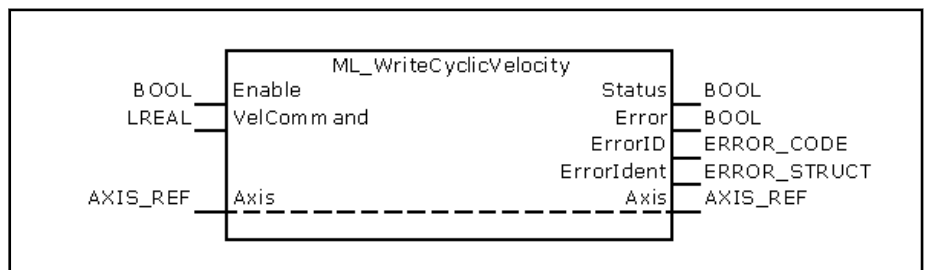


Fig. 6-93: ML_WriteCyclicVelocity

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	As long as the "Enable" output is TRUE, "VelCommand" is written on the axis. The velocity value only becomes effective when the cyclic velocity command value channel is open.
	VelCommand	LREAL	Indicates the velocity value to be directed at by the axis in the next cycle. The velocity value only becomes effective when the cyclic velocity command value channel is open.
VAR_OUTPUT	Status	BOOL	Indicates that the velocity value has been written on the axis. The velocity value only becomes effective when the cyclic velocity command value channel is open.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the concrete axis

Tab. 6-73: Interface of the ML_WriteCyclicVelocity function block

Special features of the VelCommand input

1. The values of the "VelCommand" input become effective in the next cycle if the cyclic velocity command value channel is open.
2. The value of the "VelCommand" input has the scaling of the assigned axis.

Troubleshooting

The function block ML_WriteCyclicVelocity generates the following error messages:

- [MLC_TABLE](#), page 360 - error when accessing the axis/control.

6.9 Function blocks for controlled motion

6.9.1 General information on controlled motion

This chapter describes how to provide the complete control for the cyclic command value specification to the PLC. This is often necessary to implement control algorithms across axes, e.g. gantry or synchronization networks. During this command value specification, the axis may not receive commands from other function blocks, e.g. MC_MoveAbsolute.



The function blocks for the controlled motions require the **function package MOT** of the control

The controlled motion is started with the function block ML_ControlOn. This function block may be called from the following states: 'Standstill', 'Continuous Motion', 'Discrete Motion' and 'Synchronized Motion'. When this function block is called, the axis switches into the "Controlled Motion" state. The state is completed with the function block ML_ControlOff. Changing into "Continuous Motion" state.

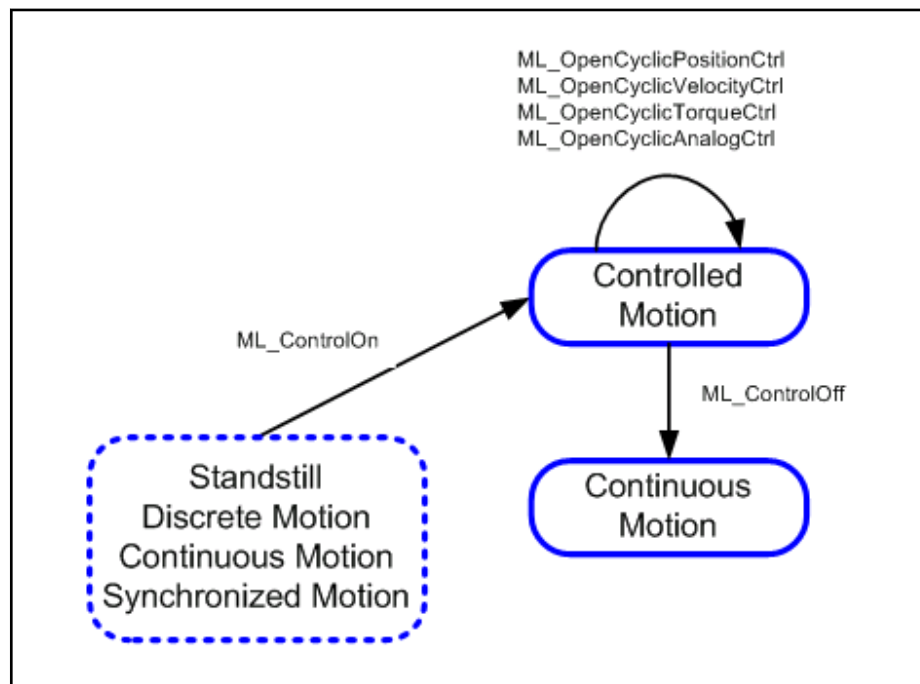


Fig. 6-94: Integrating the "Controlled Motion" state into the PLCopen state machine

Only Motion function blocks of type ML_OpenCyclic*Ctrl are allowed while an axis is in the "Controlled Motion" state.

The following blocks are available:

Function blocks for starting and ending controlled motion

- [ML_ControlOn](#), page 267
- [ML_ControlOff](#), page 268

Function block to write the current values of a PLC controller to the controller axis of the MLC

- [ML_ControlWriteActualData](#), page 269

Analog command value channel, operating mode "Cyclic Analog Command"

- [ML_OpenCyclicAnalogChannelCtrl](#), page 270
- [ML_WriteCyclicAnalogCtrl](#), page 271

Position command value channel, operation mode "Cyclic Position Command"

- [ML_OpenCyclicPositionChannelCtrl](#), page 272
- [ML_WriteCyclicPositionCtrl](#), page 273

Power command value channel, operation mode "Cyclic Torque Command"

- [ML_OpenCyclicTorqueChannelCtrl](#), page 274
- [ML_WriteCyclicTorqueCtrl](#), page 275

Velocity command value channel, operation mode "Cyclic Velocity Command"

- [ML_OpenCyclicVelocityChannelCtrl](#), page 276
- [ML_WriteCyclicVelocityCtrl](#), page 277

The supported function blocks can vary depending on axis type.

6.9.2 ML_ControlOn

Brief description The function block ML_ControlOn is used to provide the complete control for the cyclic command value specification to the PLC. When this function block is activated, the axis state changes to "Controlled Motion". The "Controlled Motion" state be completed using with the function block [ML_ControlOff](#) on page 268.



If ML_ControlOn is activated, the last Motion command remains effective until a function block of type ML_OpenCyclic*Ctrl was executed.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Controlled Motion

Tab. 6-74: Library assignment

Interface description

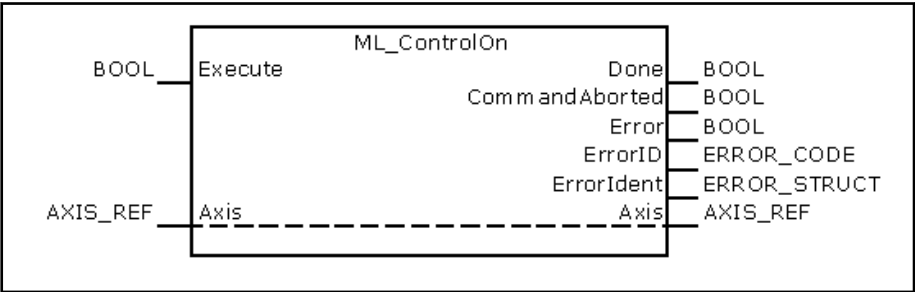


Fig. 6-95: FB ML_ControlOn

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge.
VAR_OUTPUT	Done	BOOL	Command execution successful. The axis has switched into the "Controlled Motion" state.
	CommandAborted	BOOL	Command was aborted by another command.
	Error	BOOL	Indicates that an error has occurred in the function block instance.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the triggered axis

Tab. 6-75: Interface of FB MC_ControlOn

Troubleshooting The ML_ControlOn function block generates the following error messages:

- [MLC_TABLE](#), page 360 - error when accessing the axis/control.

6.9.3 ML_ControlOff

Brief description The function block ML_ControlOff is used to complete the cyclic control of an axis from the PLC. When this command is executed, the axis leaves the "Controlled Motion" state and is transferred into the "Continuous Motion" state. The last active command value is retained until a new Motion command is initiated.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Controlled Motion

Tab. 6-76: Library assignment

Interface description

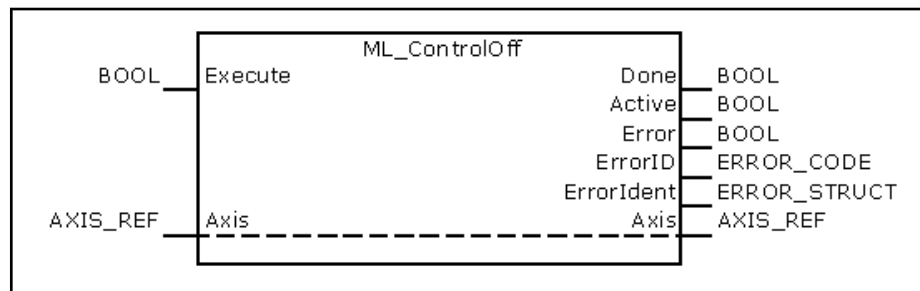


Fig. 6-96: FB ML_ControlOff

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge.
VAR_OUTPUT	Done	BOOL	Command execution successful. The axis has left the "Controlled Motion" state.
	Active	BOOL	It indicates that the deactivation process is running.
	Error	BOOL	Indicates that an error has occurred in the function block instance.
	ErrorID	ERROR_CODE	Brief note on error cause

	Name	Type	Comment
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the concrete axis.

Tab. 6-77: Interface of FB MC_ControlOff

Troubleshooting The ML_ControlOff function block generates the following error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.9.4 ML_ControlWriteActualData

Brief description The ML_ControlWriteActualData function block writes the current values of a PLC controller to the controller axis of the MLC.



The function block response on the "Error" output is not conforming to standards. Thus, in case of an error, the "Error" output is set to TRUE.

If the error is recovered, the "Error" changes from TRUE to FALSE automatically.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Controlled Motion

Tab. 6-78: Library assignment

Interface description

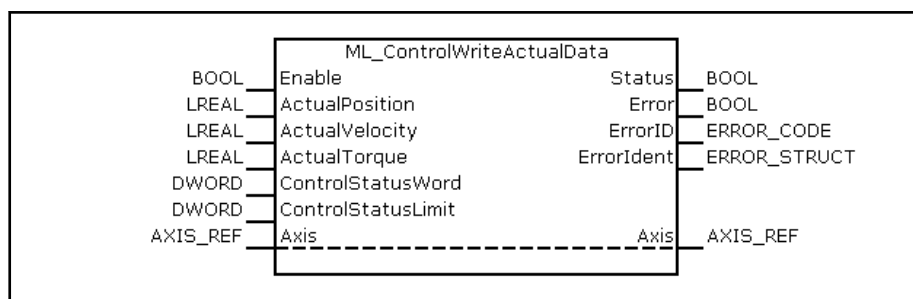


Fig. 6-97: ML_ControlWriteActualData

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	As long as the "Enable" output is TRUE,
	ActualPosition	LREAL	Actual position of the controller
	ActualVelocity	LREAL	Actual velocity of the controller
	ActualTorque	LREAL	Actual value torque/force of controller
	ControlStatusWord	DWORD	A-0-0019, Status word of controller
	ControlStatusLimit	DWORD	This value is currently not used
VAR_OUTPUT	Status	BOOL	It indicates that the actual values were transmitted to the axis.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause

	Name	Type	Comment
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the triggered axis.

Tab. 6-79: Interface of the FB ML_ControlWriteActualData

Troubleshooting The function block ML_ControlWriteActualData generates the following error messages in:

- [MLC_TABLE, page 360](#) - Error when accessing the axis/control.

6.9.5 ML_OpenCyclicAnalogChannelCtrl

Brief description The function block opens the cyclic analog command value channel for the specified axis. The function block is only allowed in the "Controlled Motion" state. A master axis can also be specified if the cyclic analog value for this axis has been calculated from the values of another axis. This specification prevents the programming of master axis recursions that can lead to reversed coupling effects.



The function block has to be used in a PLC task (synchronous to the Sercos cycle). Otherwise, jerky axis motions can result.

When the "Execute" input is deactivated, all outputs are set to FALSE. The cyclic analog command value channel remains open if it has been open before.

If the function block was interrupted by another Motion function block, the function block is aborted and the cyclic analog command value channel is closed.

This block is only available with HydraulicDrives.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Controlled Motion

Tab. 6-80: Library assignment

Interface description

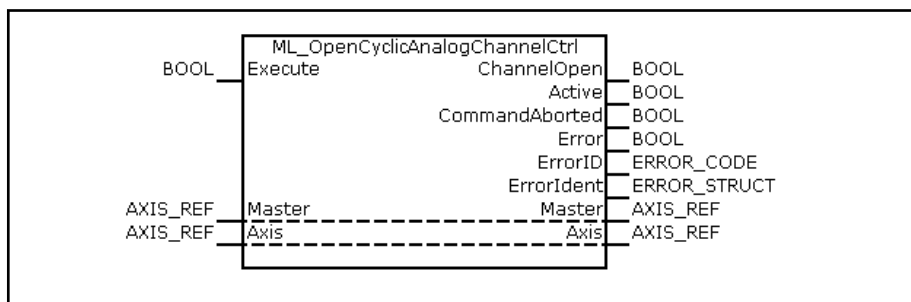


Fig. 6-98: FB ML_OpenCyclicAnalogChannelCtrl

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge.
VAR_OUTPUT	ChannelOpen	BOOL	Indicates that the cyclic analog command value channel is open and that the axis follows the analog value of the last cycle.
	Active	BOOL	It indicates that the cyclic analog command value channel is opened.

	Name	Type	Comment
	CommandAborted	BOOL	Command was aborted by another command.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Master	AXIS_REF	It contains information on the master axis.
	Axis	AXIS_REF	It contains information on the triggered axis.

Tab. 6-81: Interface of FB ML_OpenCyclicAnalogChannelCtrl

Troubleshooting The function block ML_OpenCyclicAnalogChannelCtrl generates the following error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.9.6 ML_WriteCyclicAnalogCtrl

Brief description The function block ML_WriteCyclicAnalogCtrl writes a cyclic analog value for an axis if the axis is in the "Controlled Motion" state. If the cyclic analog command value channel is open, the analog value is applied to the axis in the next cycle.



To open the cyclic analog command value channel, use the function block [ML_OpenCyclicAnalogChannelCtrl, on page 270](#).

The function block has to be used in a PLC task (synchronous to the Sercos cycle). Otherwise, jerky axis motions can result.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Controlled Motion

Tab. 6-82: Library assignment

Interface description

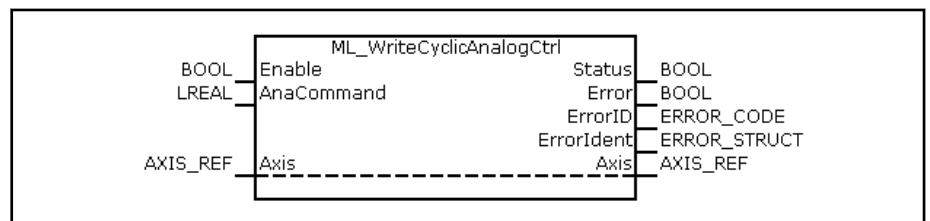


Fig. 6-99: ML_WriteCyclicAnalogCtrl

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	As long as the "Enable" output is TRUE, the analog value "Ana-Command" is written on the axis. The analog value only becomes effective when the cyclic analog command value channel is open.
	AnaCom- mand	LREAL	It specifies the analog value to be transferred to the axis. The analog value only becomes effective when the cyclic analog command value channel is open.
VAR_OUTPUT	Status	BOOL	It indicates that the analog value was transmitted to the axis. The analog value only becomes effective when the cyclic analog command value channel is open.

	Name	Type	Comment
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the triggered axis.

Tab. 6-83: Interface of the FB ML_WriteCyclicAnalogCtrl

Special features of the AnaCommand input

1. The value of the "AnaCommand" input becomes effective in the next cycle if the cyclic analog command value channel is open.
2. The value of the "AnaCommand" input does not have the scaling of the assigned axis but is active without unit.

Troubleshooting

The function block ML_WriteCyclicAnalogCtrl generates the following error messages in:

- [MLC_TABLE, page 360](#) - Error when accessing the axis/control.

6.9.7 ML_OpenCyclicPositionChannelCtrl

Brief description

The function block opens the cyclic position command value channel for the specified axis. The function block is only allowed in the "Controlled Motion" state. A master axis can also be specified if the cyclic position command value for this axis has been calculated from the values of another axis. This specification prevents the programming of master axis recursions that can lead to reversed coupling effects.



The function block has to be used in a PLC task (synchronous to the Sercos cycle). Otherwise, jerky axis motions can result.

When the "Execute" input is deactivated, all outputs are set to FALSE. The cyclic position command value channel remains open if it was opened before.

If the function block was interrupted by another Motion function block, the function block is aborted and the cyclic position command value channel is closed.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Controlled Motion

Tab. 6-84: Library assignment

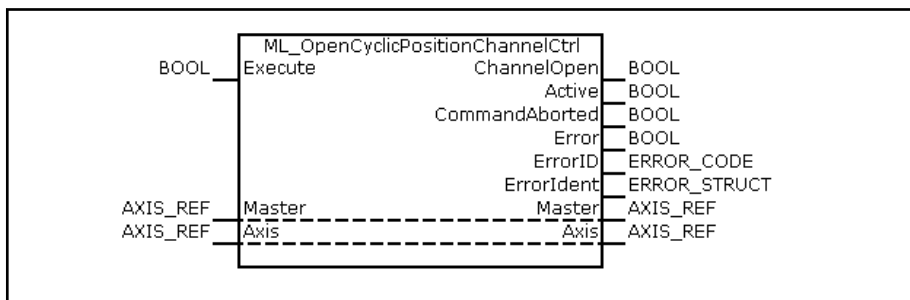
Interface description

Fig. 6-100: FB ML_OpenCyclicPositionChannelCtrl

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge.
VAR_OUTPUT	ChannelOpen	BOOL	It indicates that the cyclic position command value channel is open and that the axis follows the velocity value of the last cycle.
	Active	BOOL	It indicates that the cyclic position command value channel is opened.
	CommandAborted	BOOL	Command was aborted by another command.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Master	AXIS_REF	It contains information on the master axis.
	Axis	AXIS_REF	It contains information on the triggered axis.

Tab. 6-85: Interface of FB *ML_OpenCyclicPositionChannelCtrl***Troubleshooting**

The function block *ML_OpenCyclicPositionChannelCtrl* generates the following error messages:

- [MLC_TABLE](#), page 360 - error when accessing the axis/control.

6.9.8 ML_WriteCyclicPositionCtrl

Brief description

The function block *ML_WriteCyclicPositionCtrl* writes a cyclic position value for an axis if the axis is in the "Controlled Motion" state. If the cyclic position command value channel is open, the position value is applied to the axis in the next cycle.

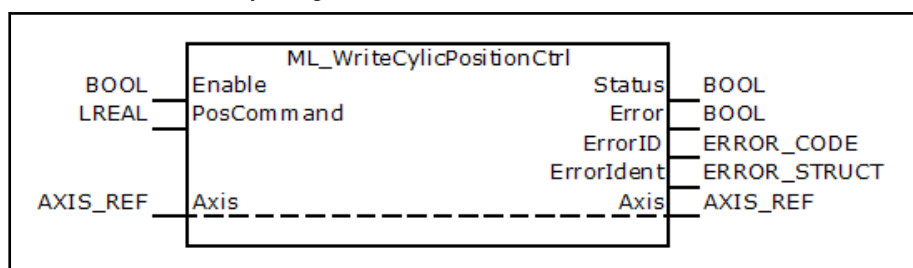


To open the cyclic position command value channel, use the function block [ML_OpenCyclicPositionChannelCtrl](#), on page 272.

The function block has to be used in a PLC task (synchronous to the Sercos cycle). Otherwise, jerky axis motions can result.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Controlled Motion

Tab. 6-86: Library assignment

Interface descriptionFig. 6-101: *ML_WriteCyclicPositionCtrl*

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	As long as the "Enable" output is TRUE, "PosCommand" is written on the axis. The position value only becomes effective when the cyclic position command value channel is open.
	PosCommand	LREAL	It specifies the position value to be transferred to the axis. The position value only becomes effective when the cyclic position command value channel is open.
VAR_OUTPUT	Status	BOOL	It indicates that the position value was transmitted to the axis. The position value only becomes effective when the cyclic position command value channel is open.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the triggered axis.

Tab. 6-87: Interface of the FB ML_WriteCyclicPositionCtrl

Special features of the PosCommand input

1. The value of the "PosCommand" input becomes effective in the next cycle if the cyclic position command value channel is open.
2. The value of the "PosCommand" input has the scaling of the assigned axis.

Troubleshooting

The function block ML_WriteCyclicPositionCtrl generates the following error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.9.9 ML_OpenCyclicTorqueChannelCtrl

Brief description

The function block opens the cyclic force/torque command value channel for the specified axis. The function block is only allowed in the "Controlled Motion" state. A master axis can also be specified if the cyclic force/torque value for this axis has been calculated from the values of another axis. This specification prevents the programming of master axis recursions that can lead to reversed coupling effects.



The function block has to be used in a PLC task (synchronous to the Sercos cycle). Otherwise, jerky axis motions can result.

When the "Execute" input is deactivated, all outputs are set to FALSE. The cyclic force/torque command value channel remains open if it has been open before.

If the function block was interrupted by another Motion function block, the function block is interrupted and the cyclic force/torque command value channel is closed.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Controlled Motion

Tab. 6-88: Library assignment

Interface description

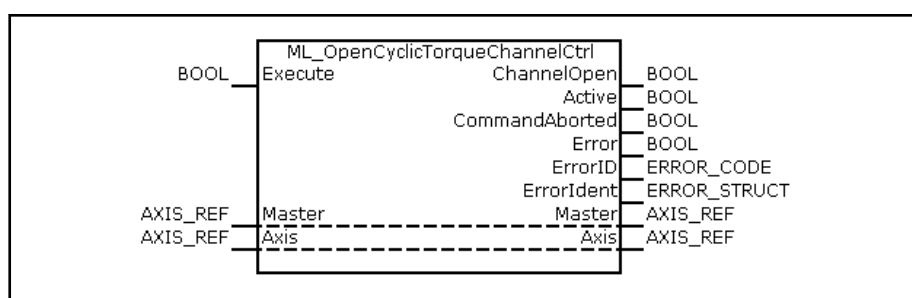


Fig. 6-102: FB ML_OpenCyclicTorqueChannelCtrl

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge.
VAR_OUTPUT	ChannelOpen	BOOL	Command execution successful. The cyclic force/torque command value channel was opened. The axis is following the force/torque value.
	Active	BOOL	It indicates that the cyclic force/torque command value channel is opened.
	CommandAborted	BOOL	Command was aborted by another command.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Master	AXIS_REF	It contains information on the master axis.
	Axis	AXIS_REF	It contains information on the triggered axis.

Tab. 6-89: Interface of ML_OpenCyclicTorqueChannelCtrl FB

Troubleshooting

The function block ML_OpenCyclicTorqueChannelCtrl generates the following error messages:

- [MLC_TABLE, page 360](#) - error when accessing the axis/control.

6.9.10 ML_WriteCyclicTorqueCtrl

Brief description

The function block ML_WriteCyclicTorqueCtrl writes a cyclic force/torque value for an axis if the axis is in the "Controlled Motion" state. Only if the cyclic force/torque command value channel is open, the force value become effective in the next cycle on the axis.



To open the cyclic torque/force command value channel, use the function block [ML_OpenCyclicTorqueChannelCtrl on page 274](#).

The function block has to be used in a PLC task (synchronous to the Sercos cycle). Otherwise, jerky axis motions can result.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Controlled Motion

Tab. 6-90: Library assignment

Interface description

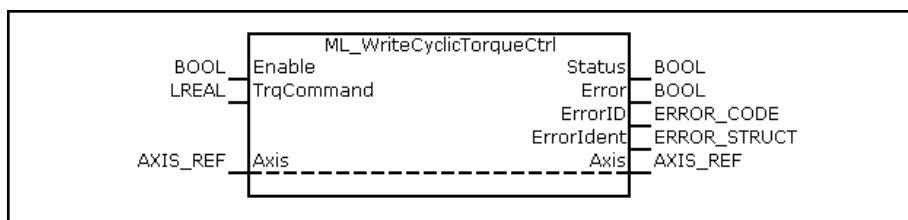


Fig. 6-103: ML_WriteCyclicTorqueCtrl

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	As long as the "Enable" output is TRUE, "TrqCommand" is written on the axis. The force/torque value only becomes effective when the cyclic force/torque command value channel is open.
	TrqCommand	LREAL	It specifies the force/torque value to be transferred to the axis. The force/torque value only becomes effective when the cyclic force/torque command value channel is open.
VAR_OUTPUT	Status	BOOL	It indicates that the force/torque value was transmitted to the axis. The force/torque value only becomes effective when the cyclic force/torque command value channel is open.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the triggered axis.

Tab. 6-91: Interface of FB ML_WriteCyclicTorqueCtrl

Special features of the TrqCommand input

1. The value of the "TrqCommand" input becomes effective in the next cycle if the cyclic force/torque command value channel is open.
2. The value of the "TrqCommand" input has the scaling of the assigned axis.

Troubleshooting

The function block ML_WriteCyclicTorqueCtrl generates the following error messages:

- [MLC_TABLE](#), page 360 - error when accessing the axis/control.

6.9.11 ML_OpenCyclicVelocityChannelCtrl

Brief description

The function block opens the cyclic velocity command value channel for the specified axis. The function block is only allowed in the "Controlled Motion" state. A master axis can also be specified if the cyclic velocity value for this axis has been calculated from the values of another axis. This specification prevents the programming of master axis recursions that can lead to reversed coupling effects.



The function block has to be used in a PLC task (synchronous to the Sercos cycle). Otherwise, jerky axis motions can result.

When the "Execute" input is deactivated, all outputs are set to FALSE. The cyclic velocity command value channel remains open if it has been open before.

If the function block was interrupted by another Motion function block, the function block is aborted and the cyclic velocity command value channel is closed.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen / Controlled Motion

Tab. 6-92: Library assignment

Interface description

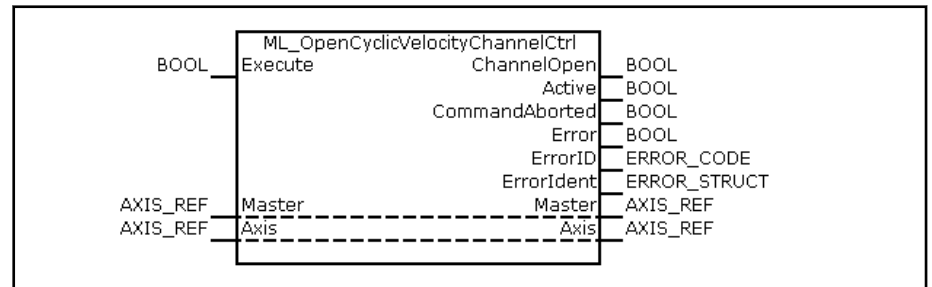


Fig. 6-104: FB ML_OpenCyclicVelocityChannelCtrl

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge.
VAR_OUTPUT	ChannelOpen	BOOL	Indicates that the cyclic velocity command value channel is open and that the axis follows the velocity value of the last cycle.
	Active	BOOL	It indicates that the cyclic velocity command value channel is opened.
	CommandAborted	BOOL	Command was aborted by another command.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Master	AXIS_REF	It contains information on the master axis.
	Axis	AXIS_REF	It contains information on the triggered axis.

Tab. 6-93: Interface of the ML_OpenCyclicVelocityChannelCtrl FB

Troubleshooting

The function block ML_OpenCyclicVelocityChannelCtrl generates the following error messages:

- [MLC_TABLE](#), page 360 - error when accessing the axis/control.

6.9.12 ML_WriteCyclicVelocityCtrl

Brief description

The function block ML_WriteCyclicVelocityCtrl writes a cyclic velocity value for an axis if the axis is in the "Controlled Motion" state. If the cyclic velocity command value channel is open, the velocity value is applied to the axis in the next cycle.



To open the cyclic velocity command value channel, use the function block [ML_OpenCyclicVelocityChannelCtrl](#), on page 276.

The function block has to be used in a PLC task (synchronous to the Sercos cycle). Otherwise, jerky axis motions can result.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen / Controlled Motion

Tab. 6-94: Library assignment

Interface description

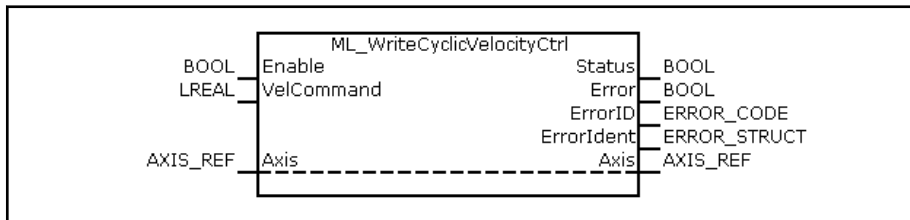


Fig. 6-105: ML_WriteCyclicVelocityCtrl

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	As long as the "Enable" output is TRUE, "VelCommand" is written on the axis. The velocity value only becomes effective when the cyclic velocity command value channel is open.
	VelCommand	LREAL	It specifies the velocity value to be transferred to the axis. The velocity value only becomes effective when the cyclic velocity command value channel is open.
VAR_OUTPUT	Status	BOOL	It indicates that the velocity value was transmitted to the axis. The velocity value only becomes effective when the cyclic velocity command value channel is open.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the triggered axis.

Tab. 6-95: Interface of the ML_WriteCyclicVelocityCtrl FB

Special features of the VelCommand input

1. The value of the "VelCommand" input becomes effective in the next cycle if the cyclic velocity command value channel is open.
2. The value of the "VelCommand" input has the scaling of the assigned axis.

Troubleshooting

The function block ML_WriteCyclicVelocityCtrl generates the following error messages:

- [MLC_TABLE](#), page 360 - error when accessing the axis/control.

6.10 Parameter access

6.10.1 Access to data of an axis, control, kinematics (axis group) or another object



The basic function package of the IndraLogic XLC/IndraMotion MLC 14VRS system is sufficient for the function blocks to parameters.

Structure of and access to parameters of the PLC program

The parameter data of an axis, a control or a programmable limit switch can be accessed by read or write access. The following parameter types exist in IndraMotion MLC:

- C-parameters (control parameters), referring to an actual control affecting the axes.
- A-parameters (axis parameters), each referring to one particular axis.
- K-parameters (kinematic parameters) that relate to a kinematics (axis group).
(see ML_Robot, Accessing Kinematic Data (Axis Group)).
- M parameters (touch probe parameters), each relating to one touch probe. They can only be accessed via PLCopen function blocks.
- N parameters (pls parameters), each referring to one programmable limit switch.
- O parameters (oscilloscope parameters), each referring to one oscilloscope function.

In addition to these parameters, there are other groups which relate to devices on a specific bus (S-parameters, Sercos interface) or to a specific product (P-parameters).

The structure of all IndraMotion parameters is identical:

Example: A-0-0001, with "A" as a parameter type ID which is followed by the group number "0" and the four-digit parameter number "0001".

Based on the parameter number, access to these parameters is possible in the PLC program.

A-, ..., O-, P- and S-parameter declaration files for global constants are available for the user in the ML_Base.library.

Declaration files for Global Constants

- [GVL_A_Param_ID, page 130](#)
- [GVL_C_Param_ID, page 130](#)
- [GVL_M_Param_ID, page 130](#)
- [GVL_N_Param_ID, page 131](#)
- [GVL_O_Param_ID, page 131](#)
- [GVL_P_Param_ID, page 132](#)
- [GVL_S_Param_ID, page 132](#)

If the parameter number is known:

A-parameters address constants

```
VAR_GLOBAL CONSTANT
  FP_A_0_0000: MB_IDN:= 4294967296 ; (* Dummy A parameter *)
  FP_A_0_0001: MB_IDN:= 4294967297 ; (* Logical axis number *)
  FP_A_0_0002: MB_IDN:= 4294967298 ; (* Axis name *)
  (* ..... *)
END_VAR
```

As an alternative to the constant, the number can also be attached to the input "ParameterNumber" of the respective function block.

Offsets der Parameter

Parameter	Decimal	Hexadecimal
S-0-0000	0	16#0
P-0-0000	32768	16#8000
A-0-0000	4294967296	16#1 0000 0000
C-0-0000	8589934592	16#2 0000 0000

Parameter	Decimal	Hexadecimal
K-0-0000	184683593728	16#2B 0000 0000
M-0-0000	193273528320	16#2D 0000 0000
N-0-0000	197568495616	16#2E 0000 0000
O-0-0000	201863462912	16#2F 0000 0000

Tab. 6-96: Parameter offsets

Several axes can be actuated with the control.

Thus, the function blocks are provided with an address input (VAR_IN_OUT CONSTANT) of type AXIS_REF, RIL_CommonTypes.library which has to be assigned as follows:

Access to A-parameters:

- The address identifier is automatically generated by IndraWorks MLC (axis name/axis number), type AXIS_REF.

Access to the N Parameters:

- The address identifier is automatically generated by IndraWorks MLC or can instead be defined by the user (object name/object number), type AXIS_REF.

Access to O parameters:

- The address identifier is automatically generated by IndraWorks MLC or can instead be defined by the user (object name/object number), type AXIS_REF.

Access to C-parameters:

- Use the address identifier of an arbitrary axis of this control or declaration with values.
 - CntrlNo:= LOCAL_CNTRL or CNTRL_1...
 - AxisNo:= NO_OBJECT.

References at axis input, example: MB_ReadParameter

```

VA1:AXIS_REF:=(CntrlNo:=LOCAL_CNTRL,AxisNo:=AXIS_1)
               (*A parameters, axis VA1, axis number 1*);

MLC_A: AXIS_REF:=(CntrlNo:=LOCAL_CNTRL,AxisNo:=NO_OBJECT)
               (* C parameters, control MLC_A *);

PLS_A: AXIS_REF:=(CntrlNo:=LOCAL_CNTRL,AxisNo:=PLS_1)
               (*N parameters, programmable limit switch PLS_A, obj.no.1*);

OSZ1: AXIS_REF:=(CntrlNo:=LOCAL_CNTRL,AxisNo:=OSCI_CHANNEL_1)
               (*O parameters, oscilloscope OSZ1, obj.no.1*);

```

6.10.2 Time required for data transport with read and WriteParameter function blocks

The MB_Read***Parameter and the MB_Write***Parameter function blocks are responsible for the data transfer between PLC and the Motion component of the control or drives.

The respective function block accesses the respective parameter. The subsequent times are average access times and are valid for IndraMotion MLC L65.1 (PlcTask: 1ms (cyclic), Motion and Sercos cycle time 2ms).

Parameters	Number of bytes	Function block	average access time	
			MPB07	MPB18
S-0-0256	UDINT, 4	MB_ReadParameter	18ms	3.8ms
P-0-0072	List, 1024	MB_WriteListParameter	284ms	20.5ms
P-0-0072	List, 1024	MB_ReadListParameter	150ms	3ms
A-0-0224	REAL, 4	MB_ReadRealParameter	3ms	3ms
A-0-0101	LREAL, 8	MB_ReadLrealParameter	3ms	3ms
A-0-0023	UDINT, 4	MB_ReadParameter	3ms	2.9ms
C-0-0852	LREAL, 8	MB_WriteLrealParameter	2 ms	2 ms
A-0-3020	LREAL, 8	MB_WriteLrealParameter	25ms	30.5ms
A-0-0030	REAL, 4	MB_WriteRealParameter	4 ms	4 ms
A-0-0002	STRING	MB_ReadStringParameter	2 ms	2 ms

Tab. 6-97: Average access times on parameters depending on the drive type

6.10.3 MB_Command

Brief description The MB_Command function block allows the command execution. The command can be started, controlled, observed or aborted.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-98: Library assignment

Interface description

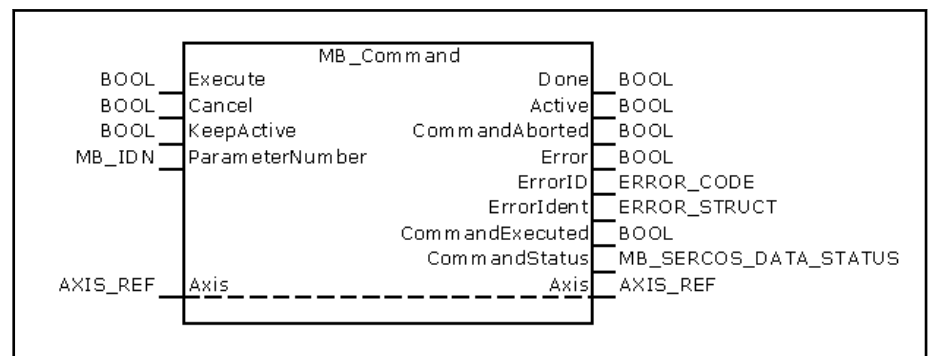


Fig. 6-106: FB MB_Command

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
	Cancel	BOOL	Aborts the running command at a rising edge
	KeepActive	BOOL	Keeps the command active after processing the status graph
	Parameter number	MB_IDN	Number of the parameter (refer to Access to data of an axis, control, kinematics (axis group) or another object, page 278)

	Name	Type	Comment
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data after preprocessing running
	CommandAborted	BOOL	Command successfully aborted and reset
	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 284)
	CommandExecuted	BOOL	Command successfully completed
	CommandStatus	MB_SERCOS_DATA_STATUS	Status of the selected parameter MB_SERCOS_DATA_STATUS , page 82
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-99: Interface of FB MB_Command

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.



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).

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 a successful execution. If the input "KeepActive" is deactivated, 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 active processing of the command is displayed at the "Active" output and the respective date at the "CommandStatus" output.

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.

If an error occurs while processing the command, this is signaled by the "Error" output and by "ErrorID", "ErrorIdent" for at least one call.

CommandStatus:

- 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 min./max. range

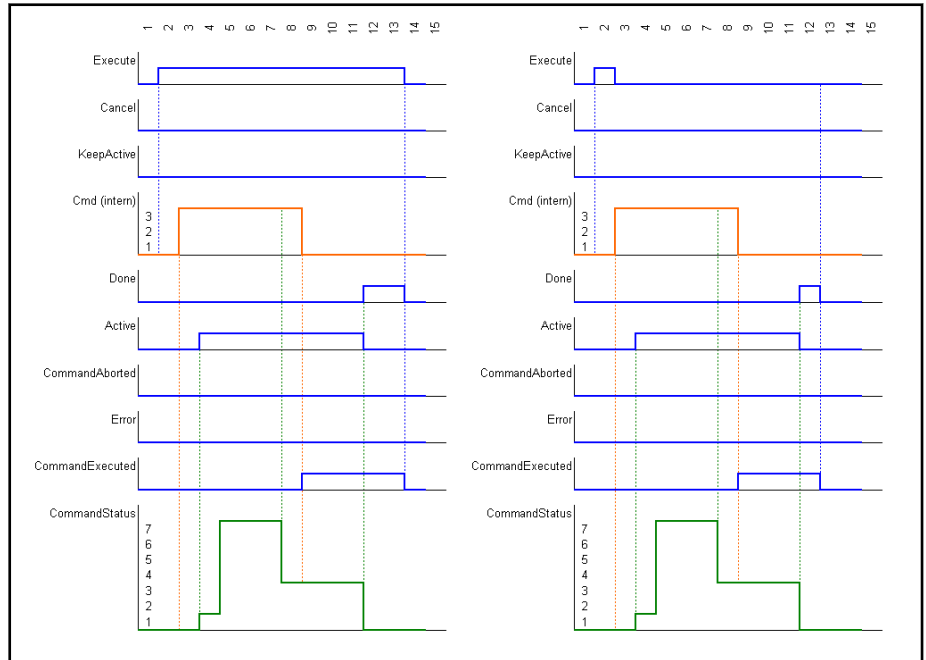
Signal-time diagram

Fig. 6-107: Signal-time diagram for signal flow "Execute-Done"

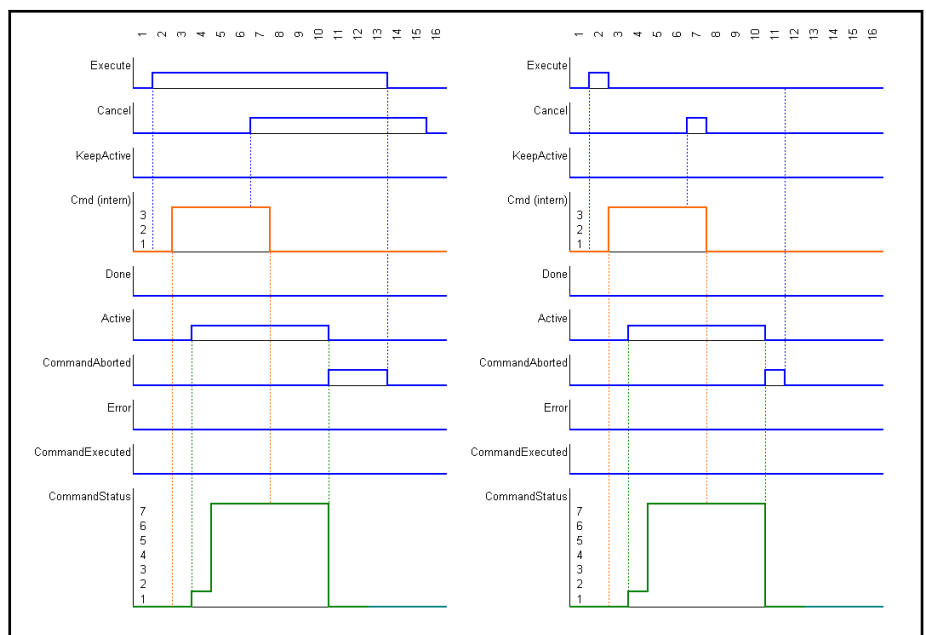


Fig. 6-108: Signal-time diagram for signal flow "Execute-Cancel-CommandAborted"

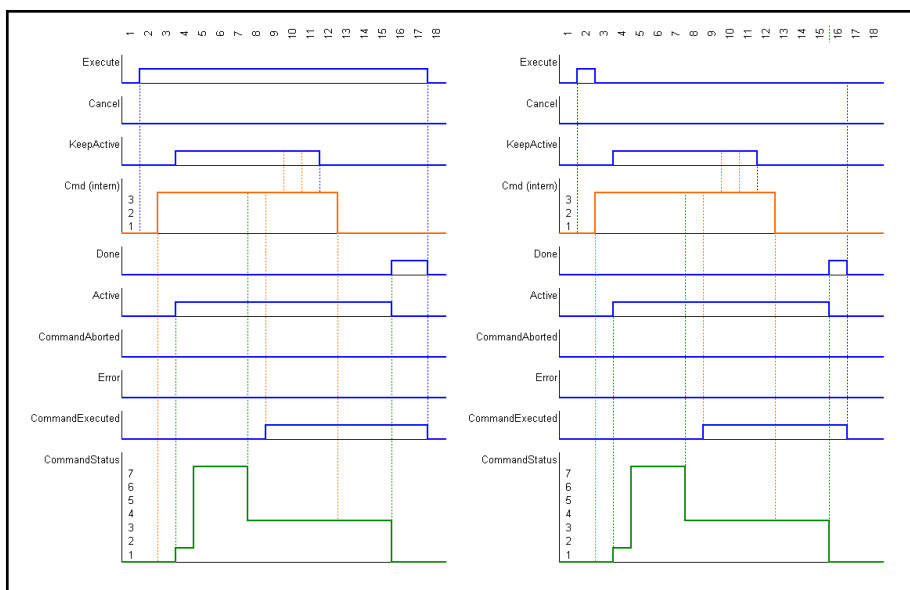


Fig. 6-109: Signal-time diagram for signal flow "Execute-KeepActive-Done"

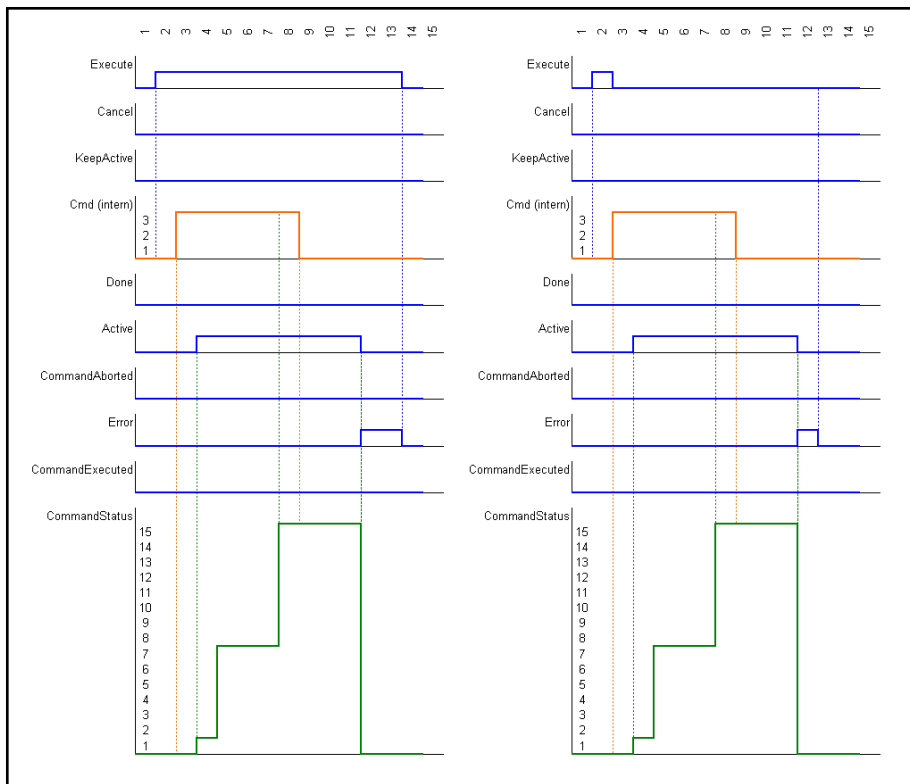


Fig. 6-110: Signal-time diagram for signal flow "Execute-Error"

Troubleshooting

The function block MB_Command generates the following types of error messages:

ERROR_TABLEs

- MLC_TABLE, table number 16#0030
 - MB_ReadSercosDataStatus,
 - MB_ReadSercosAttribute.
 - MB_ReadParameter,

- MB_WriteParameter,
- INDRV_TABLE, table number 16#0070
- F_RELATED_TABLE, table number 16#0170

ErrorID	Table	Additional1	Additional2	Description
INPUT_RANGE_ERROR	F_RELATED_TABLE	16#0002	16#0001	Retriggering not supported
INPUT_RANGE_ERROR	F_RELATED_TABLE	16#0002	16#0002	Cancel/ Keep Active not permitted for ctrl commands
INPUT_RANGE_ERROR	F_RELATED_TABLE	16#0002	16#0003	Parameter is no command
STATE_MACHINE_ERROR	F_RELATED_TABLE	16#0006	State	Invalid status of the function block
DEVICE_ERROR	F_RELATED_TABLE	16#1701	16#0000	Error during command processing (control)
DEVICE_ERROR	INDRV_TABLE	Value	16#0000	Error during command processing (drive) Value contained in S-0-0390.

Tab. 6-100: Error numbers caused by MB_Command

Example on automatic Reset

Executing command S-0-0099

In the following example, the Sercos command S-0-0099.0.0, C0500 Reset state class 1, is executed.

Variable definition

```
VAR
    fbMB_Command:    MB_Command;
END_VAR
```

Executing program code S-0-0099

```
fbMB_Command.Execute:=      TRUE;
fbMB_Command.ParameterNumber:= FP_S_0_0099;

fbMB_Command(Axis:=Drive1);

IF fbMB_Command.Done THEN
    ; // Command executed successfull
END_IF
```

Error is deleted after successful execution (if the error caused was eliminated before).

Example on "Cancel"

Executing command S-0-0139

In the following, the Sercos command S-0-0139.0.0, C1600 Command parking axis is executed for 5 seconds and than stopped using the "Cancel" input. The command can only be executed in Sercos phase 2 or in the parameterization mode PM.

Variable definition

```
VAR
    fbMB_Command:    MB_Command;
    fbTON:            TON;
END_VAR
```

Executing program code S-0-0139

```
fbMB_Command.Execute:=      TRUE;
fbMB_Command.ParameterNumber:= FP_S_0_0139;

IF fbMB_Command.CommandStatus = MB_SERCOS_DATA_STATUS.MB_SERCOS_DATA_IN_PROCESS THEN
```

ML_PLCOpen.library

```

// command started successfull, start timer
fbTON(IN:= TRUE, PT:= T#5S );
// Cancel the command after 5 seconds
fbMB_Command.Cancel := fbTON.Q;
END_IF

fbMB_Command(Axis:=Drive1);

IF fbMB_Command.Done THEN
; // finished
END_IF

```

Example on KeepActive

After the successful execution, the command "parking axis" is stopped again.

Executing command S-0-0149

In the following, the Sercos command S-0-0149.0.0, C1300 Command travel to fixed stop is executed and then canceled by resetting the "KeepActive" input.

The command can only be executed using drive enabled.

Variable definition

```

VAR
  fbMB_Command:    MB_Command;
  fbTON:           TON;
END_VAR

```

Executing program code S-0-0149

```

fbMB_Command.Execute:=      TRUE;
fbMB_Command.ParameterNumber:= FP_S_0_0149;

IF fbMB_Command.CommandStatus = MB_SERCOS_DATA_STATUS.MB_SERCOS_DATA_END_OK THEN
  // command started successfull, start timer
  fbTON(IN:= TRUE, PT:= T#5S );
END_IF
// Cancel the command after 5 seconds
fbMB_Command.KeepActive := NOT fbTON.Q;

fbMB_Command(Axis:=Drive1);

IF fbMB_Command.Done THEN
; // finished
END_IF

```

After the successful execution, the command Move to fixed stop is stopped again.

6.10.4 MB_Command, special features of Sercos-based axes

A command supported by parameters can be triggered in three different ways (Example A-0-1030, Command: Clear all axis errors):

- **Starting from IndraWorks:** "Clear error" button in the axis context menu **Diagnostics ► Clear Error**.
- **With the parameter editor:** Enter the binary value "2#00000000000000011" (=16#0003) into the parameter. This triggers the error deletion process.

After the error is deleted, the PLC automatically resets the parameter value back to "2#0000000000000000".

- **From the PLC:** Via the "MB_Command" function block.

6.10.5 MB_ReadLintParameter

Brief description The function block returns the value of a user-specific parameter. The return value is represented as LINT 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
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-101: Library assignment

Interface description

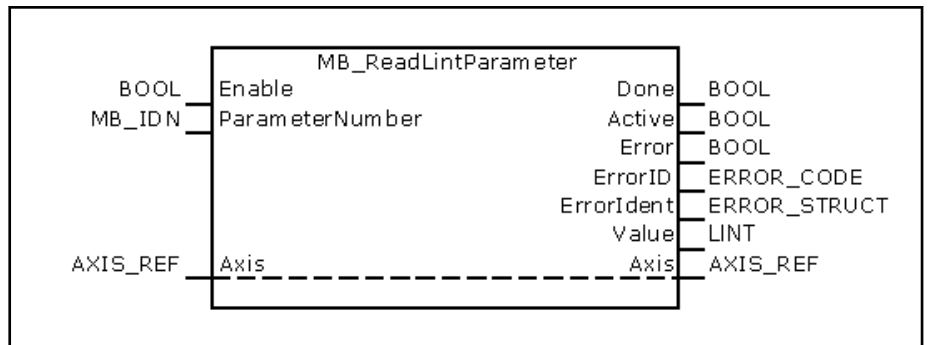


Fig. 6-111: MB_ReadLintParameter FB

	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 (refer to Access to data of an axis, control, kinematics (axis group) or another object, page 278)
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 288)
	Value	LINT	8 byte value, represented as LINT
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the actual axis and/or the addressed object.

Tab. 6-102: Interface of the MB_ReadLintParameter function block

Signal-time diagram

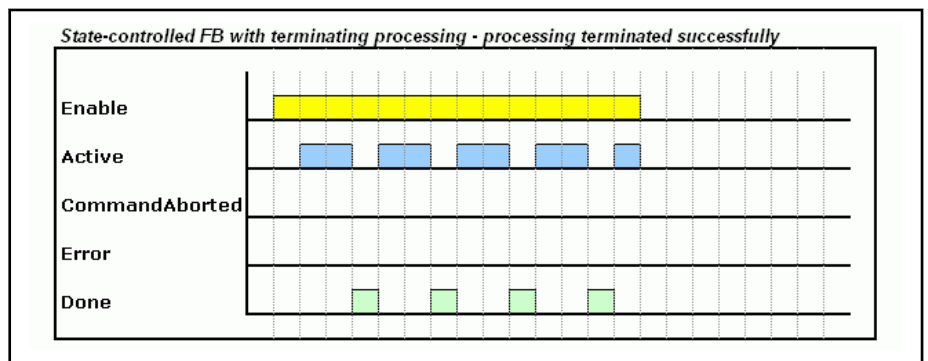


Fig. 6-112: Default signal-time diagram



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_ReadLintParameter 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

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 357 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 360 - error when accessing the axis/the control.

6.10.6 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.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCOpen/parameters

Tab. 6-103: Library assignment

Interface description

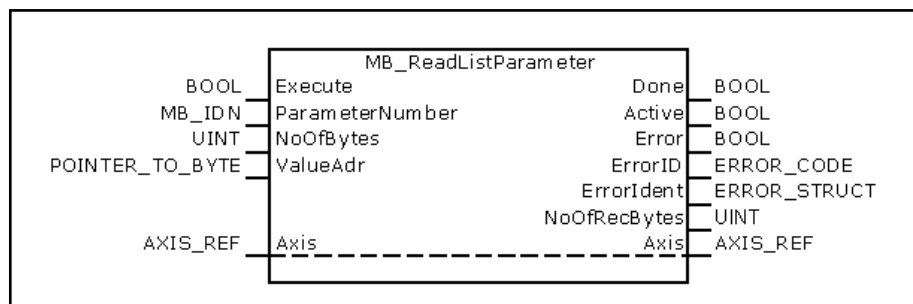


Fig. 6-113: MB_ReadListParameter FB

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Reading the parameter is triggered by the positive edge at "Execute"
	Parameter number	MB_IDN	Number of the parameter (refer to Access to data of an axis, control, kinematics (axis group) or another object, page 278)
	NoOfBytes	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	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information (" Troubleshooting " on page 291)
	NoOf Re-cBytes	UINT	Number of bytes actually read
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it

Tab. 6-104: Interface of the MB_ReadListParameter function block

Signal-time diagram

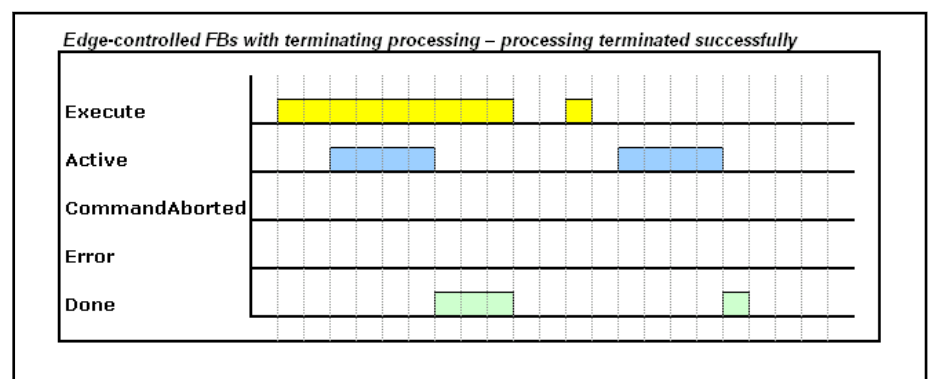


Fig. 6-114: Default signal-time diagram

Program example: MB_ReadList-Parameter

The program example shows the reading of a list parameter (C-0-2001, ELS, point table1) of the MLC with 1,024 values (DINT) and the saving to the ReadListArray.

Declaration of "drive1" as constant in MlcVarGlobal

```
(* Automatically generated code by MLC. *)
(* Please don't edit, the code will be overwritten. For usercode use UserVarGlobal! *)
{attribute 'linkalways'}
VAR_GLOBAL CONSTANT

(*Reale Achse "drive1" mit Achsnummer 1*)
(*Real axis "drive1" with axis number 1*)

drive1: AXIS_REF := (CntrlNo:=LOCAL_CNTRL, AxisNo:=AXIS_1);
(* ... *)
END_VAR
```

Declaration

```
PROGRAM ReadListParameter
```

```
VAR
  ReadListArray: ARRAY [0..1023] OF DINT; (* Data array for the received bytes *)

  FB_ReadListParameter: MB_ReadListParameter;
  bExecute_RLP:        BOOL;              (* Start ReadListParameter *)
  diParameterNo_RLP:  MB_IDN:= FP_C_0_2001; (* MLC Parameter ControlCAM1 der MLC *)
  uiNoOfBytes_RLP:    UINT := 4096;      (* DINT := 4 Bytes *)
  bDone_RLP:          BOOL;              (* TRUE := FB ready *)
  bActive_RLP:         BOOL;              (* The FB works *)
  bError_RLP:          BOOL;              (* Error flag *)
  enErrorID_RLP:       ERROR_CODE;        (* Error class *)
  stErrorIdent_RLP:    ERROR_STRUCT;      (* Error definition *)
  uiNoOfRecBytes_RLP:  UINT;              (* Number of received bytes *)
END_VAR
```

Program example in the function block diagram

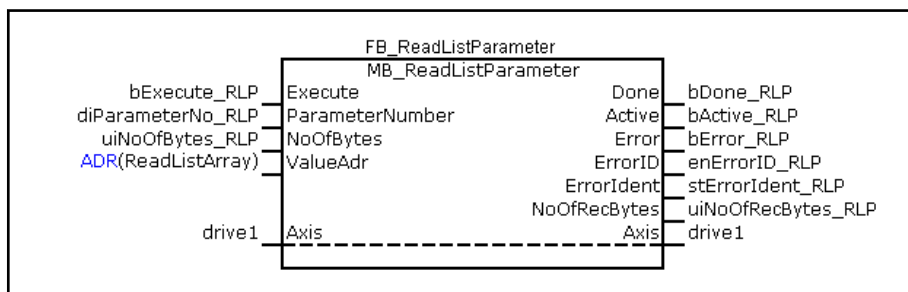


Fig. 6-115: FBD program example: ReadListParameter

Program example in structured text

```
FB_ReadListParameter(
  Execute:= bExecute_RLP, (* Start ReadListParameter *)
  ParameterNumber:= diParameterNo_RLP, (* MLC Parameter ControlCAM1 *)
  NoOfBytes:= uiNoOfBytes_RLP, (* DINT := 4 Bytes *)
  ValueAdr:= ADR(ReadListArray), (* Address of ReadListArray *)
  Axis:= drive1, (* Address const.: drive1 *)
  Done=> bDone_RLP, (* TRUE := FB ready *)
  Active=> bActive_RLP, (* The FB works *)
  Error=> bError_RLP, (* Error occurs *)
  ErrorId=> enErrorID_RLP, (* Error class *)
  ErrorIdent=> stErrorIdent_RLP, (* Error definition *)
  NoOfRecBytes=> uiNoOfRecBytes_RLP); (* Number of received bytes *)
```



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_ReadListParameter 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

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 357 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 360 - error when accessing the axis/the control.

6.10.7 MB_ReadLrealListParameter

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.



In contrast to [MB_ReadListParameter](#), page 288 in which an ARRAY of BYTE with n bytes is read, only one ARRAY of LREAL with m BYTES that is m/8 LREAL is read in this case.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-105: Library assignment

Interface description

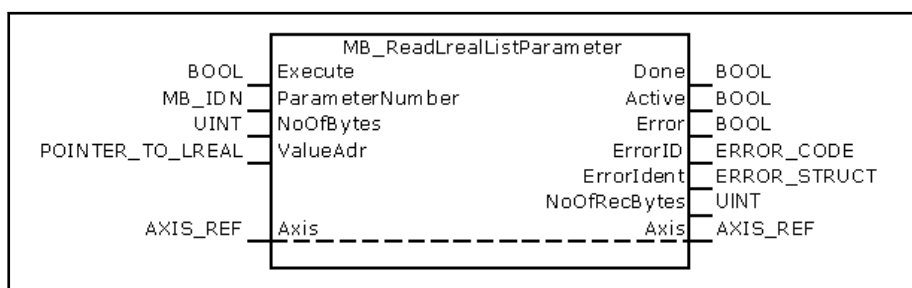


Fig. 6-116: MB_ReadLrealListParameter

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Reading the parameter is triggered by the positive edge at "Execute"
	Parameter number	MB_IDN	Number of the parameter (refer to Access to data of an axis, control, kinematics (axis group) or another object , page 278)
	NoOfBytes	UINT	Maximum number of data to be read in bytes
	ValueAdr	POINTER TO LREAL	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

	Name	Type	Comment
	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 293)
	NoOf RecBytes	UINT	Number of bytes actually read
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it

Tab. 6-106: Interface of the MB_ReadLrealListParameter function block

Signal-time diagram

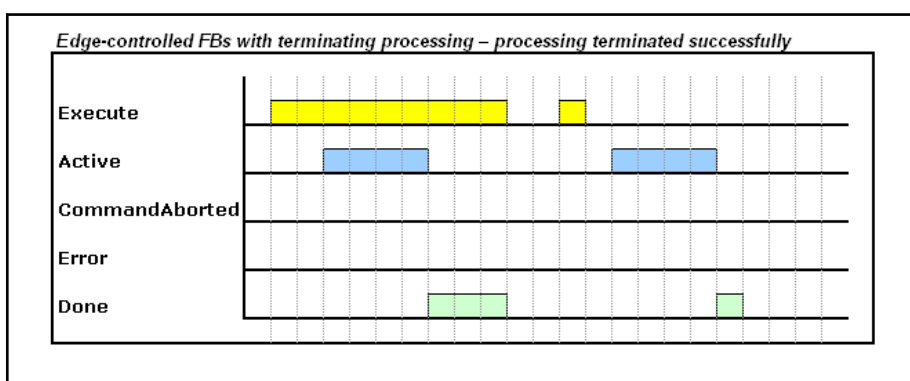


Fig. 6-117: Default signal-time diagram

Program example MB_ReadLrealListParameter

The program example shows the reading of a list parameter (A-0-3027, FlexProfile 0, list of slave axis starting velocities) of the MLC with 1,024 values (LREAL) and the saving to the ReadListArray.

Declaration of "drive1" as constant in MlcVarGlobal

```
(* Automatically generated code by MLC. *)
(* Please don't edit, the code will be overwritten. For usercode use UserVarGlobal! *)
{attribute 'linkalways'}
VAR_GLOBAL CONSTANT

(*Reale Achse "drive1" mit Achsnummer 1*)
(*Real axis "drive1" with axis number 1*)

drive1: AXIS_REF := (CntrlNo:=LOCAL_CNTRL,AxisNo:=AXIS_1);
(* ...*)
END_VAR
```

Declaration

```
PROGRAM ReadLrealListParameter
VAR
  ReadListArray: ARRAY [0..1023] OF LREAL; (* Data array for the received bytes *)

  FB_ReadLrealListParameter: MB_ReadLrealListParameter;
  bExecute_RLP: BOOL; (* Start ReadLrealListParameter *)
  diParameterNo_RLP: LREAL := FP_A_0_3027; (* A-0-3027 *)
  uiNoOfBytes_RLP: UINT := 8192; (* LREAL := 8 Bytes *)
  bDone_RLP: BOOL; (* TRUE := FB ready *)
  bActive_RLP: BOOL; (* The FB works *)
  bError_RLP: BOOL; (* Error flag *)
  enErrorID_RLP: ERROR_CODE; (* Error class *)
  stErrorIdent_RLP: ERROR_STRUCT; (* Error definition *)
  uiNoOfRecBytes_RLP: UINT; (* Number of received bytes *)
END_VAR
```

Program example in the function block diagram

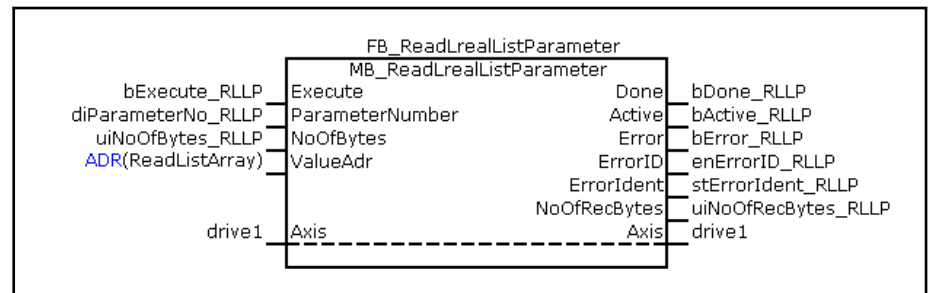


Fig. 6-118:

Program example in structured text

```

FB_ReadLrealListParameter(
  Execute:=      bExecute_RLP,          (* Start ReadLrealListParameter *)
  ParameterNumber:= diParameterNo_RLP,  (* A-0-3027 *)
  NoOfBytes:=    uiNoOfBytes_RLP,       (* LREAL:= 8 Bytes *)
  ValueAdr:=     ADR(ReadListArray),    (* Address of ReadListArray *)
  Axis:=         drive1,                (* Address const.: drive1 *)
  Done=>         bDone_RLP,              (* TRUE := FB ready *)
  Active=>       bActive_RLP,            (* The FB works *)
  Error=>        bError_RLP,             (* Error occurs *)
  ErrorId=>      enErrorID_RLP,          (* Error class *)
  ErrorIdent=>   stErrorIdent_RLP:      (* Error definition *)
  NoOfRecBytes=> uiNoOfRecBytes_RLP);    (* Number of received bytes *)

```



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_ReadLrealListParameter 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

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#), page 357 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#), page 360 - error when accessing the axis/the control.

6.10.8 MB_ReadLrealParameter

Brief description The function block returns the value of a user-specific parameter. The return value is represented as a LREAL 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
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCOpen/parameters

Tab. 6-107: Library assignment

Interface description

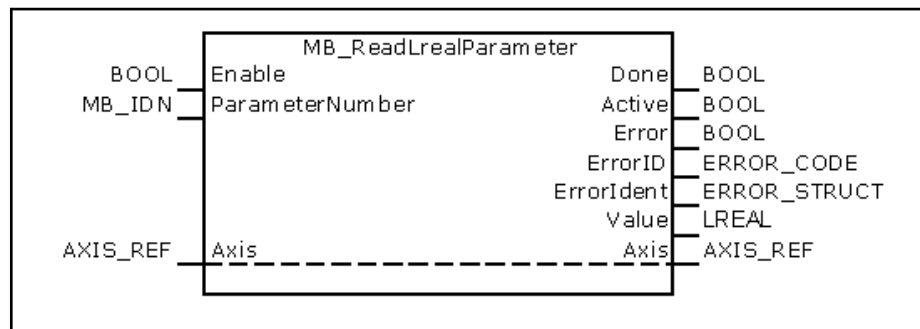


Fig. 6-119: MB_ReadLrealParameter FB

	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 (refer to Access to data of an axis, control, kinematics (axis group) or another object, page 278)
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 295)
	Value	LREAL	8 byte value represented as LREAL number
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it

Tab. 6-108: Interface of the MB_ReadLrealParameter function block

Signal-time diagram

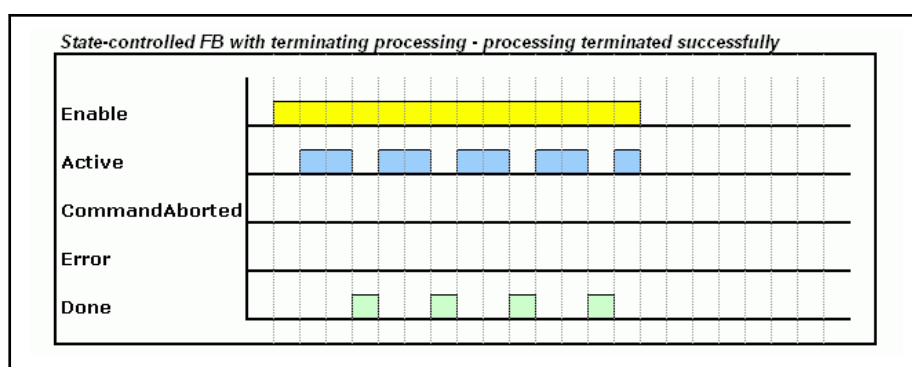


Fig. 6-120: Default signal-time diagram



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_ReadLrealParameter 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

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 357 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 360 - error when accessing the axis/the control.

6.10.9 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
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-109: Library assignment

Interface description

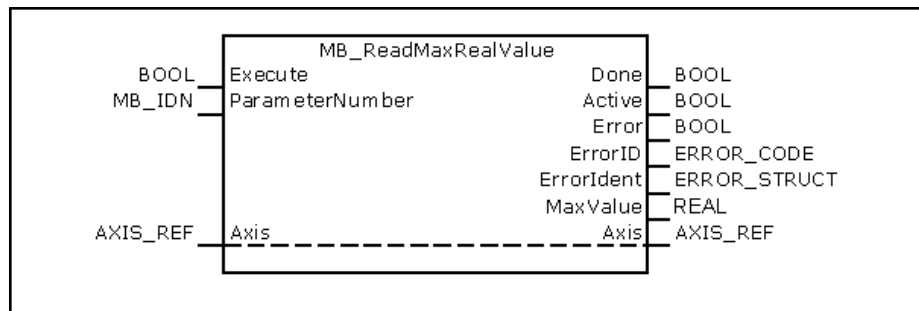


Fig. 6-121: MB_ReadMaxRealValue FB

	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 297)
	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. 6-110: Interface of the MB_ReadMaxRealValue function block

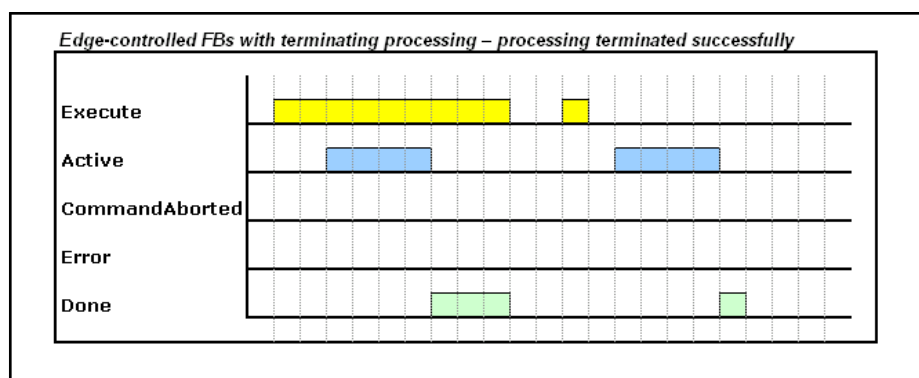
Brief description
Signal-time diagram

Fig. 6-122: 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.

6.10.10 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
IndraLogic XLC/IndraMotion MLC 14VRS	ML_PLCopen/parameters
Basic function package PLC	

Tab. 6-111: Library assignment

Interface description

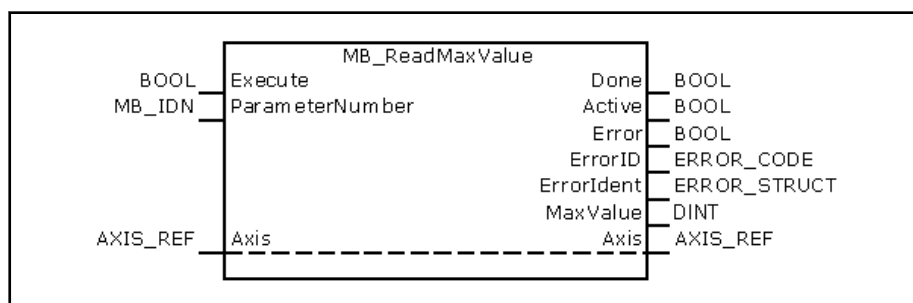


Fig. 6-123: MB_ReadMaxValue FB

	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 299)
	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. 6-112: Interface of the MB_ReadMaxValue FB

Brief description

Signal-time diagram

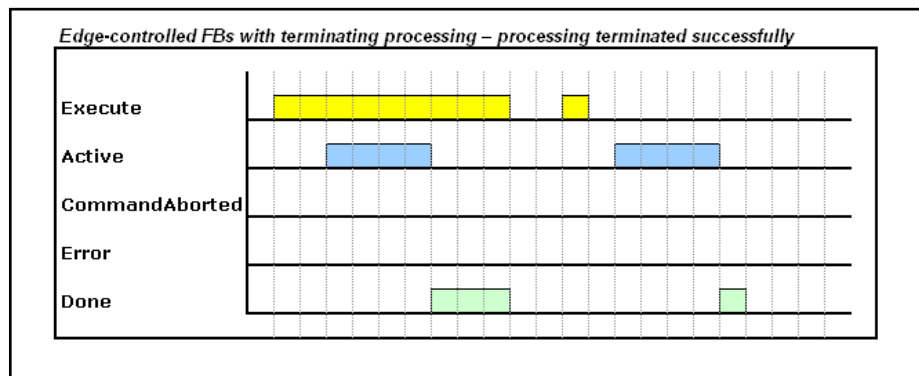


Fig. 6-124: Default signal-time diagram



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.

6.10.11 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
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-113: Library assignment

Interface description

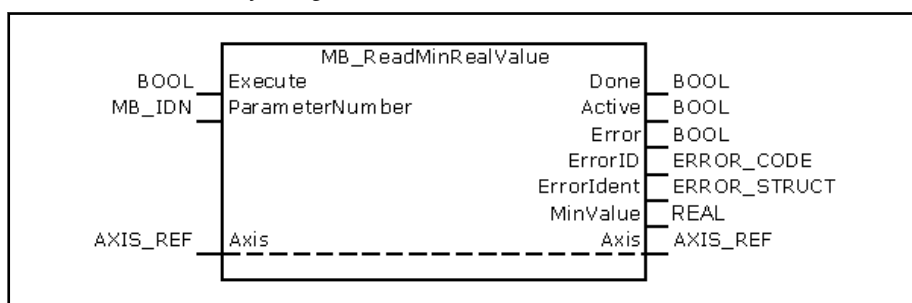


Fig. 6-125: MB_ReadMinRealValue FB

	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

	Name	Type	Comment
	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 300)
	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. 6-114: Interface of the MB_ReadMinRealValue function block

Signal-time diagram

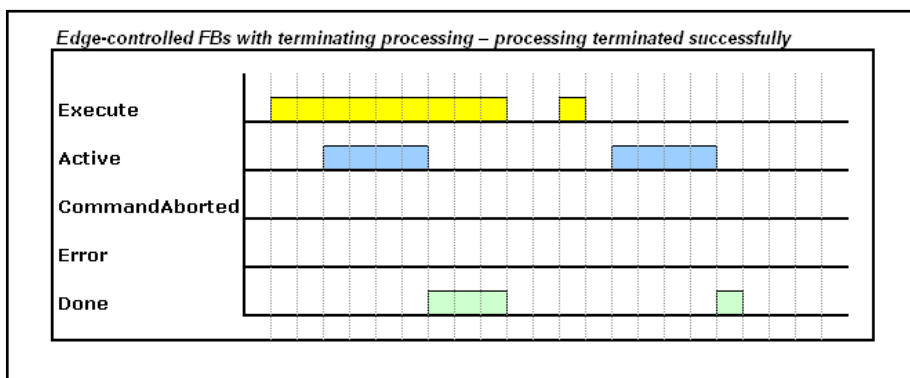


Fig. 6-126: Default signal-time diagram



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.

6.10.12 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
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCOpen/parameters

Tab. 6-115: Library assignment

Interface description

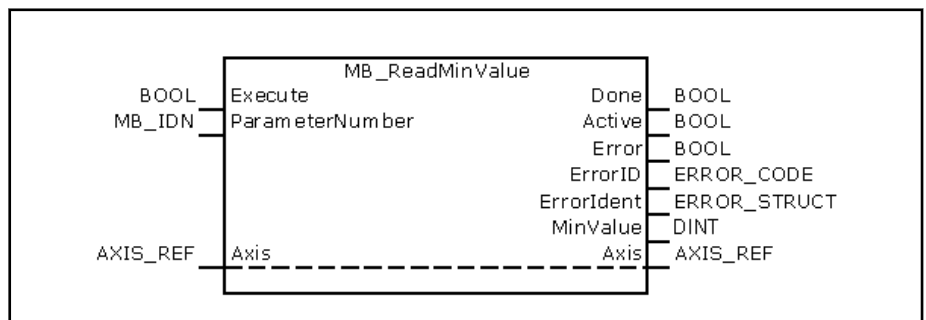


Fig. 6-127: MB_ReadMinValue FB

	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
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information (" Troubleshooting " on page 302)
	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. 6-116: Interface of the MB_ReadMinValue FB

Signal-time diagram

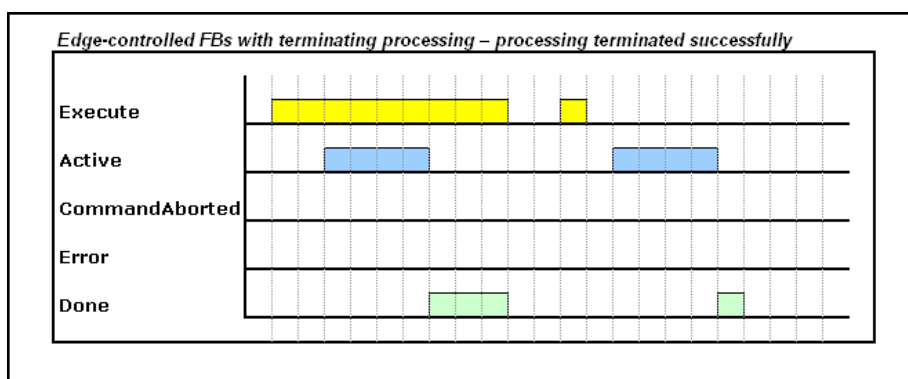


Fig. 6-128: Default signal-time diagram



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.

6.10.13 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
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-117: Library assignment

Interface description

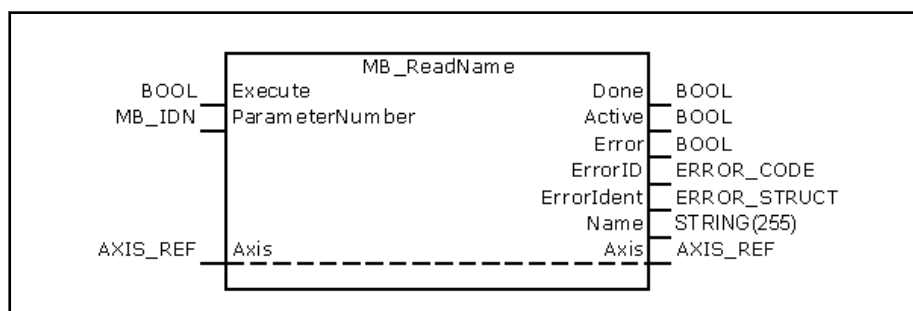


Fig. 6-129: MB_ReadName FB

	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 304)
	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. 6-118: Interface of the MB_ReadName function block

Signal-time diagram

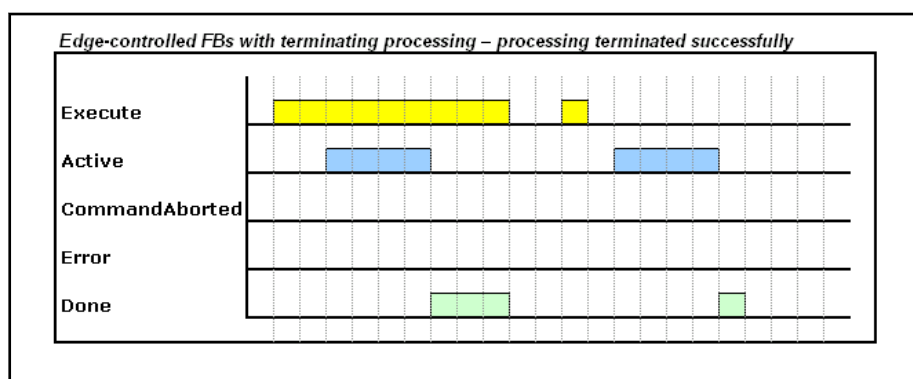


Fig. 6-130: Default signal-time diagram



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.

6.10.14 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
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-119: Library assignment

Interface description

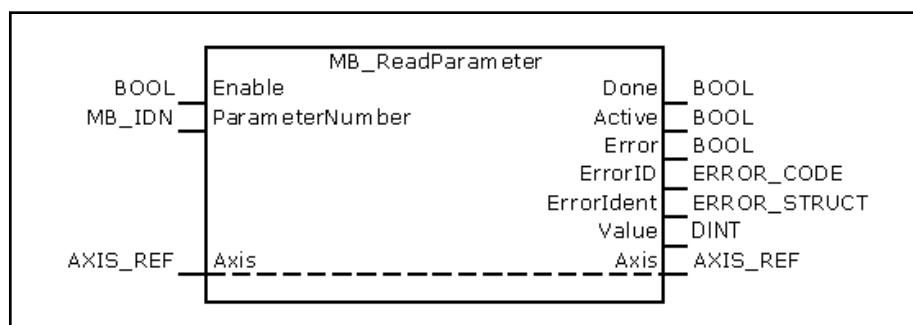


Fig. 6-131: MB_ReadParameter FB

	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 305)
	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. 6-120: Interface of the MB_ReadParameter FB

Signal-time diagram

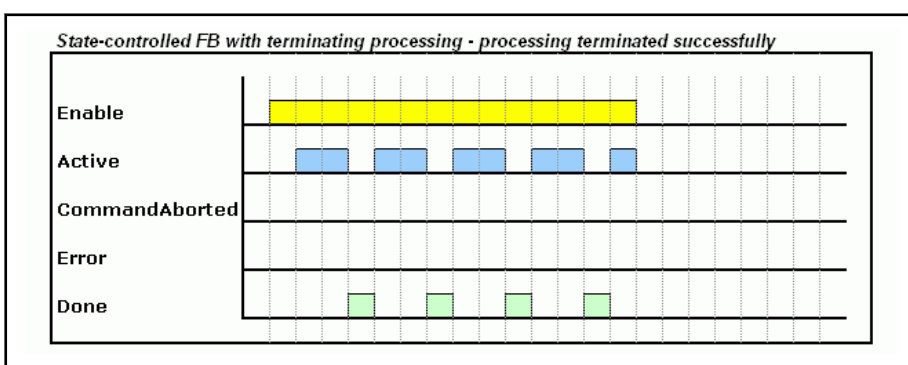


Fig. 6-132: Default signal-time diagram



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.

6.10.15 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
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCOpen/parameters

Tab. 6-121: Library assignment

Interface description

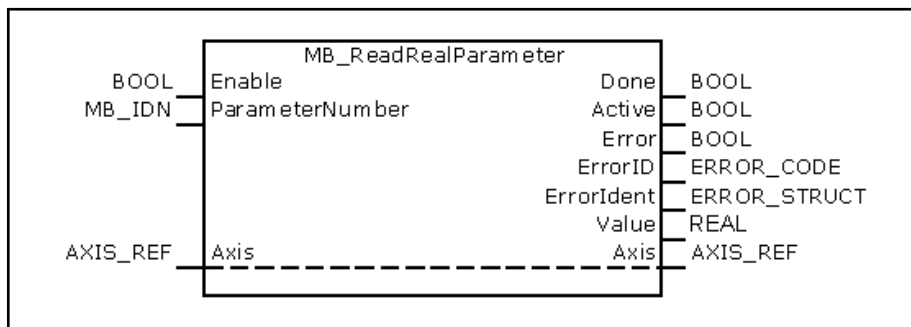


Fig. 6-133: MB_ReadRealParameter FB

	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 307)

	Name	Type	Comment
	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. 6-122: Interface of the MB_ReadRealParameter FB

Signal-time diagram

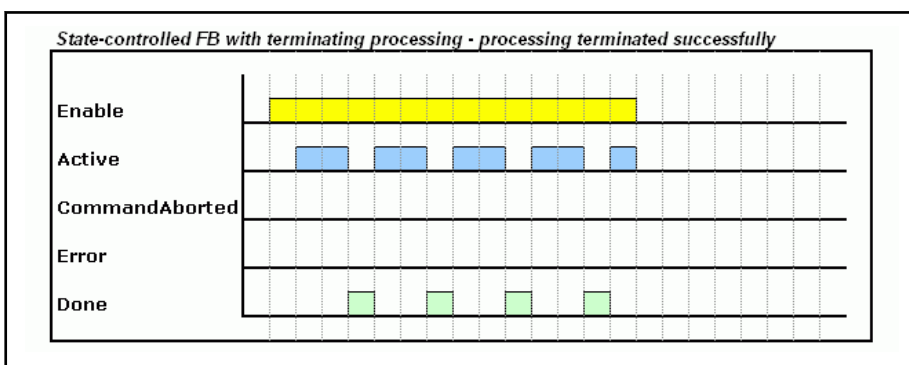


Fig. 6-134: Default signal-time diagram



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.

6.10.16 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
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-123: Library assignment

Interface description

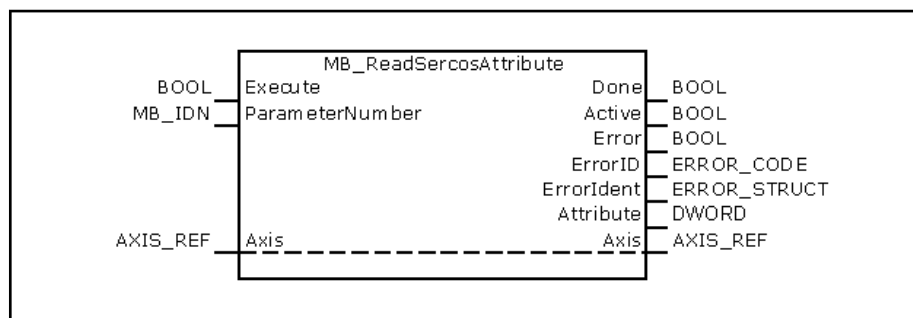


Fig. 6-135: MB_ReadSercosAttribute FB

	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 309)
	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. 6-124: 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

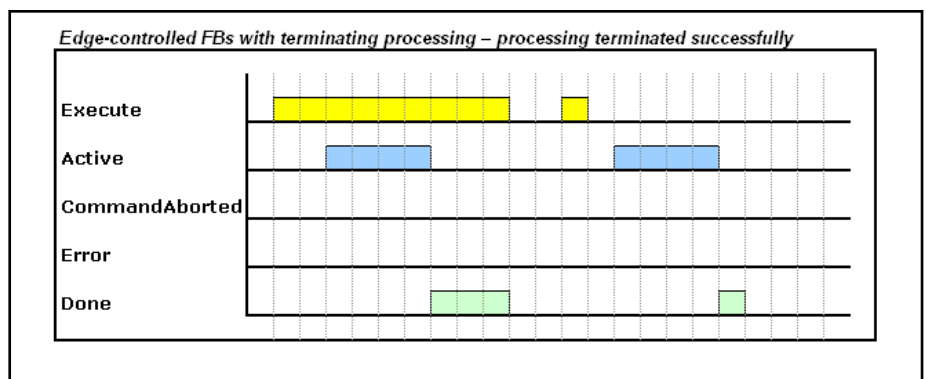


Fig. 6-136: Default signal-time diagram



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.

6.10.17 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
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-125: Library assignment

Interface description

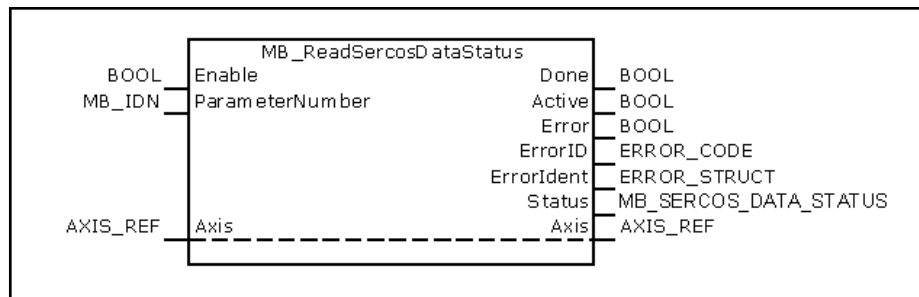


Fig. 6-137: FB MB_ReadSercosDataStatus

	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 311)
	Status	MB_SERCOS_DATA_STATUS	Status of the selected parameter [External link could not be resolved.]
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-126: Interface of the MB_ReadSercosDataStatus function block

Brief description Using this function block and a [MB_WriteParameter](#), page 322,, 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

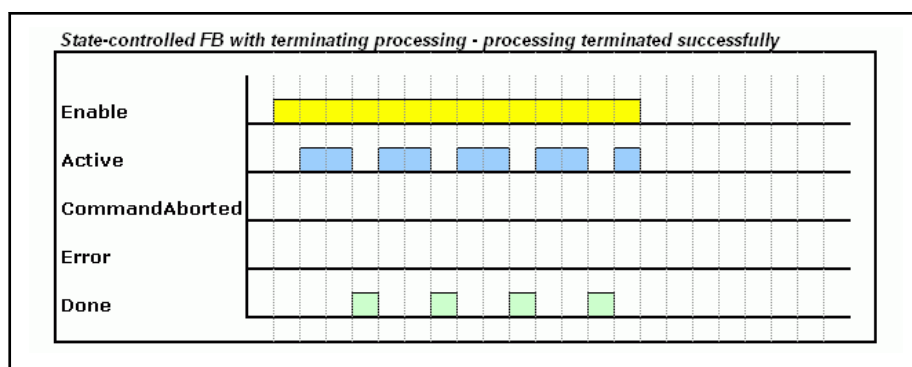


Fig. 6-138: Default signal-time diagram



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.

6.10.18 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
IndraLogic XLC/IndraMotion MLC 14VRS	ML_PLCopen/parameters
Basic function package PLC	

Tab. 6-127: Library assignment

Interface description

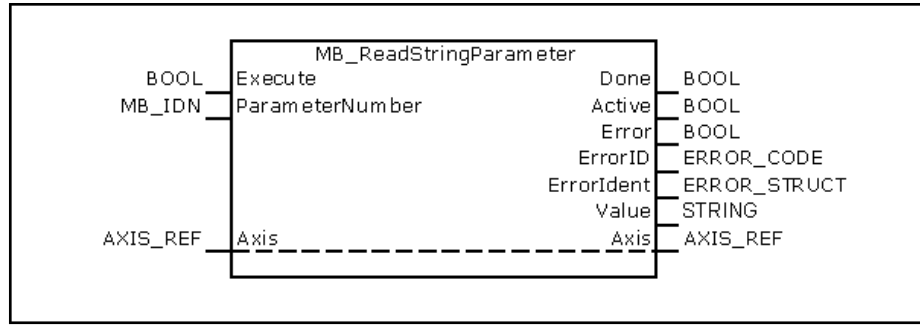


Fig. 6-139: MB_ReadStringParameter FB

	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 313)
	Value	STRING	Character string with up to 255 characters
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-128: Interface of the MB_ReadStringParameter FB

Signal-time diagram

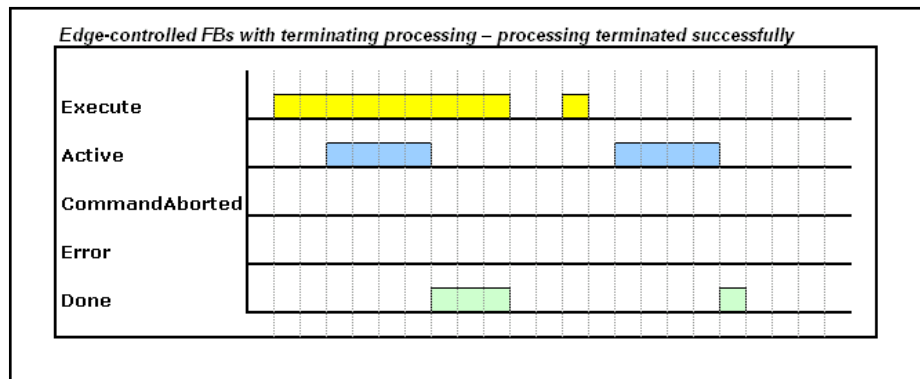


Fig. 6-140: Default signal-time diagram



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.

6.10.19 MB_ReadStringParameterExt

Brief description

The function block returns the value of a user-specific parameter.

The read character string is saved with the entered address "ValueAdr". The length of the returned character string is limited by the "MaxLen" input to avoid that the character string exceeds the valid memory.

The "StringLen" output shows the number of the actual read characters.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-129: Library assignment

Interface description

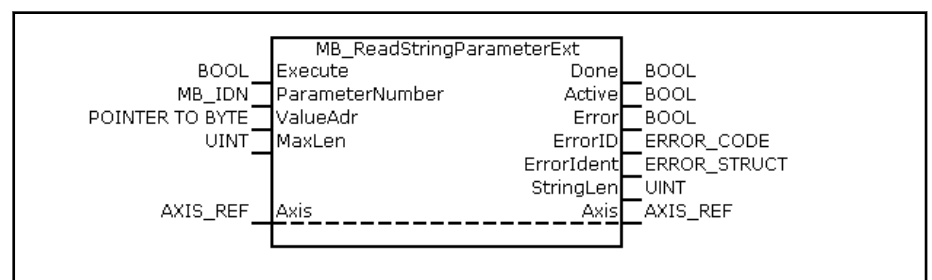


Fig. 6-141: MB_ReadStringParameter FB

	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 (refer to Access to data of an axis, control, kinematics (axis group) or another object, page 278)

	Name	Type	Comment
	ValueAdr	POINTER TO BYTE	Address the character string is saved to
	MaxLen	UINT	Max. length of the returned character string
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 314)
	StringLen	UINT	Number of characters read
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-130: Interface of the MB_ReadStringParameter FB

Signal-time diagram

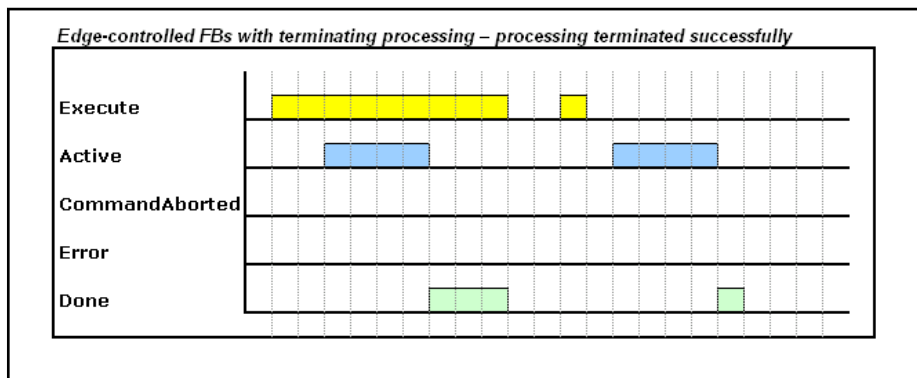


Fig. 6-142: Default signal-time diagram



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#0000	Wrong data type
INPUT_INVALID_ERROR	MLC_TABLE	16#F0220523	16#0000	ValueAdr equal to 0

For more errors, refer to

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 357 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 360 - error when accessing the axis/the control.

6.10.20 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
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-131: Library assignment

Interface description

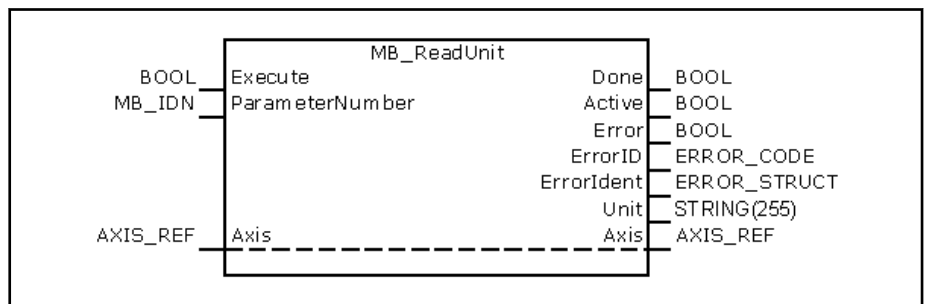


Fig. 6-143: MB_ReadUnit FB

	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. 6-132: Interface of the MB_ReadName function block

Signal-time diagram

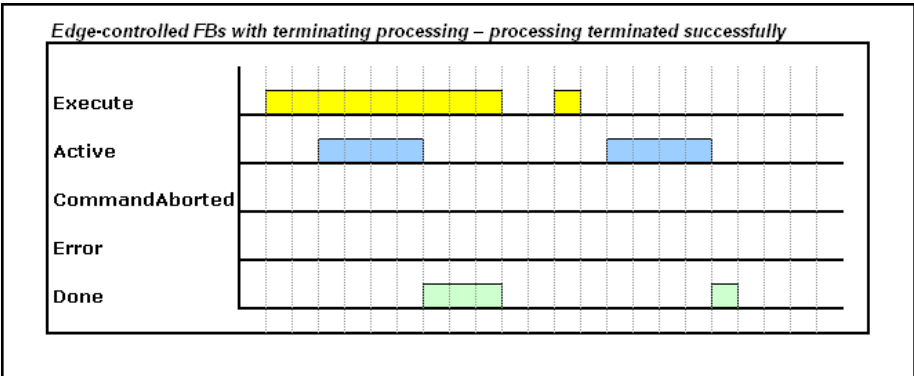


Fig. 6-144: Default signal-time diagram



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.

6.10.21 MB_WriteLintParameter

Brief description The function block writes a value in a parameter specified by the user.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCOpen/parameters

Tab. 6-133: Library assignment

Interface description

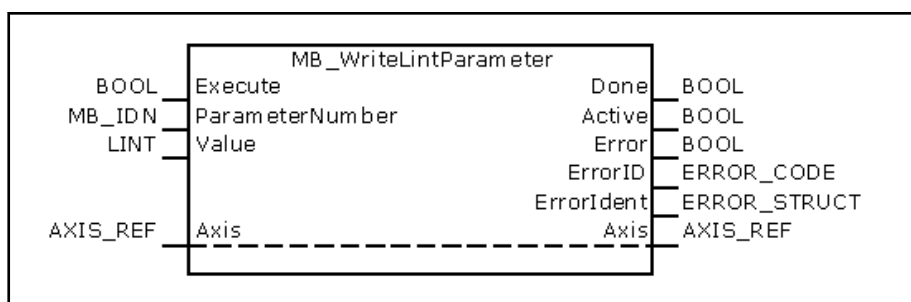


Fig. 6-145: MB_WriteLintParameter

	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 (refer to Access to data of an axis, control, kinematics (axis group) or another object, page 278)
	Wert	LINT	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 318)
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it

Tab. 6-134: Interface of the MB_WriteLintParameter function block

Signal-time diagram

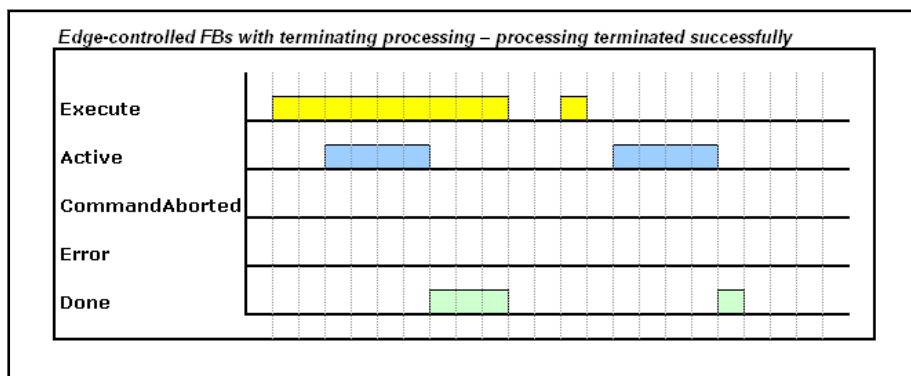


Fig. 6-146: Default signal-time diagram



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_WriteLintParameter 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

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 357 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 360 - error when accessing the axis/the control.

6.10.22 MB_WriteListParameter

Brief description

The function block writes memory content of a defined length (NoOfByte) provided from the initial address (ValueAdr) onwards to a user-specified parameter.



The specified number of bytes must exactly match the data volume to be transferred!

If this number is too high, data waste is transferred. If it is too small, data is missing in the target parameter!



The data type of the elements of the parameter to be written and the data type of the elements of the ARRAY have to match.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCOpen/parameters

Tab. 6-135: Library assignment

Interface description

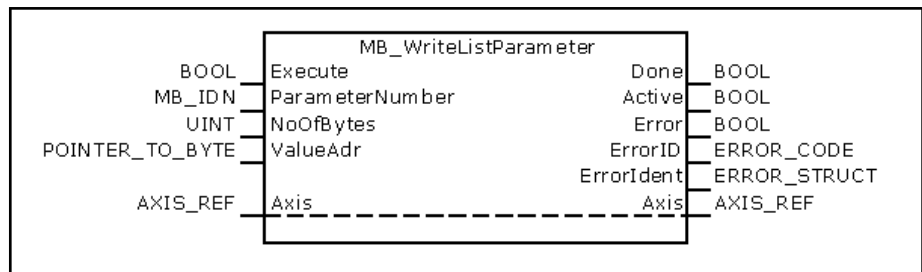


Fig. 6-147: MB_WriteListParameter FB

	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 (refer to Access to data of an axis, control, kinematics (axis group) or another object, page 278)
	NoOfBytes	UINT	Number of bytes to be written
	ValueAdr	POINTER TO BYTE	Pointer to the initial address of the memory in which the data to be written is saved.
VAR_OUTPUT	Done	BOOL	The value was successfully written 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
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it

Tab. 6-136: Interface of the MB_WriteListParameter function block

Signal-time diagram

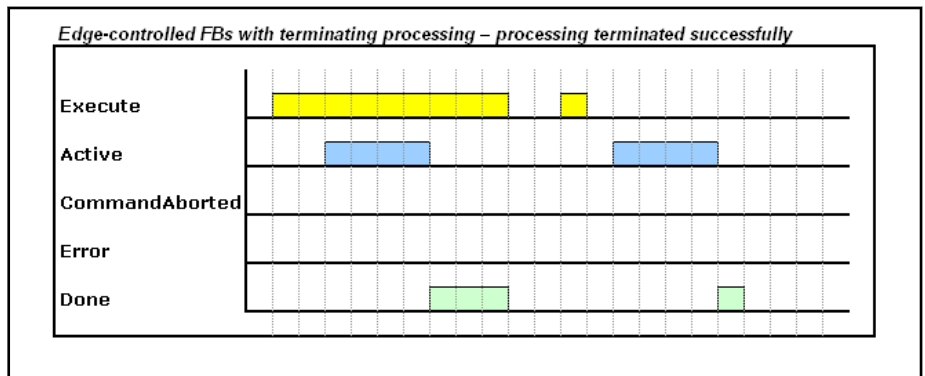


Fig. 6-148: Default signal-time diagram

Program example: MB_WriteList-Parameter

This program example shows the writing of the list parameter (C-0-2001, ELS, point table1) of the MLC with 1,024 values (DINT) from the WriteListArray.

Declaration of "drive1" as constant in MlcVarGlobal

```
(* Automatically generated code by MLC. *)
(* Please don't edit, the code will be overwritten. For usercode use UserVarGlobal! *)
{attribute 'linkalways'}
VAR_GLOBAL CONSTANT

(*Reale Achse "drive1" mit Achsnummer 1*)
(*Real axis "drive1" with axis number 1*)

drive1: AXIS_REF := (CntrlNo:=LOCAL_CNTRL,AxisNo:=AXIS_1);
(* ... *)
END_VAR
```

Declaration

```
PROGRAM WriteListParameter
VAR
```

ML_PLCOpen.library

```

WriteListArray:      ARRAY [0..1023] OF DINT; (* Data array for the write bytes *)

FB WriteListParameter: MB WriteListParameter;
bExecute_WLP:        BOOL;                  (* Start WriteListParameter *)
diParameterNo_WLP:   MB_IDN:= FP_C_0_2001;  (* MLC Parameter ControlCAM1 *)
uiNoOfBytes_WLP:     UINT := 4096;          (* DINT := 4 Bytes *)
bDone_WLP:           BOOL;                  (* TRUE := FB ready *)
bActive_WLP:         BOOL;                  (* The FB works *)
bError_WLP:          BOOL;                  (* Error flag *)
enErrorID_WLP:       ERROR_CODE;           (* Error class *)
stErrorIdent_WLP:    ERROR_STRUCT;         (* Error definition *)

END_VAR

```

Program example in the function block diagram

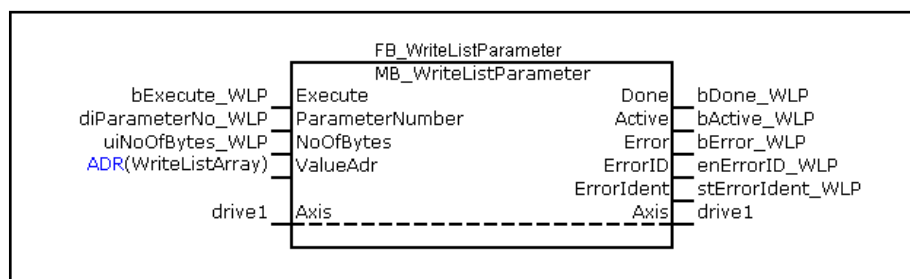


Fig. 6-149: FBD program example: WriteListParameter

Program example in structured text

```

FB_WriteListParameter(
  Execute:= bExecute_WLP,          (* Start WriteListParameter *)
  ParameterNumber:= diParameterNo_WLP, (* MLC Parameter ControlCAM1 *)
  NoOfBytes:= uiNoOfBytes_WLP,      (* DINT := 4 Bytes *)
  ValueAdr:= ADR(WriteListArray),  (* Address of WriteListArray *)
  Axis:= drive1,                   (* Address const.: drive1 *)
  Done=> bDone_WLP,                 (* TRUE := FB ready *)
  Active=> bActive_WLP,             (* The FB works *)
  Error=> bError_WLP,               (* Error occurs *)
  ErrorId=> enErrorID_WLP,          (* Error class *)
  ErrorIdent=> stErrorIdent_WLP);  (* Error definition *)

```



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_WriteListParameter 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

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 357 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 360 - error when accessing the axis/the control.

6.10.23 MB_WriteLrealParameter

Brief description This function block writes a value to a user-specific parameter.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-137: Library assignment

Interface description

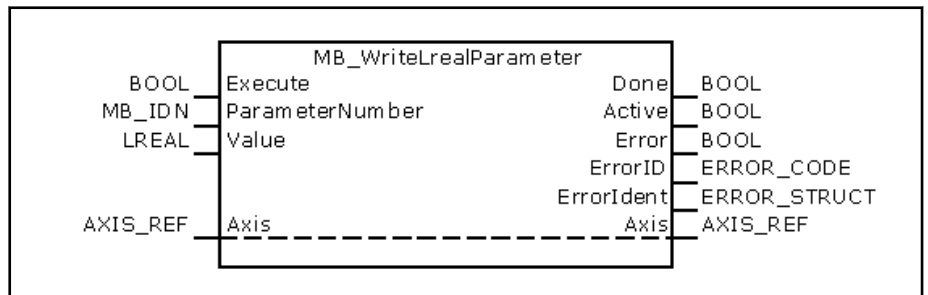


Fig. 6-150: FB MB_WriteLrealParameter

	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 (refer to Access to data of an axis, control, kinematics (axis group) or another object , page 278)
	Value	LREAL	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 322)
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it

Tab. 6-138: Interface of the MB_WriteRealParameter FB

Signal-time diagram

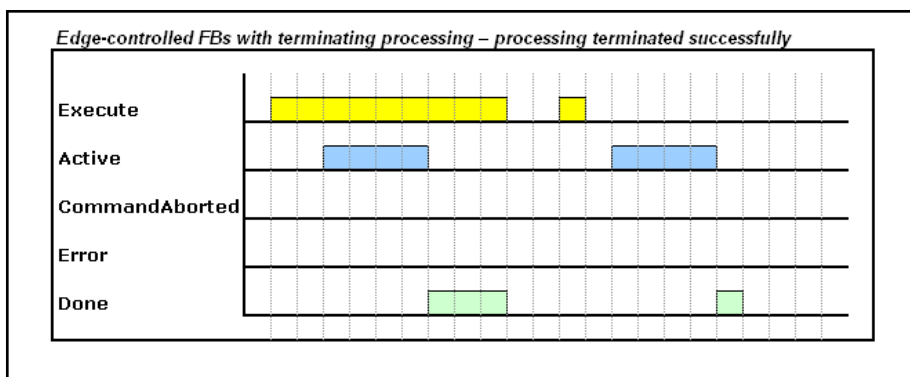


Fig. 6-151: Default signal-time diagram



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_WriteLrealParameter 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

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 357 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 360 - error when accessing the axis/the control.

6.10.24 MB_WriteParameter

Brief description

The function block writes a value in a parameter specified by the user.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCOpen/parameters

Tab. 6-139: Library assignment

Interface description

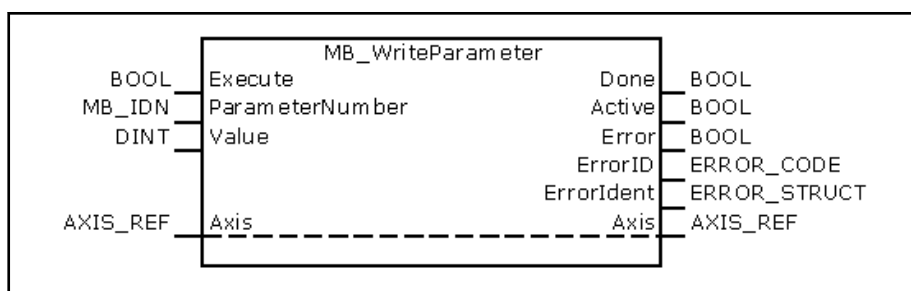


Fig. 6-152: MB_WriteParameter FB

	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 324)
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it

Tab. 6-140: Interface of the MB_WriteParameter FB

Signal-time diagram

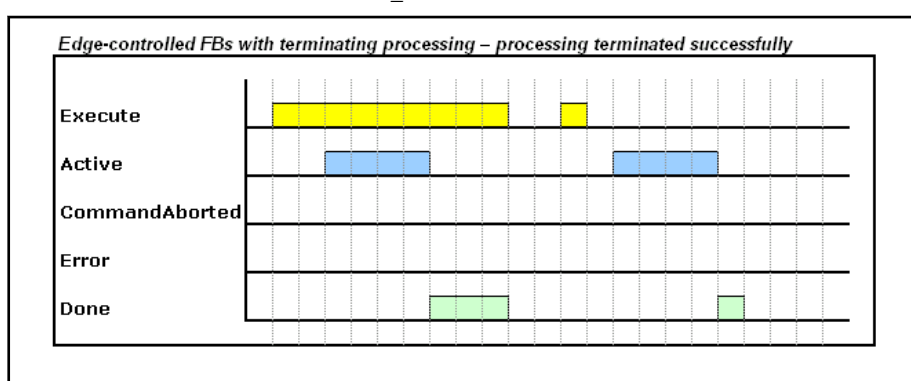


Fig. 6-153: Default signal-time diagram



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.

6.10.25 MB_WriteRealParameter

Brief description This function block writes a value to a user-specific parameter.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCOpen/parameters

Tab. 6-141: Library assignment

Interface description

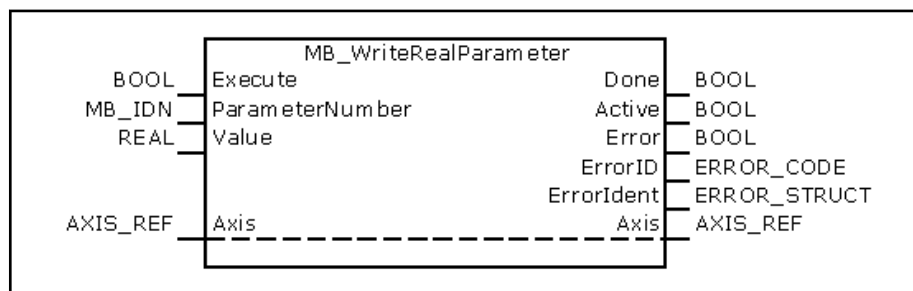


Fig. 6-154: MB_WriteRealParameter FB

	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

	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 325)
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it

Tab. 6-142: Interface of the MB_WriteRealParameter FB

Signal-time diagram

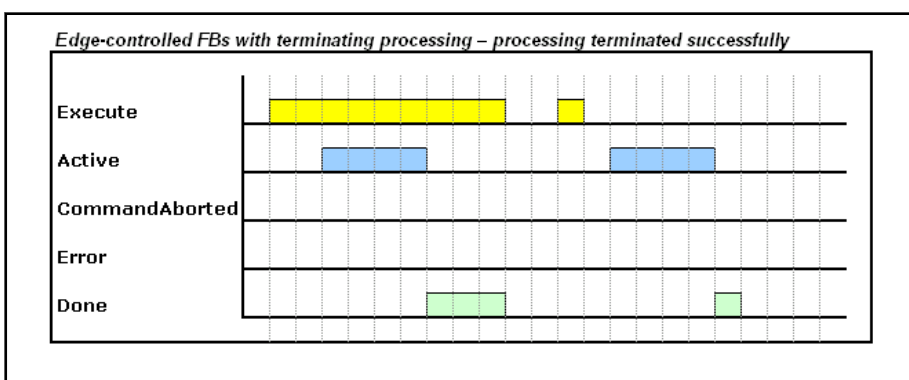


Fig. 6-155: Default signal-time diagram



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.

6.10.26 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
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-143: Library assignment

Interface description

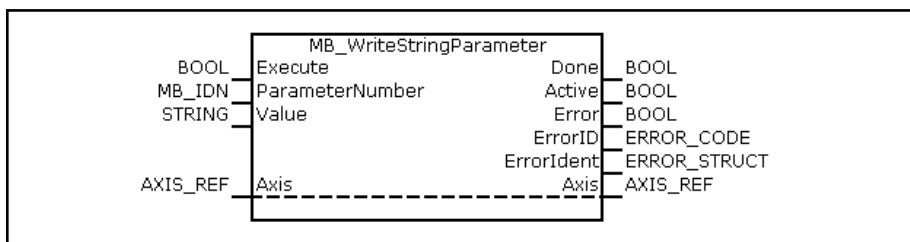


Fig. 6-156: MB_WriteStringParameter FB

	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
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 327)
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it

Tab. 6-144: Interface of the MB_WriteStringParameter FB

Signal-time diagram

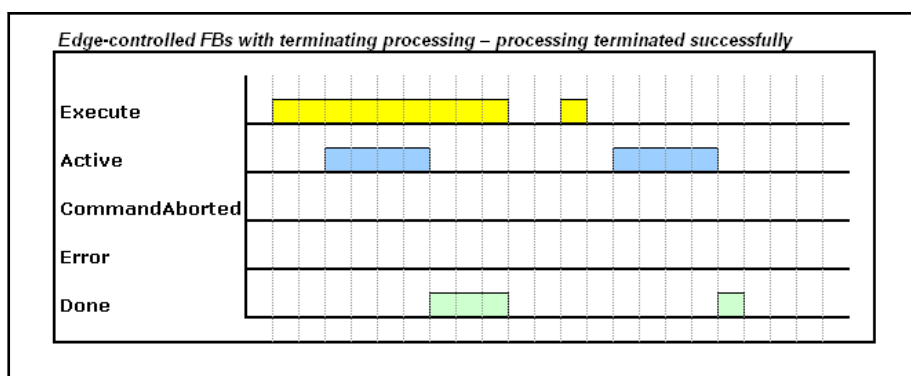


Fig. 6-157: Default signal-time diagram



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.

6.10.27 MB_WriteStringParameterExt

Brief description

This function block writes a character string to a user-specific parameter. The character string to be written is located under the "ValueAdr" address. It is accessed via pointer.

Assignment: Target system/library

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-145: Library assignment

Interface description

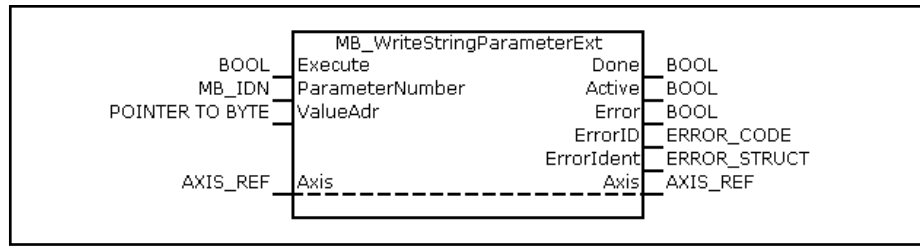


Fig. 6-158: MB_WriteStringParameter FB

	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 (refer to Access to data of an axis, control, kinematics (axis group) or another object, page 278)
	ValueAdr	POINTER TO BYTE	Address used to store the character string 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 329)
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it

Tab. 6-146: Interface of the MB_WriteStringParameter FB

Signal-time diagram

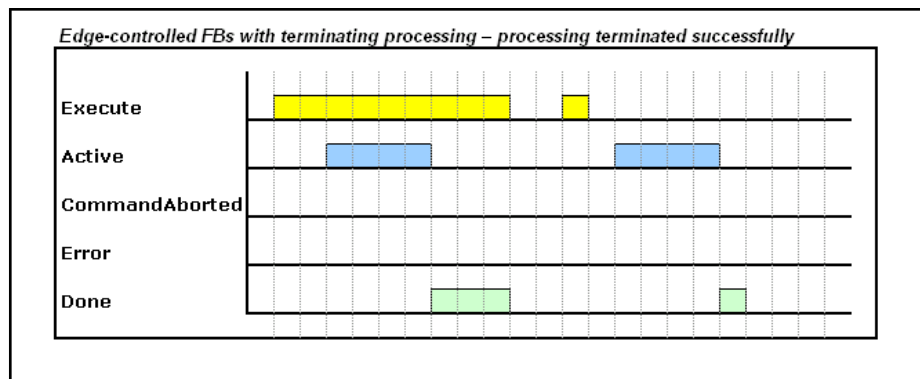


Fig. 6-159: Default signal-time diagram



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#00	Wrong data type
INPUT_INVALID_ERROR	MLC_TABLE	16#F0220523	16#00	ValueAdr equal to 0

For more errors, refer to

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 357 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 360 - error when accessing the axis/the control.

6.10.28 MC_ReadActualPosition

Brief description

The function block reads the current position of the drive.



A modification of the axis at the input of the function block requires another 0/1 transition at "Enable".

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-147: Library assignment

Interface description

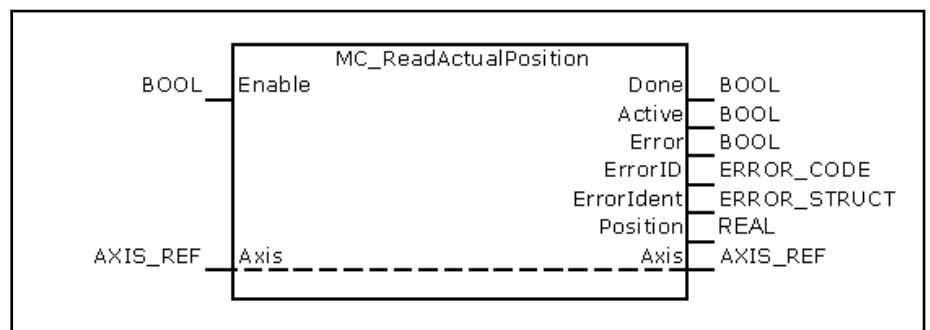


Fig. 6-160: MC_ReadActualPosition FB

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Continuous reading of the position value as long as "Enable" is TRUE
VAR_OUTPUT	Done	BOOL	A valid value is available

	Name	Type	Comment
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information (" Troubleshooting " on page 330)
	Position	REAL	Current absolute position of the drive ¹¹⁾
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-148: Interface of the MC_ReadActualPosition FB

Signal-time diagram

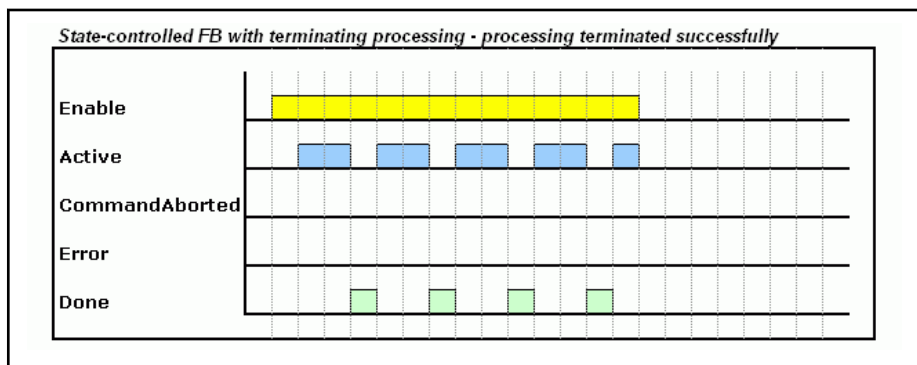


Fig. 6-161: Default signal-time diagram

Parameters used
(Sercos-based axes)

The function block reads the current position of the drive, (A-0-0100, Actual position value).

Troubleshooting

The MC_ReadActualPosition function block generates the following error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- [MLC_TABLE, page 360](#) - Error when accessing the axis/control.

6.10.29 MC_ReadAxisError

Brief description

The function block queries the error state of the addressed axis. One instance of the function block is to be used for each axis.



A modification of the axis at the input of the function block requires another 0/1 transition at "Enable".

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-149: Library assignment

¹¹⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

Interface description

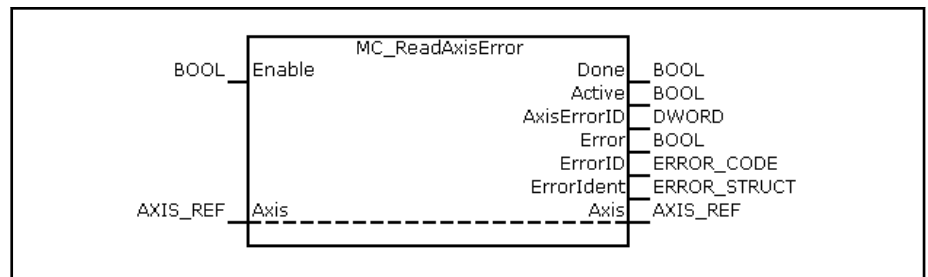


Fig. 6-162: MC_ReadAxisError FB

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Continuous reading of the error information as long as "Enable" is TRUE
VAR_OUTPUT	Done	BOOL	Value available
	Active	BOOL	Data processing runs when preprocessing is complete
	AxisErrorID	DWORD	It indicates the error of the drive of the axis
	Error	BOOL	Indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Short note to the error cause of the function block call, not of the axis error!
	ErrorIdent	ERROR_STRUCT	Detailed information on the error of the function block call, not the axis error! "Troubleshooting" on page 331
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-150: Interface of the MC_ReadActualPosition FB

Signal-time diagram

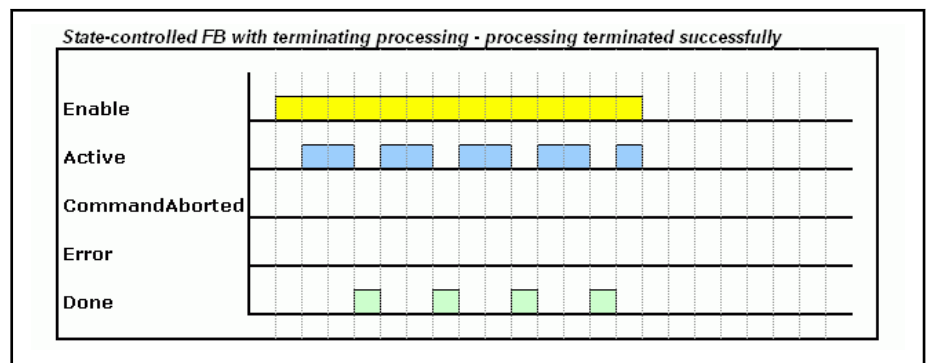


Fig. 6-163: Default signal-time diagram

Parameters used
(Sercos-based axes)

This function block retrieves the error status (A-0-0020, Axis diagnostic message) of the axis addressed.

Troubleshooting

The MC_ReadAxisError function block generates the following error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- [MLC_TABLE, page 360](#) - Error when accessing the axis/control.

6.10.30 MC_ReadStatus

Brief description This function block reads the current status of the axis and outputs this in a decoded way.



A modification of the axis at the input of the function block requires another 0/1 transition at "Enable".



Due to the extension of the function block, it was modified in a way which is incompatible with the previous version!

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS Basic function package PLC	ML_PLCopen/parameters

Tab. 6-151: Library assignment

Interface description

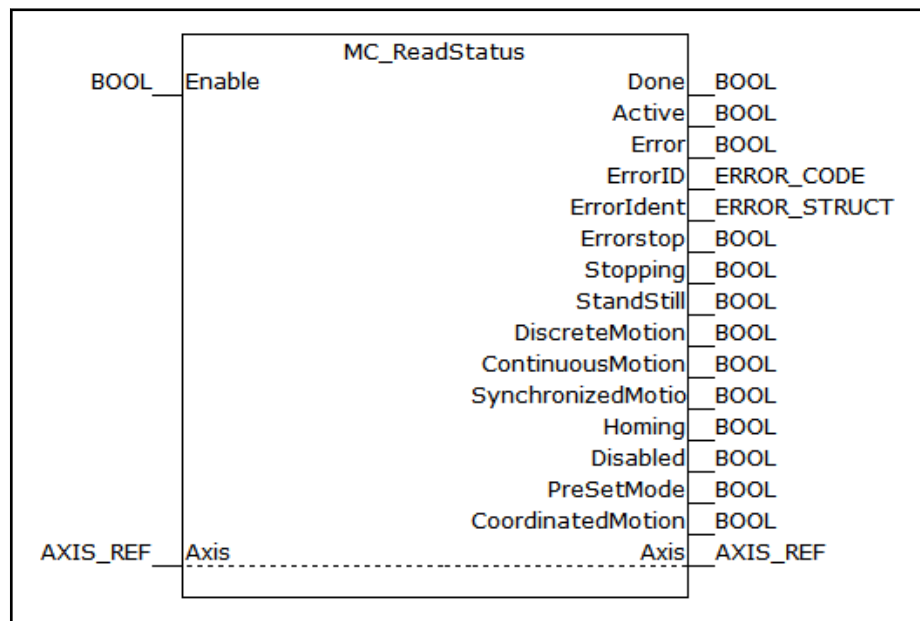


Fig. 6-164: MC_ReadStatus FB

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	Continuous reading of the axis state as long as "Enable" is TRUE
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information (" Troubleshooting " on page 333)
	Errorstop	BOOL	Drive/axis in "Errorstop" state
	Stopping	BOOL	Drive/axis in "Stopping" state
	StandStill	BOOL	Drive/axis in "Standstill" state
	DiscreteMotion	BOOL	Drive/axis in "DiscreteMotion" state

	Name	Type	Comment
	Continuous-Motion	BOOL	Drive/axis in "ContinuousMotion" state
	Synchron-izedMotion	BOOL	Drive/axis in "SynchronizedMotion" state
	Homing	BOOL	Drive in "Homing" state
	Disabled	BOOL	Drive/axis in "Disabled" state
	PreSetMode	BOOL	Drive in "PreSetMode" state
	Coordinated-Motion	BOOL	Drive/axis in "CoordinatedMotion" state
VAR_IN_OUT CONSTANT	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-152: Interface of the MC_ReadStatus FB

Signal-time diagram

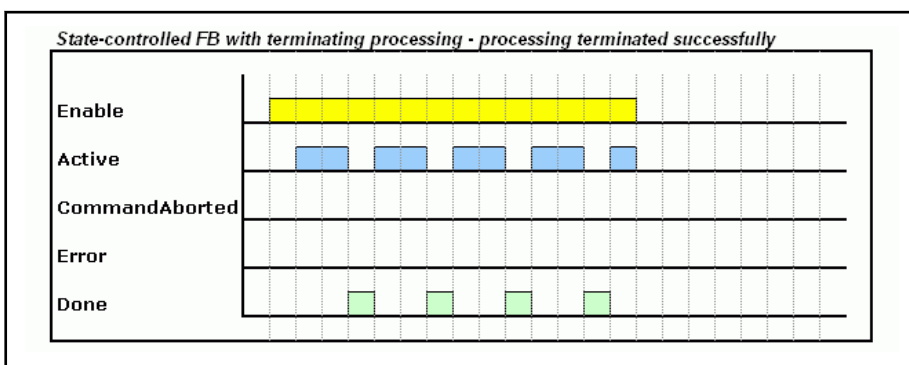


Fig. 6-165: Default signal-time diagram

Parameters used This function block reads the actual state of the axis. (A-0-0021, Axis status; A-0-0022, Extended axis status) and decodes and outputs it. (Also see [State diagram of a real axis](#), page 176.)

Troubleshooting The MC_ReadStatus function block generates the following error messages:

ErrorID	Table	Add1	Add2	Description
ACCESS_ERROR	MLC_TABLE	16#F0220501	16#00000000	Wrong data type

For more errors, refer to

- [MLC_TABLE](#), page 360 - Error when accessing the axis/control.

6.11 Function blocks to use the touch probe function

6.11.1 Data and function blocks to use the touch probe function

With a binary sensor (touch probe), the touch probe function allows a highly accurate position determination in the running production process.

For example, it is possible to register the printed mark on a trajectory to be processed with the touch probe (positive edge/negative edge or both).

The objective for the start is to bring the axis to which the touch probe is assigned into a defined position to the printed mark and to keep it there during the running manufacturing process.

Due to the technology, the printed mark vary around an average value with regard to the desired axis position.

The measurement signal (positive edge/negative edge or both) is thus expected to be around the desired value. The window for the desired edge can be specified as absolute, start position/end position of the expectation window, or as relative, center point plus positive/negative deviation.

The failure of a printed mark or a deviation of the edge in a position which is beyond the window is regarded as a imprint error which has to be registered and reacted to.

During the manufacturing process, it is necessary that the position and width of the expectation window can be modified.

The touch probe signal is delayed from the place of registration (touch probe) to the place of processing (control). This delay might be different for the positive and the negative edge and can be compensated for.

As an alternative to the continuous operation, a single shot measurement should be possible. Furthermore, a time measurement between two events is desirable.

Based on the IndraDrive touch probe function, hardware supported touch probe function blocks have been developed.



The function blocks to use the touch probe function require the **function package MOT** of the control

Global variable lists and data types

- [ProbeSelect, page 175](#), defines the file which contains the data of the touch probes 1..100.
- [PROBECONFIG, page 175](#), data types, data of touch probes 1..100.
- more data types for using the touch probe function are specified in the RIL_CommonTypes library.

Function blocks

- [ML_InitTouchProbe, page 336](#), initialized and configures the touch probe function.
- [ML_WriteExpectWindow, page 349](#) configures the expectation windows without a complete initialization run with the ML_InitTouchProbe being necessary.
- [ML_TouchProbe, page 344](#), used to activate the unique time position and time measurement and provides an automated time difference measurement.
- [ML_TouchProbeContinuous, page 347](#), used to activate the continuous time position and time measurement and provides an automated time difference measurement.
- [ML_AbortTrigger, page 334](#), completes an individual as well as a continuous measurement.

6.11.2 ML_AbortTrigger

Brief description The ML_AbortTrigger FB completed

- a single shot measurement is triggered by [ML_TouchProbe, page 344](#) or

- a continuous measurement is triggered by [ML_TouchProbeContinuous](#), page 347.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopn/TouchProbe

Tab. 6-153: Library assignment

Interface description

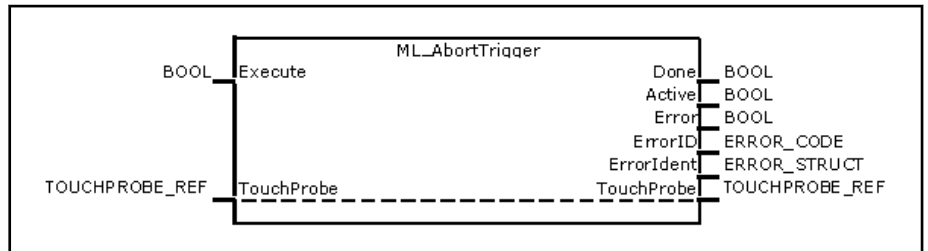


Fig. 6-166: Function block ML_AbortTrigger

	Name	Type	Comment
VAR_INPUT	Execute	BOOL	Starts the interruption
VAR_OUTPUT	Done	BOOL	The measurement was completed successfully
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information (" Troubleshooting: " on page 335)
VAR_IN_OUT CONSTANT	TouchProbe	TOUCHPROBE_REF	It contains information on the actual touch probe

Tab. 6-154: Interface of the ML_AbortTrigger FB

Signal-time diagram

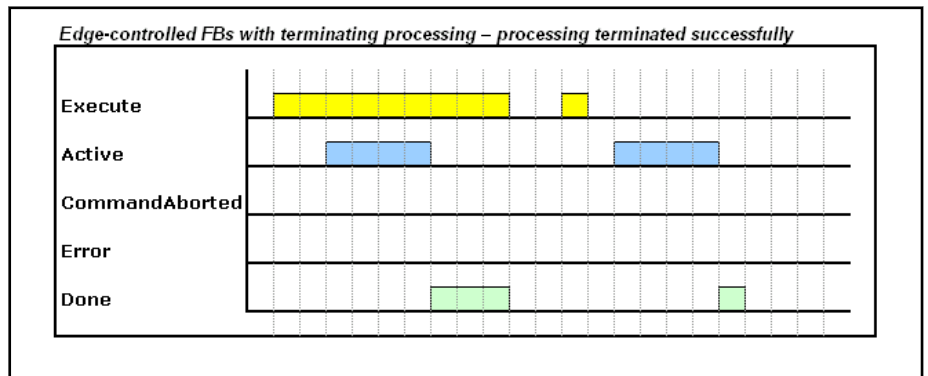


Fig. 6-167: Signal-time diagram: ML_AbortTrigger

Troubleshooting: The function block ML_AbortTrigger generates the following error messages:

ErrorID	Table	Add1	Add2	Description
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220521	16#00000000	TOUCHPROBE_REF is not valid or wrong configuration
ACCESS_ERROR	MLC_TABLE	16#F02D700C	16#00000000	sercos: Operation data is write-protected (other settings) Parameters M-0-0006, M-0-0024, M-0-0405 write-protected Parameters C-0-0400, C-0-0503: Different cycle times

For more errors, refer to:

- [MLC_TABLE, page 360](#) - error when accessing the touch probe/control.

6.11.3 ML_InitTouchProbe

Brief description The function block ML_InitTouchProbe initializes and configures the touch probe function. Its processing is a precondition for the start of the touch probe function via [ML_TouchProbe, page 344](#), or [ML_TouchProbeContinuous, page 347](#).

After its processing, the expectation window can be adapted online via [ML_WriteExpectWindow, page 349](#), .



The touch probe function can be enabled in all Sercos phases using ML_InitTouchProbe.

However, the function blocks ML_TouchProbe and ML_TouchProbeContinuous can only be executed in Sercos phase CP4.

Cause: It can only be written to the parameter "M-0-0024, Command: Activation" in P4. Executing the function block in a different Sercos phase generated the error "F02D700C, sercos: Operation data is write-protected (other settings)".

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCOpen/TouchProbe

Tab. 6-155: Library assignment

Interface description

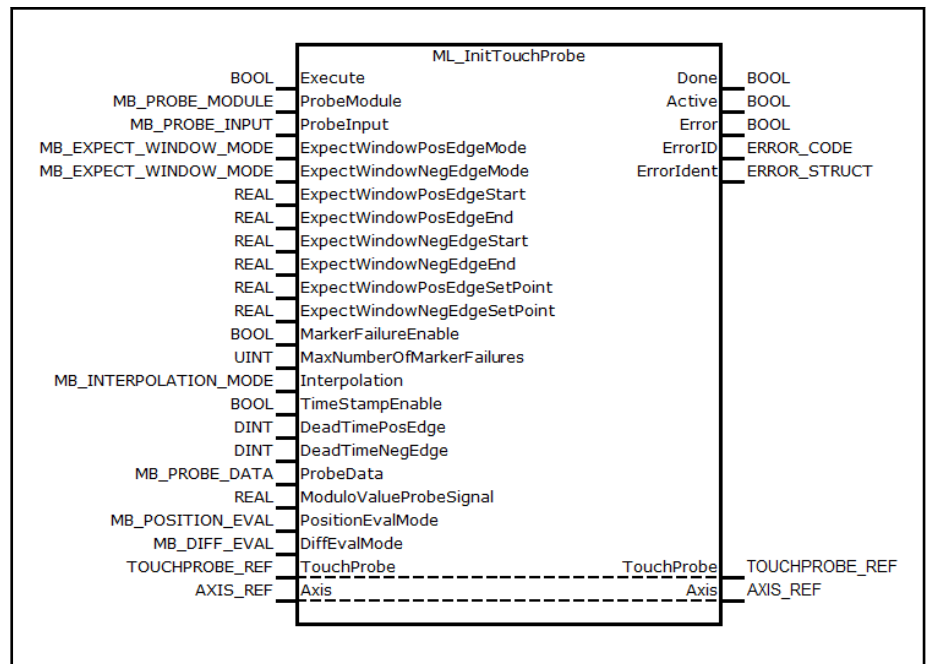


Fig. 6-168: ML_InitTouchProbe function block

	Name	Type	Comment ¹²⁾
VAR_INPUT	Execute	BOOL	Starts the initialization at the rising edge
	ProbeModule	MB_PROBE_MODULE	Selection input modules OnBoard module, function module 1...4 M-0-0005, Input selection
	ProbeInput	MB_PROBE_INPUT	Selection of the input for the trigger signal for the selected module M-0-0005, Input selection
	ExpectWindow-PosEdgeMode	MB_EXPECT_WINDOW_MODE	Activation and configuration of the expectation window of the positive edge M-0-0007, Control word
	ExpectWindow-NegEdgeMode	MB_EXPECT_WINDOW_MODE	Activation and configuration of the expectation window of the negative edge M-0-0007, Control word
	ExpectWindow-PosEdgeStart	REAL	Start position expectation window, positive edge M-0-0007, Control word M-0-0202, Starting position of the expectation window, positive edge
	ExpectWindow-PosEdgeEnd	REAL	End position expectation window, positive edge M-0-0007, Control word M-0-0203, End position of the expectation window, positive edge

¹²⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

	Name	Type	Comment ¹²⁾
	ExpectWindow-NegEdgeStart	REAL	Start position expectation window, negative edge M-0-0007, Control word M-0-0204, Starting position of the expectation window, negative edge
	ExpectWindow-NegEdgeEnd	REAL	End position expectation window, negative edge M-0-0007, Control word M-0-0205, End position of the expectation window, negative edge
	ExpectWindow-PosEdgeSet-Point	REAL	Set point of the expectation window, positive edge M-0-0007, Control word M-0-0202, Starting position of the expectation window, positive edge M-0-0203, End position of the expectation window, positive edge
	ExpectWindow-NegEdgeSet-Point	REAL	Set point of the expectation window, negative edge M-0-0007, Control word M-0-0204, Starting position of the expectation window, negative edge M-0-0205, End position of the expectation window, negative edge
	MarkerFailureEnable	BOOL	Marker monitoring: 0:= Not active; 1:= Active M-0-0007, Control word
	MaxNumberOfMarkerFailure	UINT	M-0-0206, Maximum number of marker failures
	Interpolator	MB_INTERPOLATION_MODE	Interpolation mode: 0:= No interpolation; 1:= Linear interpolation M-0-0007, Control word
	TimeStampEnable	BOOL	Enable time stamp 0:= Time stamp is not recorded 1:= Time stamp is recorded M-0-0007, Control word
	DeadTimePosEdge	DINT	Dead-time compensation, positive edge M-0-0200, Dead-time compensation, positive edge
	DeadTimeNegEdge	DINT	Dead-time compensation, negative edge M-0-0201, Dead-time compensation, negative edge
	ProbeData	MB_PROBE_DATA	Signal selection (source of supply) M-0-0006, Maximum number of marker failures
	ModuloValue-ProbeSignal	REAL	Modulo value of the signal to be measured (reference source) (see tab. 6-157 "Measuring signal selection (ProbeData)" on page 339)

12) The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

	Name	Type	Comment ¹²⁾
	PositionEval-Mode	MB_POSITION_EVAL	Activation positive/negative edge M-0-0007, Control word
	DiffEvalMode	MB_DIFF_EVAL	Difference measurement M-0-0007, Control word
VAR_OUTPUT	Done	BOOL	Initialization carried out
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information (" Troubleshooting: " on page 343)
VAR_IN_OUT CONSTANT	TouchProbe	TOUCHPROBE_REF	It contains information on the actual touch probe
	Axis	AXIS_REF	Contains information on the specific axis

Tab. 6-156: Interface of the ML_InitTouchProbe FB

Functional Description

- Assignment of touch probe number and axis,
 - Configuration of the touch probe connection, module/module input,
- The 8 OnBoard inputs of the control and the 16 inputs of up to 4 Fast IO function modules respectively (CFL-01.1-E2) can be selected. The hardware and project planning of the Fast I/O function module are described in the XLC/MLC Functional Description, chapter on function modules (additional components)



One PLC input can be assigned to several touch probes.

- ProbeData: Measuring signal selection
- The available measuring signals of the configured axes are stored in the list parameter C-0-2499, Touch probe, IDN list signal selection.
- The measuring signal is selected via the elements of the "MB_PROBE_DATA" enumeration



Up to 13V04, only A-0-0100, Actual position value, was available.

For 13V06 and higher, the following measuring signals are available:

MB_PROBE_DATA - Element	Parameters	Axis reference
TP_ACT_POS_FEEDBACK1	A-0-0100, Actual position value	Slave axis reference: The modulo value of the slave axis (A-0-0045) and the processing format (A-0-0059.7=0) are determined by the function block.
TP_SYNC_CMD_POS	A-0-2653, ELS, synchronous position command value	
TP_ELS_CMD_POS	A-0-2655, ELS, position command value	

¹²⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

MB_PROBE_DATA - Element	Parameters	Axis reference
TP_RES_MASTER_AXIS_POS	A-0-2650, ELS, resulting master axis position	Master axis reference: The modulo value of the master axis to a local master axis can only be determined by the function block during the active synchronization mode. When executing the function block outside the synchronization mode or when using a distant master axis (cross link C2C), the corresponding modulo value has to be applied at the "ModuloValueProbeSignal".
TP_EFF_MASTER_AXIS_POS	A-0-2651, ELS, effective master axis position	
TP_CAM_SHAFT_PROF_ACCESS_ANGLE	A-0-2656, ELS, cam, access phase	
TP_USER_DATA_CONTAINER_A	A-0-0521, Container of actual user-defined value A	Axis reference depending on the content (configured parameter) of the optional cyclic data container. The corresponding modulo value has to be applied at the "ModuloValueProbeSignal" input.
TP_USER_DATA_CONTAINER_B	A-0-0523, Container of actual user-defined value B	
TP_USER_DATA_CONTAINER_C	A-0-0525, Container of actual user-defined value C	
TP_USER_DATA_CONTAINER_D	A-0-0527, Container of actual user-defined value D	
TP_USER_DATA_CONTAINER_E	A-0-0529, Container of actual user-defined value E	
TP_USER_DATA_CONTAINER_F	A-0-0531, Container of actual user-defined value F	

Tab. 6-157: Measuring signal selection (ProbeData)

- ModuloValueProbeSignal: The modulo value of the measuring signal is used to compute a relative expectation window (see [tab. 6-158 "Expectation window, calculation rule" on page 341](#))
"0" has to be applied for absolutely scaled measuring signals.



The modulo value can always be applied.

For measuring signals with slave axis or master axis reference and the execution of the function block during the active synchronization mode to a local master axis, the modulo value is determined by the function block. (see [tab. 6-157 "Measuring signal selection \(ProbeData\)" on page 339](#))

The "ModuloValueProbeSignal" input is not considered.

- Definition of the expectation window,
The following decisions are to be made separately for the positive and the negative edge:
 - Expectation window off
 - Expectation window **absolute** configured:
 - ExpectWindow***EdgeStart indicates the start position,
 - ExpectWindow***EdgeEnd indicates the end position of the window,
 - ExpectWindow***EdgeSetPoint is not evaluated.

- Expectation window **relative** configured:
 - ExpectWindow***EdgeSetPoint indicates the "set point" absolute,
 - ExpectWindow***EdgeStart indicates the distance between the "set point" and the start position,
 - ExpectWindow***EdgeEnd indicates the distance between the "set point" and the end position of the window,

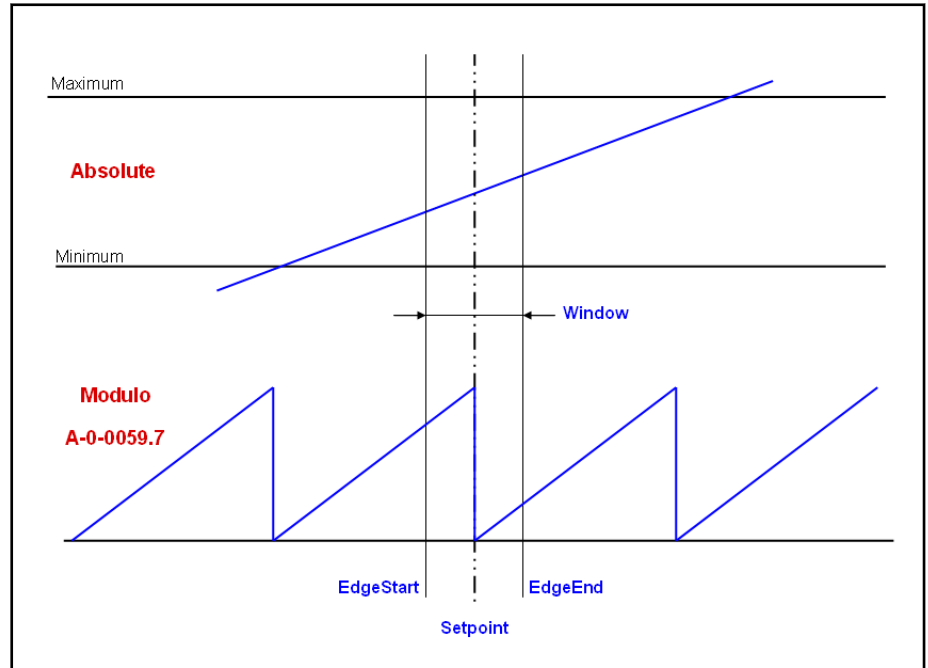


Fig. 6-169: Expectation window of an edge

A-0-0059, Scaling type for position data

A-0-0045, Modulo value

	Modulo format	Absolute	Relative
Positive	A-0-0059.7=0	Start:=ExpectWindowPosEdgeStart	Start:=ExpectWindowPosEdgeSetPoint - ExpectWindowPosEdgeStart
		End:=ExpectWindowPosEdgeEnd	End:=ExpectWindowPosEdgeSetPoint + ExpectWindowPosEdgeEnd
	A-0-0059.7=1	Start:=ExpectWindowPosEdgeStart	Start:= MOD(ExpectWindowPosEdgeSetPoint - ExpectWindowPosEdgeStart + A-0-0045, A-0-0045)
		End:=ExpectWindowPosEdgeEnd	End:= MOD(ExpectWindowPosEdgeSetPoint + ExpectWindowPosEdgeEnd, A-0-0045)

Negative	A-0-0059.7=0	Start:=ExpectWindowNegEdgeStart	Start:=ExpectWindowNegEdgeSetPoint - ExpectWindowNegEdgeStart
		End:=ExpectWindowNegEdgeEnd	End:=ExpectWindowNegEdgeSetPoint + ExpectWindowNegEdgeEnd
	A-0-0059.7=1	Start:=ExpectWindowNegEdgeStart	Start:= MOD(ExpectWindowNegEdgeSetPoint - ExpectWindowNegEdgeStart + A-0-0045, A-0-0045)
		End:=ExpectWindowNegEdgeEnd	End:= MOD(ExpectWindowNegEdgeSetPoint + ExpectWindowNegEdgeEnd, A-0-0045)

Tab. 6-158: Expectation window, calculation rule

- Configuration of the marker error processing,
The marker error monitoring requires an expectation window to be switched on.
If the activated edge is beyond the expectation window, no measurement value is detected, but a marker failure is counted.
If the "maximum number of marker failures" is exceeded, an error message (status bit touch probe) is triggered.
- Interpolation of the axis signal (the axis signal is available at the end of a Sercos cycle, the touch probe signal is displayed at an arbitrary point in time within the Sercos cycle, the axis signal from axis signal (n-1) Sercos cycle, (n) Sercos cycle and time is determined via interpolation).
If the interpolation is switched off, the signal at the end of the Sercos cycle is used.
- Enable time stamp: This is a prerequisite for the provision of the time measurement values.
- Configuration of the runtime compensation.
The compensation can be effected for the positive and the negative edge of the touch probe signal.
- PositionEvalMode: Selection: activation positive/negative edge or both to determine the position.
The determination of the position can be switched off e.g. via time measurement.
- DiffEvalMode (difference measurement):
 - 0 - Difference measurement is switched off, difference signals for ML_TouchProbe and/or ML_TouchProbe_Continuous are not calculated.
 - 1 - Difference between the last two measured values, depending on the activated edge.
Example: Both edges are activated, the difference pos/neg edge, then neg/pos edge, pos/neg edge is registered.
 - In the following figure, the extended methods 2..5 of the difference measurement are displayed:

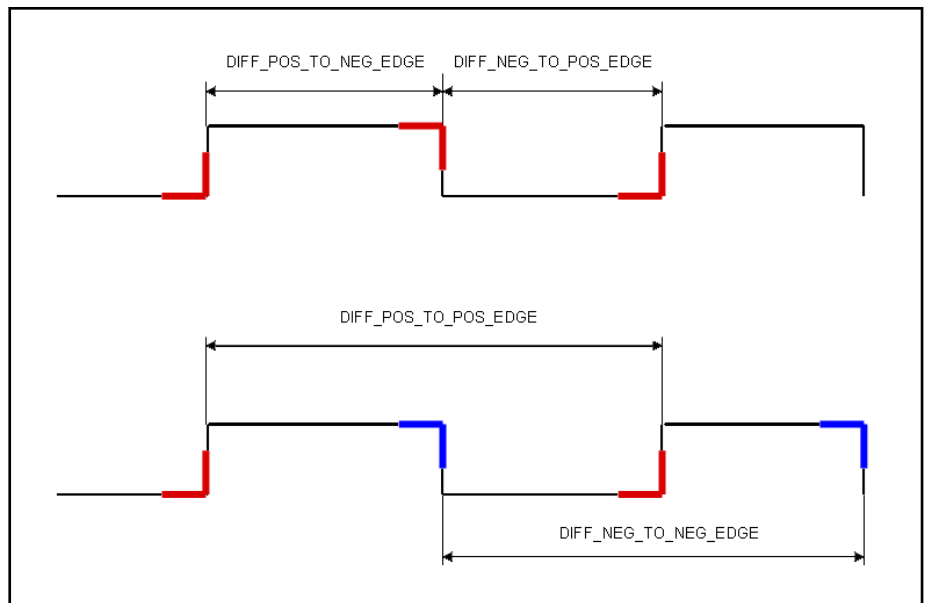


Fig. 6-170: Difference measurement methods

Signal-time diagram

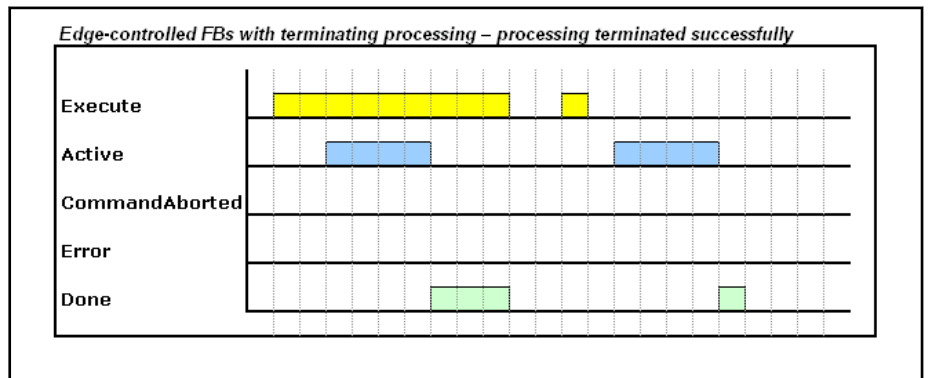


Fig. 6-171: Signal-time diagram: ML_InitTouchProbe

Troubleshooting: The ML_InitTouchProbe function block generates the following messages:

ErrorID	Table	Add1	Add2	Description
RESOURCE_ERROR	MLC_TABLE	16#F0220201	16#00000000	Invalid control number, only the local control is supported
RESOURCE_ERROR	MLC_TABLE	16#F0220202	16#00000000	Invalid axis number, only AXIS_1 .. AXIS_99 are permitted
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220502	16#00000000	ProbeModule out of range
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220503	16#00000000	ProbeInput out of range
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220504	16#00000000	ExpectWindowPosEdgeMode out of range
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220505	16#00000000	ExpectWindowNegEdgeMode out of range
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220514	16#00000000	Interpolation out of range
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220518	16#00000000	ProbeData out of range
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220519	16#00000000	PositionEvalMode out of range
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220520	16#00000000	DiffEvalMode out of range

ErrorID	Table	Add1	Add2	Description
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220521	16#00000000	TOUCHPROBE_REF is not valid or wrong configuration
ACCESS_ERROR	MLC_TABLE	16#F02D700C	16#00000000	sercos: Operation data is write-protected (other settings) Parameters M-0-0006, M-0-0024, M-0-0405 write-protected Parameters C-0-0400, C-0-0503: Different cycle times

For more errors, refer to:

- [MLC_TABLE, page 360](#) - error when accessing the touch probe/control.

6.11.4 ML_TouchProbe

Brief description The ML_TouchProbe FB activates the **one-time** position and time measurement and provides an automated time difference measurement.



The activation of the function block requires the initialization of the touch probe via [ML_InitTouchProbe, page 336](#).

After the initialization, the expectation window can be modified using the function block [ML_WriteExpectWindow, page 349](#).



The touch probe function can be enabled in all Sercos phases using ML_InitTouchProbe.

However, the function blocks ML_TouchProbe and ML_TouchProbeContinuous can only be executed in Sercos phase CP4.

Cause: It can only be written to the parameter "M-0-0024, Command: Activation" in P4. Executing the function block in a different Sercos phase generated the error "F02D700C, sercos: Operation data is write-protected (other settings)".

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen/TouchProbe

Tab. 6-159: Library assignment

Interface description

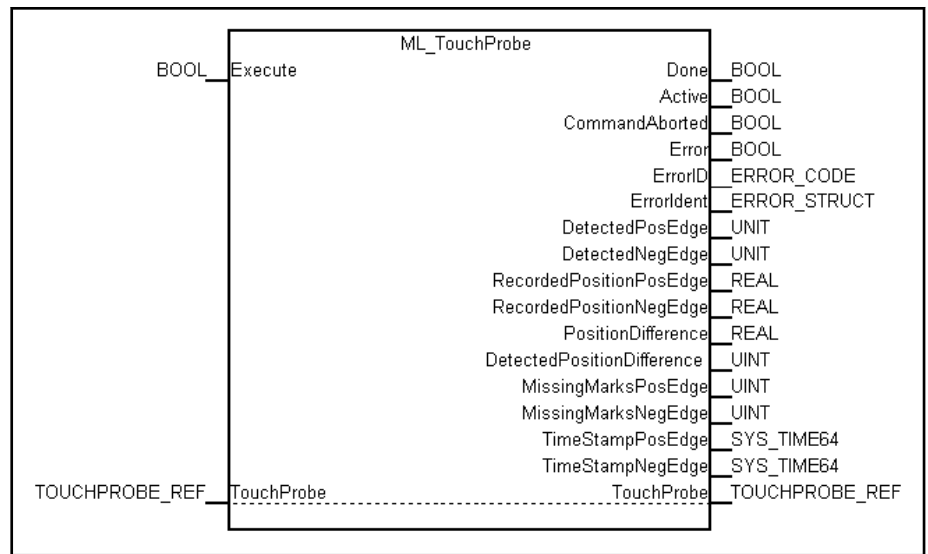


Fig. 6-172: ML_TouchProbe function block

	Name	Type	Comment ¹³⁾
VAR_INPUT	Execute	BOOL	Starts the measurement at the rising edge
VAR_OUTPUT	Done	BOOL	Measurement is carried out
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	DetectedPosEdge	UINT	M-0-0409, Counter of measured values, positive edge
	DetectedNegEdge	UINT	M-0-0410, Counter of measured values, negative edge
	RecordedPositionPosEdge	REAL	Measured value memory for positive edge M-0-0130, Measured value, positive edge
	RecordedPositionNegEdge	REAL	Measured value memory for negative edge M-0-0131, Measured value, negative edge
	PositionDifference	REAL	Difference measured value memory M-0-0132, Difference value
	DetectedPositionDifference	UINT	Counter for difference value recording M-0-0411, Counter of difference values
	MissingMarksPosEdge	UINT	M-0-0224, Number of marker failures, positive edge

¹³⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

	Name	Type	Comment ¹³⁾
	MissingMarks-NegEdge	UINT	Number of marker failures, negative edge M-0-0225, Number of marker failures, negative edge
	TimeStampPosEdge	SYS_TIME64	Time stamp memory, positive edge M-0-0140, Time stamp, positive edge
	TimeStampNegEdge	SYS_TIME64	Time stamp memory, negative edge M-0-0141, Time stamp, negative edge
VAR_IN_OUT CONSTANT	TouchProbe	TOUCHPROBE_REF	It contains information on the actual touch probe

Tab. 6-160: Interface of the ML_TouchProbe FB

Signal-time diagram

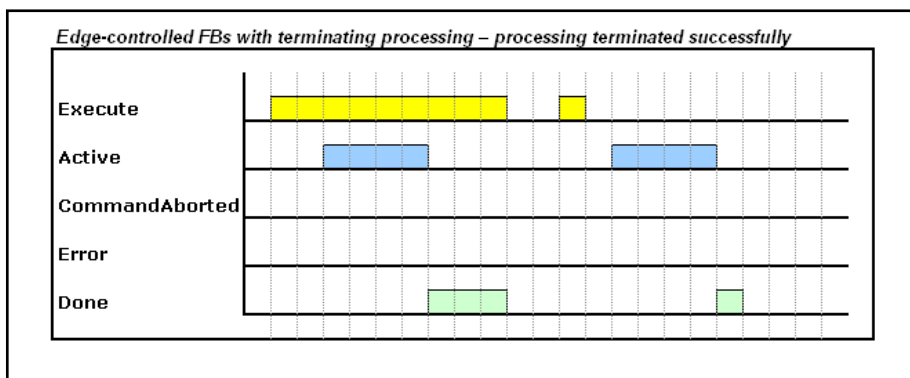


Fig. 6-173: Signal-time diagram: ML_TouchProbe

Functional Description

Measured value acquisition mode

- **Single shot measurement** - After enabling the respective touch probe (M-0-0405, Enabling), ML_TouchProbe, a value is registered only with the first measured value trigger. Prior to every further measurement, a new enabling of the touch probe is necessary!
- **Continuous measurement** - After enabling the respective touch probe (M-0-0405), [ML_TouchProbeContinuous](#), page 347., a new measured value is only applied during the first measured value trigger.

Troubleshooting: The ML_TouchProbe function block generates the following messages:

ErrorID	Table	Add1	Add2	Description
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220519	16#00000000	PositionEvalMode out of range
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220521	16#00000000	TOUCHPROBE_REF is not valid or wrong configuration
ACCESS_ERROR	MLC_TABLE	16#F02D700C	16#00000000	sercos: Operation data is write-protected (other settings) Parameters M-0-0006, M-0-0024, M-0-0405 write-protected Parameters C-0-0400, C-0-0503: Different cycle times

For more errors, refer to:

- [MLC_TABLE](#), page 360 - error when accessing the touch probe/control.

¹³⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

6.11.5 ML_TouchProbeContinuous

Brief description The ML_TouchProbeContinuous function block activates the **continuous** position and time measurement and provides an automated time difference measurement.



The activation of the function block requires the initialization of the touch probe via [ML_InitTouchProbe](#), page 336,.

After the initialization, the expectation window can be modified using the function block [ML_WriteExpectWindow](#), page 349,.



The touch probe function can be enabled in all Sercos phases using ML_InitTouchProbe.

However, the function blocks ML_TouchProbe and ML_TouchProbeContinuous can only be executed in Sercos phase CP4.

Cause: It can only be written to the parameter "M-0-0024, Command: Activation" in P4. Executing the function block in a different Sercos phase generated the error "F02D700C, sercos: Operation data is write-protected (other settings)".

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen/TouchProbe

Tab. 6-161: Library assignment

Interface description

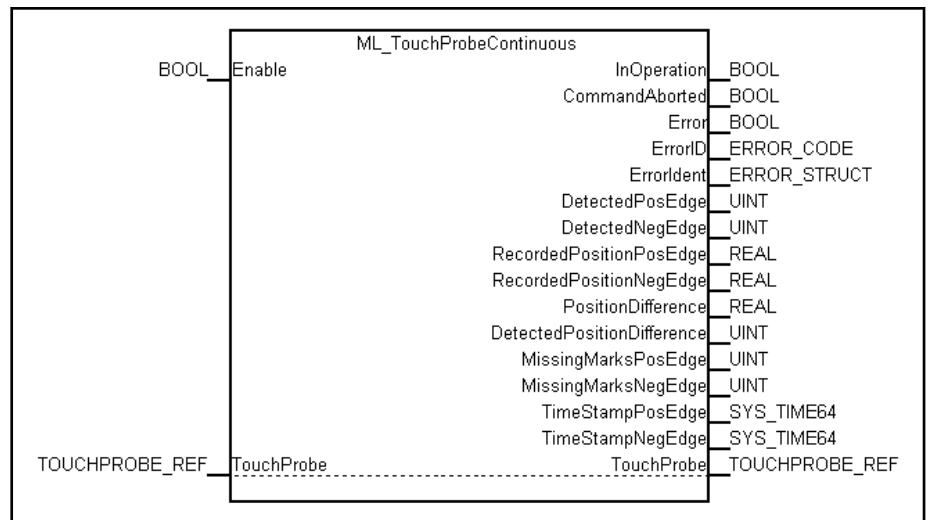


Fig. 6-174: ML_TouchProbeContinuous FB

	Name	Type	Comment ¹⁴⁾
VAR_INPUT	Enable	BOOL	Starts the measurement at the rising edge
VAR_OUTPUT	InOperation	BOOL	Measurement is running
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance

¹⁴⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

	Name	Type	Comment ¹⁴⁾
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	DetectedPosEdge	UINT	M-0-0409, Counter of measured values, positive edge
	DetectedNegEdge	UINT	M-0-0410, Counter of measured values, negative edge
	RecordedPositionPosEdge	REAL	Measured value memory for positive edge M-0-0130, Measured value, positive edge
	RecordedPositionNegEdge	REAL	Measured value memory for negative edge M-0-0131, Measured value, negative edge
	PositionDifference	REAL	Difference measured value memory M-0-0132, Difference value
	DetectedPositionDifference	UINT	Counter for difference value recording M-0-0411, Counter of difference values
	MissingMarkersPosEdge	UINT	Number of marker failures, positive edge M-0-0224, Number of marker failures, positive edge
	MissingMarkersNegEdge	UINT	Number of marker failures, negative edge M-0-0225, Number of marker failures, negative edge
	TimeStampPosEdge	SYS_TIME64	Time stamp memory, positive edge M-0-0140, Time stamp, positive edge
	TimeStampNegEdge	SYS_TIME64	Time stamp memory, negative edge M-0-0141, Time stamp, negative edge
VAR_IN_OUT CONSTANT	TouchProbe	TOUCHPROBE_REF	It contains information on the actual touch probe

Tab. 6-162: Interface of the ML_TouchProbeContinuous FB

Signal-time diagram

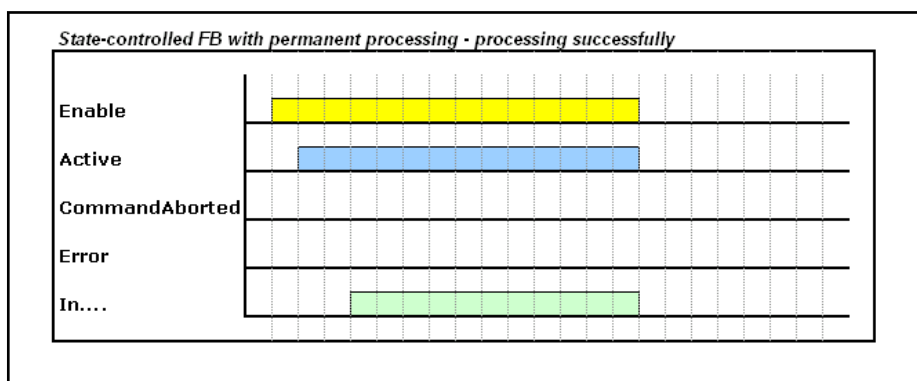


Fig. 6-175: Signal-time diagram: ML_TouchProbeContinuous

¹⁴⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

Functional Description *Measured value acquisition mode*

- **Single shot measurement** - After enabling the respective touch probe (M-0-0405), Enabling, [ML_TouchProbe](#), page 344,, a value is registered only with the first measured value trigger. Prior to every further measurement, a new enabling of the touch probe is necessary!
- **Continuous measurement** - After the touch probe (M-0-0405), [ML_TouchProbeContinuous](#), was enabled, a new measured value is only applied during the first measured value trigger.

Troubleshooting: The ML_TouchProbeContinuous function block generates the following messages:

ErrorID	Table	Add1	Add2	Description
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220521	16#00000000	TOUCHPROBE_REF is not valid or wrong configuration
ACCESS_ERROR	MLC_TABLE	16#F02D700C	16#00000000	sercos: Operation data is write-protected (other settings) Parameters M-0-0006, M-0-0024, M-0-0405 write-protected Parameters C-0-0400, C-0-0503: Different cycle times

For more errors, refer to:

- [MLC_TABLE](#), page 360 - error when accessing the touch probe/control.

6.11.6 ML_WriteExpectWindow

Brief description The function block ML_WriteExpectWindow configures the expectation window without a complete initialization run with the [ML_InitTouchProbe](#), page 336, being necessary.



If the "Enable" of the FB is activated, modifications of the inputs are accepted **immediately**.

Target system	Library/folder
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_PLCopen/TouchProbe

Tab. 6-163: Library assignment

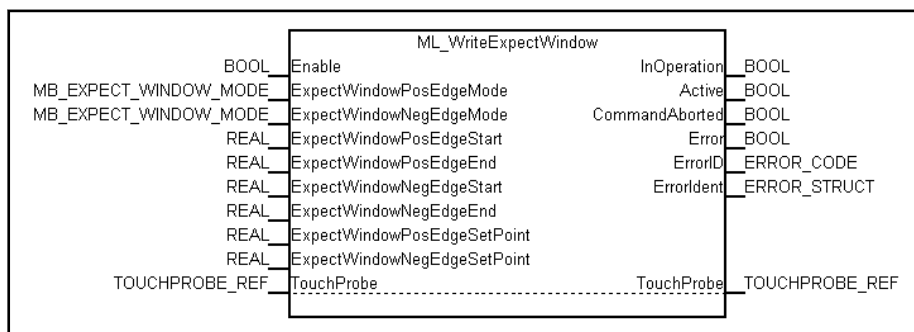
Interface description

Fig. 6-176: ML_WriteExpectWindow FB

	Name	Type	Comment ¹⁵⁾
VAR_INPUT	Enable	BOOL	Starts the motion at a rising edge
	ExpectWindow-PosEdgeMode	MB_EXPECT_WINDOW_MODE	Activates touch probe window at positive edge M-0-0007, Control word
	ExpectWindow-NegEdgeMode	MB_EXPECT_WINDOW_MODE	Activates touch probe window at negative edge M-0-0007, Control word
	ExpectWindow-PosEdgeStart	REAL	Start position expectation window, positive edge M-0-0007, Control word M-0-0202, Starting position of the expectation window, positive edge
	ExpectWindow-PosEdgeEnd	REAL	End position expectation window, positive edge M-0-0007, Control word M-0-0203, End position of the expectation window, positive edge
	ExpectWindow-NegEdgeStart	REAL	Start position expectation window, negative edge M-0-0007, Control word M-0-0204, Starting position of the expectation window, negative edge
	ExpectWindow-NegEdgeEnd	REAL	End position expectation window, negative edge M-0-0007, Control word M-0-0205, End position of the expectation window, negative edge
	ExpectWindow-PosEdgeSet-Point	REAL	Set point of the positive expectation window M-0-0007, Control word M-0-0202, Starting position of the expectation window, positive edge M-0-0203, End position of the expectation window, positive edge
	ExpectWindow-NegEdgeSet-Point	REAL	Set point of the negative expectation window M-0-0007, Control word M-0-0204, Starting position of the expectation window, negative edge M-0-0205, End position of the expectation window, negative edge
VAR_OUTPUT	InOperation	BOOL	Desired configuration reached
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information (" Troubleshooting: " on page 352)
VAR_IN_OUT CONSTANT	TouchProbe	TOUCHPROBE_REF	It contains information on the actual touch probe

Tab. 6-164: Interface of FB ML_WriteExpectWindow

15) The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

Functional Description The expectation window can be redefined for the created touch probe number using this FB.

The following decisions are to be made separately for the positive and the negative edge:

Definition of the expectation window

- Expectation window off
- Expectation window **absolute** configured:
 - ExpectWindow***EdgeStart indicates the start position,
 - ExpectWindow***EdgeEnd indicates the end position of the window,
 - ExpectWindow***EdgeSetPoint is not evaluated.
- Expectation window **relative** configured:
 - ExpectWindow***EdgeSetPoint indicates the "set point" absolute,
 - ExpectWindow***EdgeStart indicates the distance between the "set point" and the start position,
 - ExpectWindow***EdgeEnd indicates the distance between the "set point" and the end position of the window,

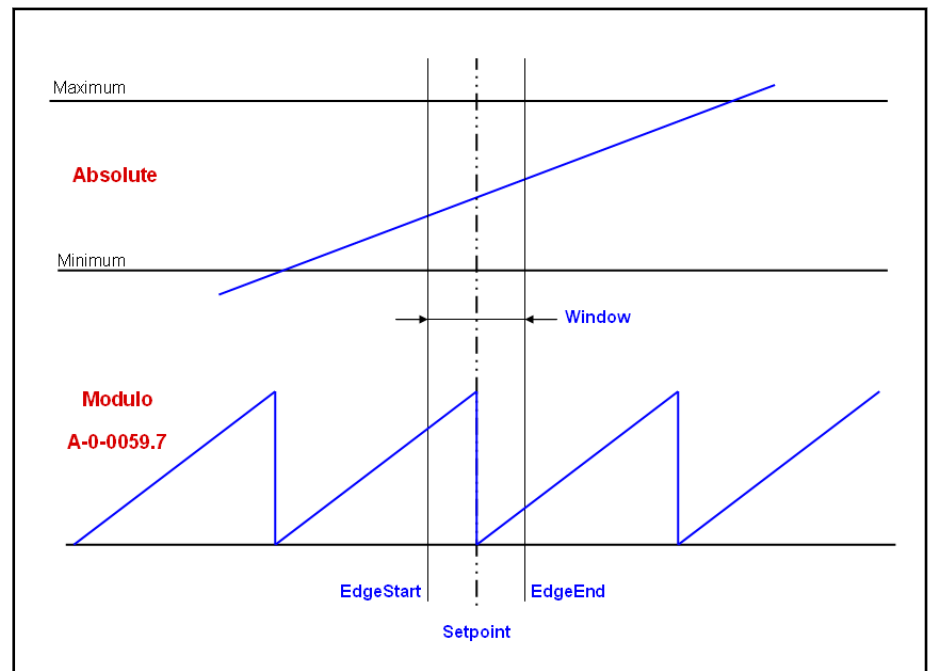


Fig. 6-177: Expectation window of an edge

A-0-0059, Scaling type for position data

A-0-0045, Modulo value

	Modulo format	Absolute	Relative
Positive	A-0-0059.7= 0	Start:=ExpectWindowPosEdgeStart	Start:=ExpectWindowPosEdgeSetPoint - ExpectWindowPosEdgeStart
		End:=ExpectWindowPosEdgeEnd	End:=ExpectWindowPosEdgeSetPoint + ExpectWindowPosEdgeEnd
	A-0-0059.7= 1	Start:=ExpectWindowPosEdgeStart	Start:= MOD(ExpectWindowPosEdgeSetPoint - ExpectWindowPosEdgeStart + A-0-0045, A-0-0045)
		End:=ExpectWindowPosEdgeEnd	End:= MOD(ExpectWindowPosEdgeSetPoint + ExpectWindowPosEdgeEnd, A-0-0045)
Negative	A-0-0059.7= 0	Start:=ExpectWindowNegEdgeStart	Start:=ExpectWindowNegEdgeSetPoint - ExpectWindowNegEdgeStart
		End:=ExpectWindowNegEdgeEnd	End:=ExpectWindowNegEdgeSetPoint + ExpectWindowNegEdgeEnd
	A-0-0059.7= 1	Start:=ExpectWindowNegEdgeStart	Start:= MOD(ExpectWindowNegEdgeSetPoint - ExpectWindowNegEdgeStart + A-0-0045, A-0-0045)
		End:=ExpectWindowNegEdgeEnd	End:= MOD(ExpectWindowNegEdgeSetPoint + ExpectWindowNegEdgeEnd, A-0-0045)

Tab. 6-165: Expectation window, calculation rule

Signal-time diagram

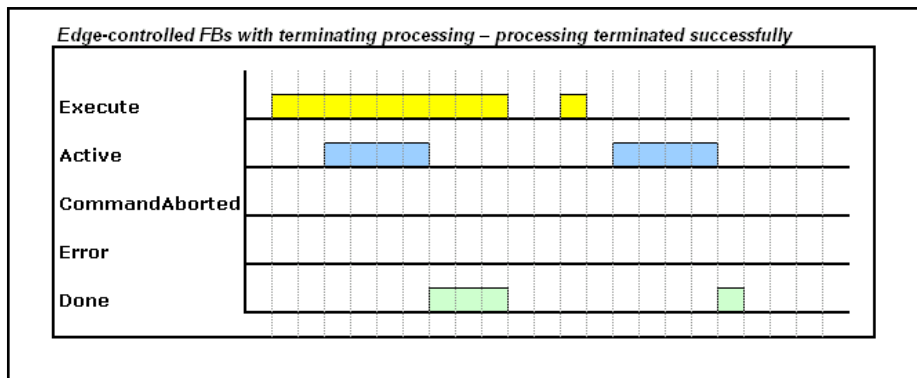


Fig. 6-178: Signal-time diagram: ML_WriteExpectWindow

Troubleshooting: The ML_WriteExpectWindow function block generates the following error messages:

ErrorID	Table	Add1	Add2	Description
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220504	16#00000000	ExpectWindowPosEdgeMode out of range
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220505	16#00000000	ExpectWindowNegEdgeMode out of range
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220506	16#00000000	ExpectWindowPosEdgeStart out of range
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220507	16#00000000	ExpectWindowPosEdgeEnd out of range
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220508	16#00000000	ExpectWindowNegEdgeStart out of range
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220509	16#00000000	ExpectWindowNegEdgeEnd out of range
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220510	16#00000000	ExpectWindowPosEdgeSetPoint out of range

ErrorID	Table	Add1	Add2	Description
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220511	16#00000000	ExpectWindowNegEdgeSetPoint out of range
INPUT_RANGE_ERROR	MLC_TABLE	16#F0220521	16#00000000	TOUCHPROBE_REF is not valid or wrong configuration

For more errors, refer to:

- [MLC_TABLE, page 360](#) - error when accessing the touch probe/control.

6.12 Time required to process Motion function blocks in the IndraMotion MLC

A Sercos and a Motion cycle time of 2 milliseconds is assumed.

The Motion-synchronous MotionTask from which measurements are conducted is also processed every 2 milliseconds.



The bus cycle clock has to be coupled to the Motion task (2ms). The bus cycle clock is coupled to the fastest PLC task (10ms) by default.

The timings for the onboard I/Os are directly proportional to the cycle time of the task in which they are used. All measurements are carried out in as 2ms Motion task. If the cycle time is reduced by 50%, the onboard I/Os are twice the fast.

The following table shows average processing times of the respective function blocks **from the point of view of the PLC**.

Example:

Time required and operating mode of the MC_Power from the point of view of the PLC

At the "Enable" input, the block instance of the MC_Power receives a 0→1 transition. It is processed.

The time that passes between reaching the network containing the function block and leaving it averages 4.6 µs. The function block transmits a task to the drive electronics.

Executing the task in the drive electronics providing a feedback for the status change at the "status" output of the function block instance of the MC_Power after the performance release for the drive requires much more time and many PLC cycles.

Aktion / T _{avr} [µs]	14VRS, L25	14VRS, L45	14VRS, L65
MC_Power, real axis	21,1	8,6	3,8
MC_MoveAbsolute, real axis	21,7	8,4	4,3
MC_MoveAbsolute, virtual axis	9,3	3,1	1
MC_MoveRelative, real axis	21,4	8,4	3,6
MC_MoveRelative, virtual axis	10,3	3	1
MC_MoveVelocity, real axis	21,4	8,1	4,2
MC_MoveVelocity, virtual axis	10,3	3,2	1

Aktion / T _{avr} [μs]	14VRS, L25	14VRS, L45	14VRS, L65
MC_MoveAdditive, real axis	20,8	8,6	3,9
MC_MoveAdditive, virtual axis	7,3	2,6	0,7
MC_Stop, real axis	22,4	8,7	4,3
MC_Stop, virtual axis	11,4	3,3	1
MC_GearIn	22,1	8,6	4
MC_GearOut	20,9	9,9	4,5
Datenzugriff, Direktvariable	13,6	5	2,6

Tab. 6-166: Average time required to process different function blocks with 50,000-fold repetition of the actuation

The time required results from the sum of the time required for the individual statements, functions to be processed and function block instances.

If further tasks are created, their switching time and the creation of task-local image memories lead to additional time required.

A particularly critical situation results from the switching to an error:

Example:

Simultaneous switching of an MC_Power and an MC_MoveAbsolute of the same axis

The starting point is the situation

- "MC_Power.Enable"= FALSE,
- "MC_MoveAbsolute.Execute"= FALSE and
- "MC_MoveAbsolute.Error"= FALSE.

The inputs "MC_Power.Enable" and "MC_MoveAbsolute.Execute" are switched to TRUE and both function block instances are processed.

This starts "MC_MoveAbsolute" **without** releasing the necessary capacity. It switches to an error.



The time required for "MC_MoveAbsolute" increases in this case to approximately three times the value of the normal processing.

Reading/writing parameters

The following shows the dependency between time requirements and the drive types MPB07 (SIS protocol) / MPB18 (SIP protocol)

Motion cycle time 2 ms, Sercos cycle time 2 ms, number of measurements 1000, PLC Task 1 ms, priority 1, cyclic

Drive type MPB07 (SIS protocol)

Parameters	Number of bytes	Function block	Average	Minimum	Maximum
S-0-0256	UDint, 4	MB_ReadParameter	18ms	11 ms	35 ms
P-0-0072	List, 1024	MB_WriteListParameter	280 ms	263 ms	309 ms
P-0-0072	List, 1024	MB_ReadListParameter	147 ms	135ms	179 ms
A-0-0224	Real, 4	MB_ReadRealParameter	3ms	3ms	18ms
A-0-0101	LReal, 8	MB_ReadLrealParameter	3ms	3ms	18ms

Parameters	Number of bytes	Function block	Average	Minimum	Maximum
A-0-0023	Hex, 4	MB_ReadParameter	3ms	3ms	18ms
C-0-0852	LReal, 8	MB_WriteLrealParameter	2 ms	2 ms	2 ms
A-0-3020	LReal, 8	MB_WriteLrealParameter	29 ms	14 ms	91ms
A-0-0030	Real, 4	MB_WriteRealParameter	4 ms	4 ms	18ms
A-0-0002	String	MB_ReadStringParameter	2 ms	2 ms	15 ms

Drive type MPB18 (SIS protocol)

Parameters	Number of bytes	Function block	Average	Minimum	Maximum
S-0-0256	UDint, 4	MB_ReadParameter	11.5 ms	10 ms	18ms
P-0-0072	List, 1024	MB_WriteListParameter	18.5 ms	18ms	24 ms
P-0-0072	List, 1024	MB_ReadListParameter	9 ms	7ms	16 ms
A-0-0224	Real, 4	MB_ReadRealParameter	3ms	3ms	3ms
A-0-0101	LReal, 8	MB_ReadLrealParameter	3ms	3ms	4 ms
A-0-0023	Hex, 4	MB_ReadParameter	3ms	3ms	4 ms
C-0-0852	LReal, 8	MB_WriteLrealParameter	2 ms	2 ms	3ms
A-0-3020	LReal, 8	MB_WriteLrealParameter	29.5 ms	14 ms	128 ms
A-0-0030	Real, 4	MB_WriteRealParameter	4 ms	4 ms	4 ms
A-0-0002	String	MB_ReadStringParameter	2 ms	2 ms	3ms

6.13 Axis types and function blocks supported by these types

The following table provides an overview on the current state of the support of the function blocks by the individual axis types.

If these function blocks require a master and a slave axis, the statement relates to the slave axis.



Unsupported function blocks generate the error: "F2229204, Command not supported by axis".



Open-loop

Only function blocks without position control are possible for real axes without position feedback (open-loop).



from 14VRS

- If a real axis, IndraDrive, MPx06 / MPx07 / MPx08 / MPx16 / MPx17 / MPx18 / MPx20, is calculated on the control using the command value interface with the drive; refer to the column **IndraDrive ► IPO Control**.
- If a real axis, IndraDrive, MPx06 / MPx07 / MPx08 / MPx16 / MPx17 / MPx18 / MPx20, is calculated on the control using the PackProfile; refer to the column **SercosDrive**

Function block	Real axes					Virtual axes	Encode/ link* axis	Controller axis *3)
	IndraDrive		HydraulicDrive*		Sercos- Drive			
	Interpolation Drive	Interpolation Control	Interpolation Drive	Interpolation Control				
MB_ChangeCamData	X	X	-	X	X	X	-	X
MB_ChangeFlexEventSet	-	X	-	X	X	X	-	X
MB_ChangeFlexProfilSet	-	X	-	X	X	X	-	X
MB_ChangeProfilSet	-	X	-	X	X	X	-	X
MB_ChangeProfilStep	-	X	-	X	X	X	-	X
MB_GearInPos	X	X	-	X	X	X	-	X
MB_Home	X	X	X	X	X	X	-	X
MB_MotionProfile	-	X	-	X	X	X	-	X
MB_Phasing	X ¹⁾	X	X	X	X	X	-	X
MB_PhasingSlave	X ¹⁾	X	X	X	X	X	-	X
MB_SetPositionControlMode ⁵⁾	X	X	-	-	-	-	-	-
MC_AddAxisToGroup * 4)	X	X	X	X	X	X	-	X
MC_CamIn	X	-	-	-	-	-	-	-
MC_CamOut	X	-	-	-	-	-	-	-
MC_GearIn	X ²⁾	X ²⁾	-	X	X	X	-	X
MC_GearOut	X ²⁾	X ²⁾	-	X	X	X	-	X
MC_MoveAbsolute	X	X	X	X	X	X	-	X
MC_MoveAdditive	X	X	X	X	X	X	-	X
MC_MoveRelative	X	X	X	X	X	X	-	X
MC_MoveVelocity	X ²⁾	X ²⁾	X	X	X	X	-	X
MC_Power	X ²⁾	X ²⁾	X	X	X	-	-	X
MC_RemoveAxisFromGroup * 4)	X	X	X	X	X	X	-	X
MC_Reset	X ²⁾	X ²⁾	X	X	X	X	-	X
MC_Stop	X ²⁾	X ²⁾	X	X	X	X	-	X
MC_TorqueControl	X ¹⁾	X	X	X	X	-	-	X
ML_FlexProfile	-	X	-	X	X	X	-	X
ML_OpenCyclicAnalogChannel	-	-	X	X	-	-	-	X
ML_OpenCyclicPositionChannel	X	X	X	X	X	X	-	X
ML_OpenCyclicTorqueChannel	X	X	X	X	-	-	-	X
ML_OpenCyclicVelocityChannel	X	X	X	X	-	-	-	X
ML_WriteCyclicAnalog	-	-	X	X	-	-	-	X
ML_WriteCyclicPosition	X	X	X	X	X	X	-	X

Function block	Real axes					Virtual axes	Encode/ link* axis	Controller axis *3)
	IndraDrive		HydraulicDrive*		Sercos- Drive			
	Interpolation Drive	Interpolation Control	Interpolation Drive	Interpolation Control				
ML_WriteCyclicTorque	X	X	X	X	-	-	-	X
ML_WriteCyclicVelocity	X	X	X	X	-	-	-	X
ML_ControlOff	X	X	X	X	X	X	-	X
ML_ControlOn	X	X	X	X	X	X	-	X
ML_ControlWriteActualData	-	-	-	-	-	-	-	X
ML_OpenCyclicAnalogChannelCtrl	-	-	X	X	-	-	-	X
ML_OpenCyclicPositionChannelCtrl	X	X	X	X	X	X	-	X
ML_OpenCyclicTorqueChannelCtrl	X	X	X	X	-	-	-	X
ML_OpenCyclicVelocityChannelCtrl	X	X	X	X	-	X	-	X
ML_WriteCyclicAnalogCtrl	-	-	X	X	-	-	-	X
ML_WriteCyclicPositionCtrl	X	X	X	X	X	X	-	X
ML_WriteCyclicTorqueCtrl	X	X	X	X	-	-	-	X
ML_WriteCyclicVelocityCtrl	X	X	X	X	-	X	-	X

* only for MLC

1) Not MPx03

2) Also with OpenLoop

3) Depending on the controller implementation or the functional scope of the device whose actual values are transferred to the controller axis (ML_ControlWriteActualData with link)

4) As of 13VRS

5) As of 13VRS and > MPx06

Tab. 6-167: Function blocks in IndraLogic XLC/IndraMotion MLC

6.14 Overview: Diagnostics

6.14.1 SERCOS_TABLE, ERROR_TABLE 16#0010

SERCOS_TABLE - ERROR_TABLE, table number 16#0010

Controls of the system IndraLogic XLC/IndraMotion MLC extend the four-digit Sercos error code to an eight-digit MLC error code as follows:

Supplements are "F" for error, "0" as severity (severity of the error) and "2D" as a sign for the origin. Subsequently, the four Sercos digits follow.

Severity 0 signifies no response at the axes/drives.

The eight-digit code is used at the IndraWorks surface (error memory).

At the block outputs, the table number "ErrorIdent/Table", 16#0030 (MLC_TABLE) or 16#0010 (SERCOS_TABLE) determines which presentation is used.

SERCOS_ TABLE ErrorIdent / Additional1	Description	MLC_ TABLE ErrorIdent / Additional1
	General error	
0000	sercos: No error in the service channel	F02D0000
0001	sercos: Service channel not open	F02D0001
0003	Max. count of parallel parameter requests exceeded	F02D0003
0007	Parameter access to remote control not allowed	F02D0007
0008	DSH-Message Index out of Range	F02D0008
0009	sercos: Invalid access to closing the service channel	F02D0009
00A0	sercos: Parameter service channel is not open	F02D00A0
00B0	sercos: Parameter element does not exist	F02D00B0
00C0	sercos: Selected drive does not exist	F02D00C0
	Element 1 (ident number)	
1001	sercos: No IDN	F02D1001
1009	sercos: Invalid access to element 1	F02D1009
	Element 2 (name)	
2001	sercos: No name	F02D2001
2002	sercos: Name transmission too short	F02D2002
2003	sercos: Name transmission too long	F02D2003
2004	sercos: Name cannot be changed (read only)	F02D2004
2005	sercos: Name is write-protected at this time	F02D2005
	Element 3 (attribute)	
3002	sercos: Attribute transmission too short	F02D3002
3003	sercos: Attribute transmission too long	F02D3003
3004	sercos: Attribute cannot be changed (read only)	F02D3004
3005	sercos: Attribute is write-protected at this time	F02D3005
	Element 4 (unit)	
4001	sercos: No units	F02D4001
4002	sercos: Unit transmission too short	F02D4002
4003	sercos: Unit transmission too long	F02D4003
4004	sercos: Unit cannot be changed (read only)	F02D4004
4005	sercos: Unit is write-protected at this time	F02D4005
	Element 5 (minimum input value)	
5001	sercos: No minimum input value	F02D5001
5002	sercos: Min. input value transmission too short	F02D5002

SERCOS_ TABLE ErrorIdent / Additional1	Description	MLC_ TABLE ErrorIdent / Additional1
5003	sercos: Min. input value transmission too long	F02D5003
5004	sercos: Min. input value cannot be changed (read only)	F02D5004
5005	sercos: Min. input value is write-protected at this time	F02D5005
	Element 6 (maximum input value)	
6001	sercos: No maximum input value	F02D6001
6002	sercos: Max. input value transmission too short	F02D6002
6003	sercos: Max. input value transmission too long	F02D6003
6004	sercos: Max. input value cannot be changed (read only)	F02D6004
6005	sercos: Max. input value is write-protected at this time	F02D6005
	Element 7 (operating date)	
7001	sercos: No operation data	F02D7001
7002	sercos: Operation data transmission too short	F02D7002
7003	sercos: Operation data transmission too long	F02D7003
7004	sercos: Operation data cannot be changed (read only)	F02D7004
7005	sercos: Operation data is write-protected at this time	F02D7005
7006	sercos: Operation data is smaller than the min input value	F02D7006
7007	sercos: Operation data is greater than the max input value	F02D7007
7008	sercos: Invalid operation data	F02D7008
7009	sercos: Operation data write-protected by a password	F02D7009
700A	sercos: Operation data is write-protected (cycl. conf)	F02D700A
700B	sercos: Invalid indirect addr., (data container, list)	F02D700B
700C	sercos: Operation data is write-protected (other settings)	F02D700C
700D	sercos: Invalid floating point number	F02D700D
700E	sercos: Reserved	F02D700E
700F	sercos: Operation data is write-protected at OM	F02D700F
7010	sercos: Procedure command already active	F02D7010
7011	sercos: Procedure command not interruptible	F02D7011
7012	sercos: Procedure command at this time not executable	F02D7012
7013	sercos: Procedure command not executable	F02D7013
710C	sercos: Data exceed maximal range	F02D710C
8001	sercos: Access temporary not possible	F02D8001

SERCOS_ TABLE ErrorIdent / Additional1	Description	MLC_ TABLE ErrorIdent / Additional1
8096	sercos: Device deactivated	F02D8096
8097	sercos: Device does not exist	F02D8097

Tab. 6-168: Sercos error and their us in the system IndraLogic XLC/IndraMotion MLC

6.14.2 MLC_TABLE, ERROR_TABLE 16#0030

MLC_TABLE - ERROR_TABLE, Table number 16#0030

The messages, warnings and error messages output by function blocks are included in the Diagnostic system (MLC_Table).

Rexroth IndraLogic XLC IndraMotion MLC 14VRS Diagnostics

DOK-XLCMLC-DIAG****V14-RERS-EN-P, R911341481

Individual function blocks of the ML_PLCopen library can use other error tables:

- F_RELATED_TABLE, error table 16#170

7 RMB_PLCopenFieldBus.library

7.1 Overview and introduction



The basic function package PLC is sufficient for the library of the IndraLogic XLC/IndraMotion MLC 14VRS system.

Significance of the function block prefixes

The RMB_PLCopenFieldBus library contains function blocks communicating with field bus axes.

- **Prefix MC_**
This prefix indicates that this function block is 100% PLCopen-compliant.
- **Prefix MB_**
This prefix indicates that this function block is executed according to the systematics of the PLCopen, deviates or supplements it or is not defined.
- **Prefix MP_**
This prefix is used for system-internally used function blocks.

The function blocks, functions, data types and constants are included in the following folders and Global Variable Lists:

- [Parameter address constants, page 361](#)
- [Data types, page 362,](#)
- [Function blocks for single-axis control \(Motion_FBs\), page 365,](#)
- [Function blocks for parameter access \(Parameter_FBs\), page 385](#)
- [System_FBs , page 420..](#)

7.2 Parameter address constants

7.2.1 Access to data of a (field bus) drive

Drive data can be obtained via read-only and partially also write parameter access. The following parameter types are distinguished for the field bus drives at the Profibus:

S-parameters The S-parameters correspond to the standard parameters specified in the Sercos interface standard.

P-parameters The P-parameters are manufacturer-specific and product-specific parameters defined by the drive manufacturer according to the Sercos interface standard.

Each system parameter is identified by its unique ID number.

P- and S-parameter declaration files for Global Constants are available for the user.

Declaration files for Global Constants

- [GVL_S_Param_ID, page 361,](#)
- [GVL_P_Param_ID, page 362](#)

7.2.2 GVL_S_Param_ID

Brief description The global constant list contains the address constant assignment to the respective parameter.

The parameter constant name is

S-0-0001 ⇒ FP_S_0_0001 etc.

Number and parameter constant name can be used alternatively.

Declaring the global constant list GVL_S_Param_ID

```
VAR GLOBAL CONSTANT
  FP_S_0_0000: MB_IDN:= 0 ; (* Dummy S parameter *)
  FP_S_0_0001: MB_IDN:= 1 ; (* NC cycle time (TNcyc) *)
  FP_S_0_0002: MB_IDN:= 2 ; (* SERCOS cycle time (TScyc) *)
  (* ..... *)
END_VAR
```

7.2.3 GVL_P_Param_ID

Brief description The global constant list contains the address constant assignment to the respective parameter.

The parameter constant name is

P-0-0001 ⇒ FP_P_0_0001 etc.

Number and parameter constant name can be used alternatively.

Declaring the global constant list GVL_P_Param_ID

```
VAR GLOBAL CONSTANT
  FP_P_0_0001: MB_IDN:= 32769 ; (* Switching frequency of the power output stage *)
  FP_P_0_0002: MB_IDN:= 32770 ; (* SPI flash aging counter *)
  FP_P_0_0003: MB_IDN:= 32771 ; (* Status of parameter buffering *)
  FP_P_0_0004: MB_IDN:= 32772 ; (* Velocity loop smoothing time constant *)
  (* ..... *)
END_VAR
```

7.3 Data types

7.3.1 Data types, general

The folder contains data types that support working with the library elements of the "RMB_PLCopenFieldbus.library".

Only the following two data types are important for the user.

- [MB_IDN \(field bus drives\), page 362](#), and
- [AXIS_REF \(field bus drives\), page 363](#), and



Elements other than the elements described may not be used due to the side effects, including malfunctions in the motion sequence of the axes!

7.3.2 MB_IDN

Brief description A system-specific parameter length can be defined with MB_IDN. These are 64 bits for the parameters of the IndraLogic XLC/IndraMotion MLC.

S- and P-parameters are automatically filled with up to 64 bits.

Library	Range
RMB_PLCopenFieldbus	Data types

Tab. 7-1: Library assignment

Type definition

```

TYPE
  MB_IDN : LINT ;
END_TYPE

```

Offsets der Parameter

Parameter	Decimal	Hexadecimal
S-0-0000	0	16#0
P-0-0000	32768	16#8000
A-0-0000	4294967296	16#1 0000 0000
C-0-0000	8589934592	16#2 0000 0000
K-0-0000	184683593728	16#2B 0000 0000
M-0-0000	193273528320	16#2D 0000 0000
N-0-0000	197568495616	16#2E 0000 0000
O-0-0000	201863462912	16#2F 0000 0000

Tab. 7-2: *Parameter offsets***7.3.3 AXIS_REF****Brief description**

The structure contains information on the desired control and on the object controlled by this control (axis).



Information is transferred via **VAR_IN_OUT** to the respective PLC function block defined in the "RMB_PLCopenFieldBus.library".



Only for field bus drives/axes connected to the field bus.

Library	Range
RMB_PLCopenFieldBus	Data types

Tab. 7-3: *Library assignment**Type definition*

```

(* Structure of PLCopen element AXIS_REF. The AXIS_REF is a structure that contains *)
(* information on the corresponding object. *)
(* It is used as a VAR_IN_OUT in all Fieldbus Motion Control Function Blocks. *)
TYPE AXIS_REF :
STRUCT
  CntrlNo: CONTROLS := LOCAL_CNTRL; (* control reference number *)
  AxisNo: OBJECTS := FieldBusAxis_1; (* Steuerungsnummer, Init: lokale Steuerung *)
  IndraDrive: MP_Drive; (* object reference number *)
  IndraDriveParameter: MP_DriveParameter; (* Achsreferenznummer, Init: FieldBusAxis_1 *)
END_STRUCT
END_TYPE

```

Functional description

The structure **AXIS_REF** (RIL_CommonTypes) contains address information on the desired control and on the object controlled by this control (axis number).

In contrast to the **AXIS_REF** structure (RIL_CommonTypes) supporting the addressing of Sercos-based IndraDrives, the **AXIS_REF** structure (RMB_PLCopenFieldbus) is extended by two additional elements (MP_Drive and MP_DriveParameter). These two elements are only used by the system.

The axis number can differ from the station number set at the drive.

The Global Variable List "MLCVarGlobal" with all Sercos-based axes is automatically created and maintained for Sercos-based axes.

The global variable list "GVL_FieldBusDrives" with all field bus drives has to be manually created and maintained for the field bus drives.

The two first elements of each drive "CntrlNo" and "AxisNo" are preassigned in this list. The elements "MP_Drive" and "MP_DriveParameter" remain unchanged. Therefore, they do not have to be included in the list.



Address information is transferred as described in the following:

Sercos-based axes

- Information was transferred up to 13VRS via **VAR_IN_OUT**.
- In 14VRS and higher, information is transferred via **VAR_IN_OUT CONSTANT**.

Field bus axes

- Information is always transferred via **VAR_IN_OUT**.

Global Variable List "GVL_FieldBusDrives", example

```
VAR GLOBAL
Drive_X: FieldBusDrives.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=FieldBusAxis_11); // IndraDrive X - example
Drive_Y: FieldBusDrives.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=FieldBusAxis_12); // IndraDrive Y - example
Drive_Z: FieldBusDrives.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=FieldBusAxis_13); // IndraDrive Z - example
Drive_n: FieldBusDrives.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=FieldBusAxis_14); // IndraDrive n - example
END_VAR
```

Namespace specification FieldBusDrives

The namespace specification "FieldBusDrives." for a field bus drive has to be in front.

Declaration AXIS_REF

```
Drive_A: FieldBusDrives.AXIS_REF: =
```

The data types are in the RMB_PLCOpenFieldBus.library, folder "Datatypes", element "AXIS_REF".

Assignment CntrlNo

```
(CntrlNo: = LOCAL_CNTRL
```

The enumeration value LOCAL_CNTRL is contained in RIL_CommonTypes.lib, folder "Data types of logical addresses", CONTROLS.

The field bus function blocks always refer to the local PLC, that is **LOCAL_CNTRL**.

Assignment AxisNo

```
AxisNo: = FieldBusAxis_1);
```

The enumeration value FieldBusAxis_1 is in the RIL_CommonTypes.lib, folder "Data types of logical addresses", OBJECTS.

To avoid ambiguities, there is an individual number range stored for field bus axes

FieldBusAxis_1 (6001) up to FieldBusAxis_99 (6099)

For field bus drives, specifications on the control and the axis number are used for the error output (ErrorIdent).

A functionality is not combined with the assignment of the axis number.



It is recommended to compare the station number and the axis number.

7.4 Function blocks for single-axis control (Motion_FBs)

7.4.1 Motion_FBs, general



The function blocks named in the following are used for the motion control.

State diagram

The following diagram defines the behavior of the MotionControl axes from the user's point of view as well as the reaction on exceptions in the programs.

In principle, motion commands are given and processed sequentially, even if the PLC is capable of parallel processing. These commands follow the state diagram of the MotionControl axes.

Each axis is in exactly one of the states defined (see the state diagram below). Each motion command represents a state transition for the axis concerned. The sequence of these transitions describes the entire behavior of the axis.

The chart focuses on the single axis.



The field bus drives only operate as single-axis function blocks!

The following function blocks with "management character" do not influence the state diagram:

- [MC_ReadActualPosition, page 415,](#)
- [MC_ReadAxisError, page 417,](#)
- [MC_ReadStatus, page 418,](#)

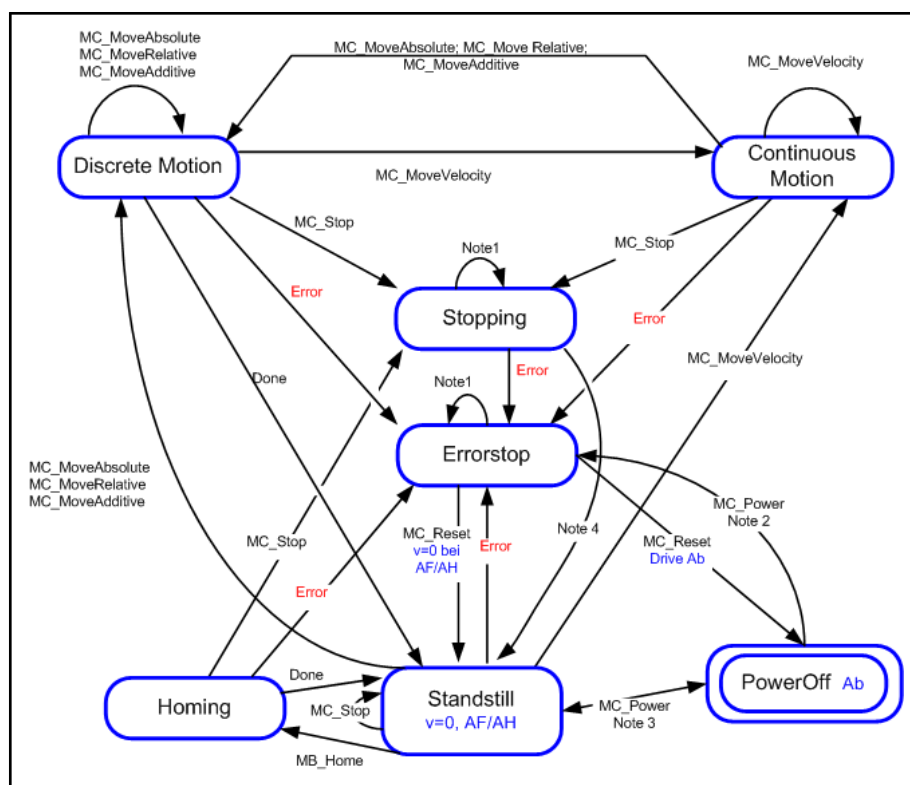


Fig. 7-1: MotionControl state diagram of an field bus axis

Legend (Note 1) - All function blocks can be called without being processed except for MC_Reset and Error. They create the respective transitions to StandStill or ErrorStop.

(Note 2) - Power.Enable = TRUE and error in the axis.

(Note 3) - Power.Enable = TRUE and error-free axis.

(Note 4) - MC_Stop.Done AND NOT MC_Stop.Execute

An error in the drive results in a power switch off after a certain period, depending on the programmed error reaction. However, the state machine remains in "ErrorStop" and when the power is switched off, "MC_Reset" goes into power-off state.

	Function
MC_Power, page 367	Controls the power adding (on or off).
MC_Reset, page 368,	Implements the transition from the "ErrorStop" state to the "StandStill" state by resetting all axis-internal errors.
MC_Stop, page 369	Implements a controlled motion stop and transfers the axis to the "Stopping" state. This stops any function block processing. When standstill is reached, the "Done" output is set. Another command can only be started after the axis reached the "Stopping" state.
MC_Stop, page 371,	Implements a controlled motion stop and transfers the axis to the "Stopping" state. This stops any function block processing. When standstill is reached, the "Done" output is set. When "ExecuteLock" is TRUE, the axis is in the "Stopping" state and cannot be moved by another command.

	Function
MB_Home, page 373	Controls axes with incremental encoder when executing the "search home" sequence. The details of the sequence must be set via axis parameters. "Homing" ends at standstill.
MC_MoveAbsolute, page 374	The function block copies the absolute predetermined "position" to the "target position" and approaches it.
MC_MoveAdditive, page 376	The function block adds the "distance" to the "target position" and approaches it.
MC_MoveRelative, page 379	The function block determines the new "target position" by adding the "distance" to the previous "actual position" and approaching it.
MC_MoveVelocity, page 382	Controls an "endless" motion at the respectively defined velocity. To end the motion, the function block instance has to be interrupted by another function block instance which issues a new command.

Tab. 7-4: Function blocks of the "Motion_FBs" folder

7.4.2 MC_Power

Brief description

The function block controls the power adding (on or off).

For **each real axis, a separate instance** of this function block is required.

Enabling the function block is a prerequisite for every motion, acceleration or deceleration.

A special situation is given for the function block [MC_Stop, page 369](#). The position is kept active by the drive. Thus, the MC_Power function block cannot be switched off when MC_Stop is active.



Together with a virtual axis, the function block results in an error message.

Library	Range
RMB_PLCopenFieldBus	Motion_FBs

Tab. 7-5: Library assignment

Interface

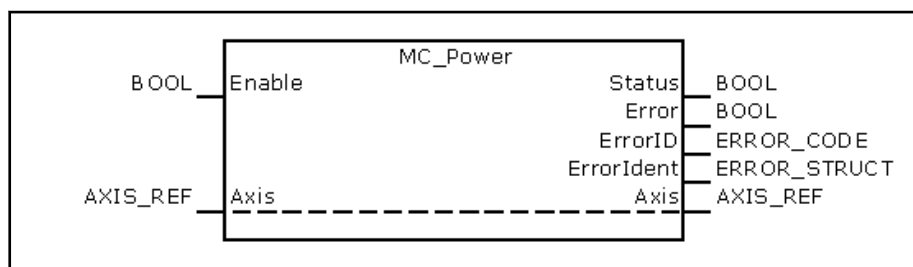


Fig. 7-2: MC_Power function block

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
VAR_INPUT	Enable	BOOL	When "Enable" is TRUE, power is added
VAR_OUTPUT	Status	BOOL	Actual state of power adding
	Error	BOOL	Indicates that an error occurred in the function block instance

	Name	Type	Comment
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 7-6: Interface of the MC_Power function block

- Error handling:**
- An error message is issued if the "Status" is still FALSE after a timeout at the set "Enable". This indicates that a hardware problem occurred when power was added.
 - An error message is also issued if it is tried to disconnect the power in an axis state in which this is not permitted.
 - If adding on power fails or malfunctions while the axis is moving, an "ErrorStop" is generated for the axis.

In case of a configuration error, the following is displayed

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.4.3 MC_Reset

Brief description This function block implements the transition from the 'ErrorStop' state to the 'StandStill' state by resetting all axis-internal errors.



The outputs of the function block instances that indicated the error(s) are not affected.

If there is no error present, triggering MC_Reset has no effect.

Library	Range
RMB_PLCOpenFieldBus	Motion_FBs

Tab. 7-7: Library assignment

Interface description

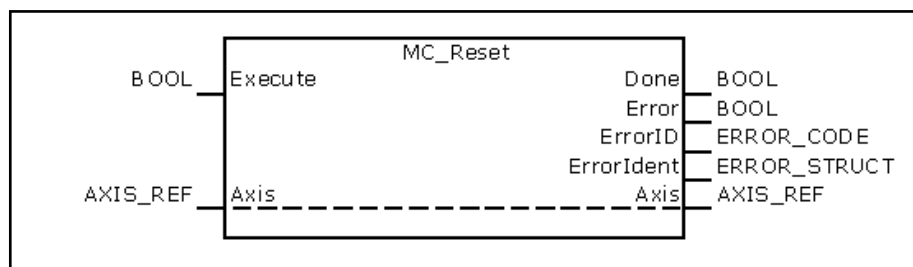


Fig. 7-3: MC_Reset function block

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
VAR_OUTPUT	Done	BOOL	Errors are reset
	Error	BOOL	Indicates that an error occurred in the function block instance

	Name	Type	Comment
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 7-8: Interface of the MC_Reset function block

Signal-time diagram

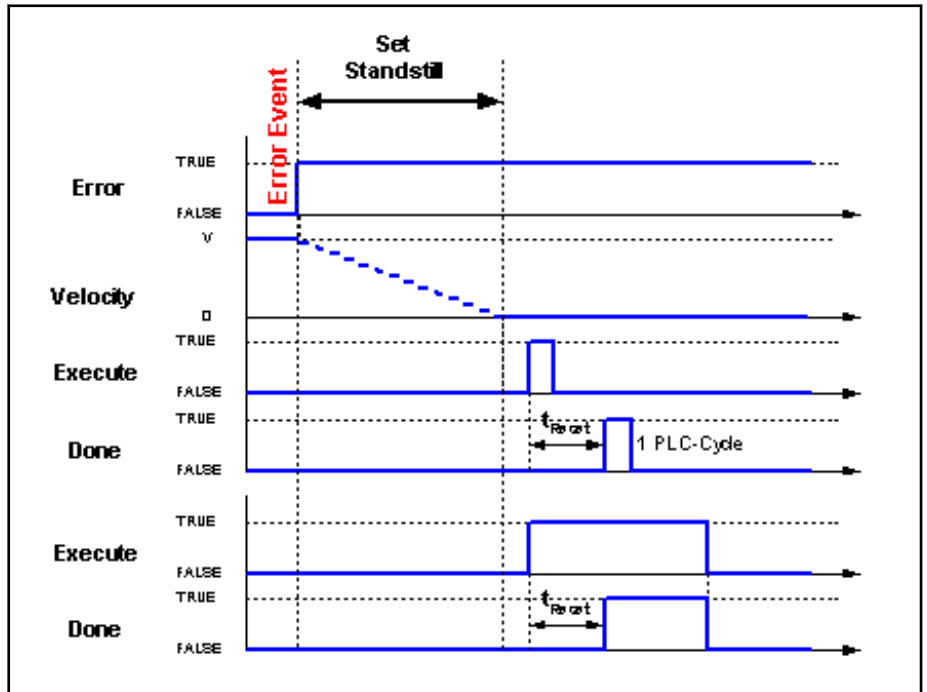


Fig. 7-4: Time diagram of MC_Reset

The function block uses the parameter "S-0-0099", C0500 Reset state class 1.

If there is no error present, triggering MC_Reset has no effect.

A clear error command "C5" is issued in any case at the drive.

Troubleshooting

The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.4.4 MC_Stop

Brief description

This function block implements a controlled motion stop and transfers the axis to the "Stopping" state. As a result, any processing function block is stopped. When standstill is reached, the "Done" output is set.

The position is kept active.



When "Execute" is TRUE, the axis remains in "Stopping" state.

Another command can only be started after the axis reached the "Stopping" state.

In the "Stopping" state, it is not possible to switch off the axis power ([MC_Power](#), [page 367](#),). If "Enable" of the MC_Power instance is disabled, its "status" nevertheless remains switched on until "Execute" of the MC_Stop instance is also disabled.



MC_Stop is not supported for an IndraDrive with active operating stop (P-0-0558 Bit 0 = 1).

The function block never sets the "Done" output to TRUE.

Library	Range
RMB_PLCoenFieldBus	Motion_FBs

Tab. 7-9: Library assignment

Interface description

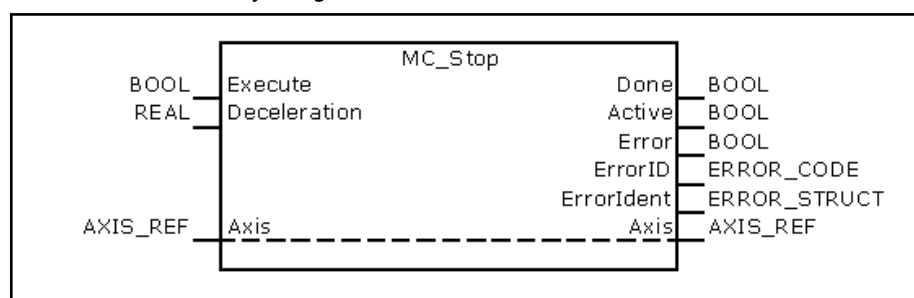


Fig. 7-5: MC_Stop function block

	Name	Type	Comment ¹⁾
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge. The function block cannot be aborted until the "Stopping" state is reached.
	Deceleration	REAL	Deceleration (always greater than 0, otherwise no "Stopping").
VAR_OUTPUT	Done	BOOL	Standstill reached
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 7-10: Interface of the MC_Stop function block

¹⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

Signal-time diagram

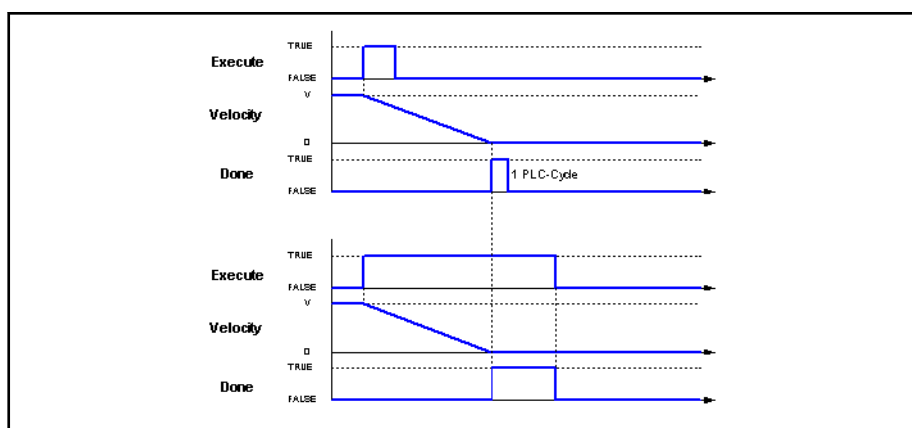


Fig. 7-6: Time diagram of MC_Stop

Troubleshooting

The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.4.5 MB_Stop

Brief description

This function block implements a controlled motion stop and transfers the axis to the 'Stopping' state. As a result, any processing function block is stopped. When standstill is reached, the "Done" output is set.

The position is kept active.



When "ExecuteLock" is TRUE, the axis is in the "Stopping" state and cannot be moved by another command.

In contrast to "MC_Stop, page 369," the function block can be interrupted by another function block as soon as "ExecuteLock" = FALSE.

In the "Stopping" state, it is not possible to switch off the axis power (MC_Power, page 367,). If "Enable" of the MC_Power instance is deactivated, its "Status" nevertheless remains switched on until the "ExecuteLock" signal of the MB_Stop instance is also deactivated.

Library	Range
RMB_PLCOpenFieldBus	Motion_PLCOpen

Tab. 7-11: Library assignment

Interface description

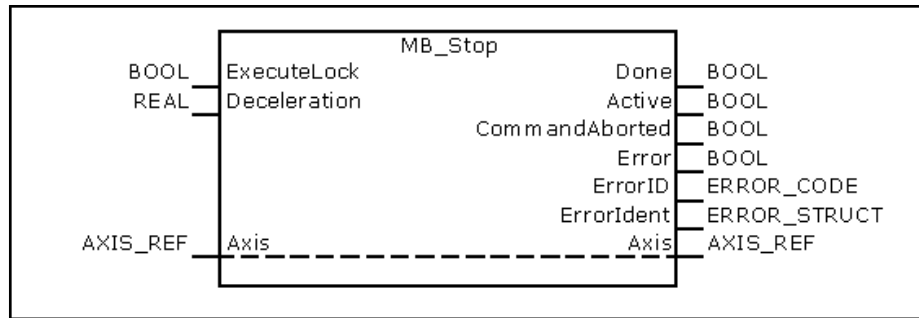


Fig. 7-7: MC_Stop FB

	Name	Type	Comment ²⁾
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
VAR_INPUT	ExecuteLock	BOOL	It starts the stopping of the axis with the rising edge. As long as "ExecuteLock" = TRUE, a motion cannot be started.
	Deceleration	REAL	Deceleration (always greater than 0, otherwise no "Stopping").
VAR_OUTPUT	Done	BOOL	Standstill is reached.
	Active	BOOL	Command is enabled. Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command is aborted by another command.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 7-12: Interface of the MB_Stop FB

Signal-time diagram

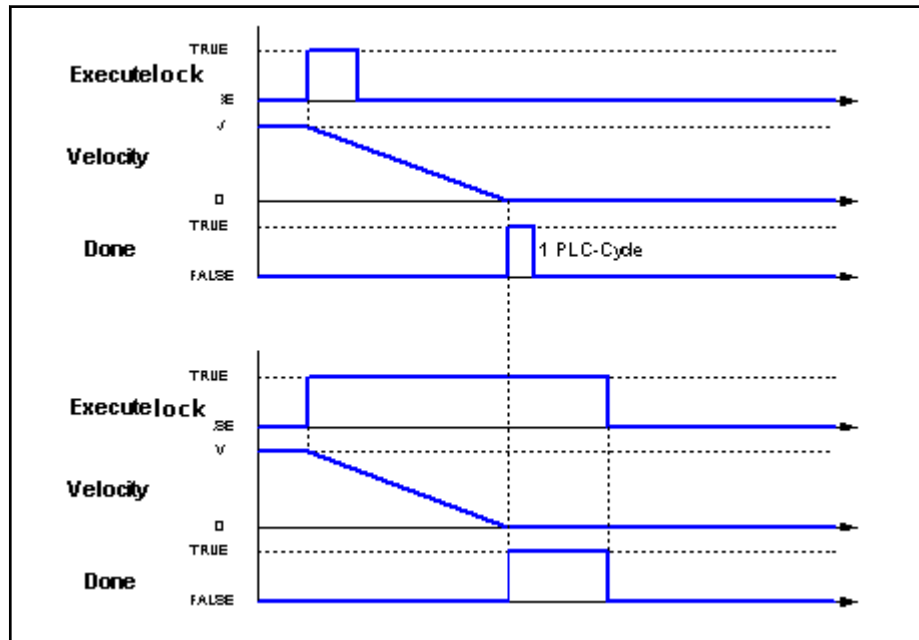


Fig. 7-8: Time diagram of MB_Stop

²⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

Troubleshooting The function block generates error messages only at a positive edge or if TRUE is set at the "ExecuteLock" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.4.6 MB_Home

Brief description This function block controls **axes with incremental encoder** when executing the "search home" sequence. The details of the sequence must be set via axis parameters. "Homing" ends at standstill.

Library	Range
RMB_PLCopnFieldBus	Motion_FBs

Tab. 7-13: Library assignment

Interface description

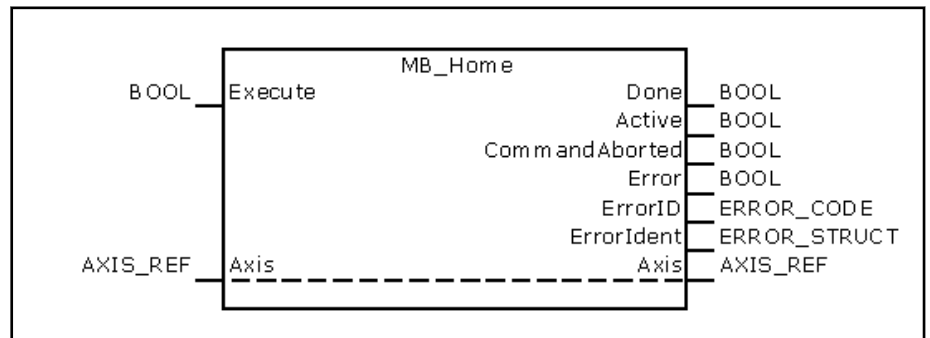


Fig. 7-9: MB_Home function block

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
VAR_OUTPUT	Done	BOOL	Homing completed, velocity 0
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 7-14: Interface of the MB_Home function block

Signal-time diagram

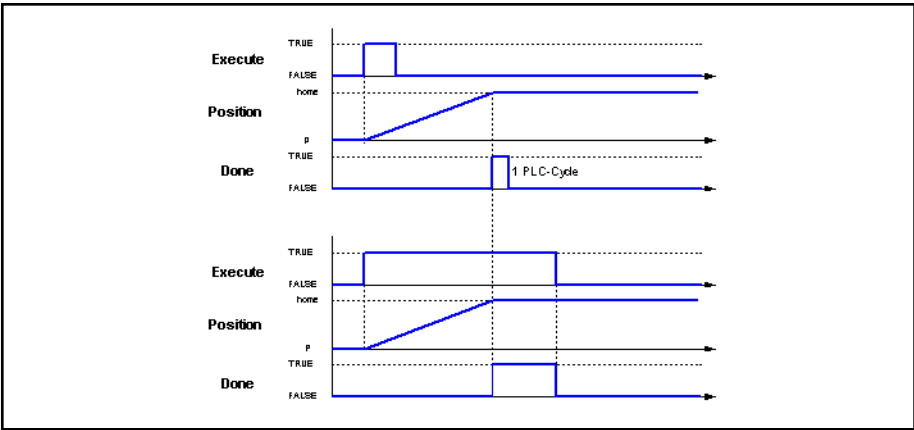


Fig. 7-10: Time diagram of MB_Home

Troubleshooting The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.4.7 MC_MoveAbsolute

Brief description The function block copies the absolute predetermined "position" to the "target position" and approaches it.

If this function block requires power added before via an instance of the [MC_Power](#), [page 367](#), (for real axes).

Library	Range
RMB_PLCOpenFieldBus	Motion_FBs

Tab. 7-15: Library assignment

Interface description

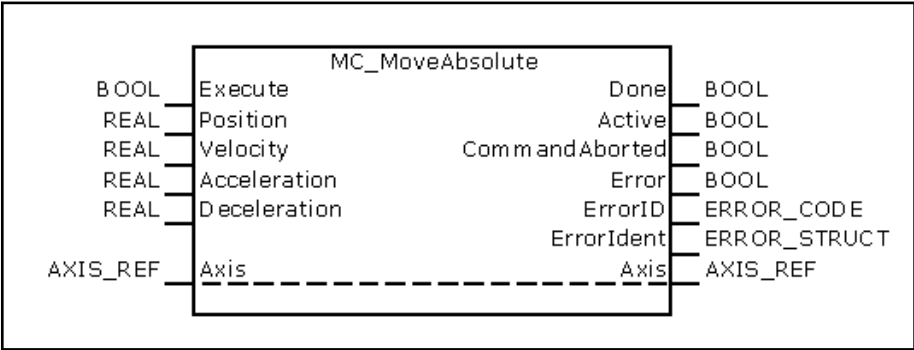


Fig. 7-11: MC_MoveAbsolute function block

	Name	Type	Comment ³⁾
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
	Position	REAL	Target position for the motion (+/-)
	Velocity	REAL	Maximum velocity value (always +) (does not necessarily have to be reached).
	Acceleration	REAL	Acceleration (always +).
	Deceleration	REAL	Deceleration (always +).
VAR_OUTPUT	Done	BOOL	Desired position reached
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 7-16: Interface of the MC_MoveAbsolute function block

Implementation example: The following figure following shows two examples of cooperation of two "MC_MoveAbsolute" function block instances:

1. The left side of the time diagram shows the case in which "Instance2" is called after "Instance1". After "Instance1" has been implemented the desired position 6,000 (velocity 0), its "Done" output enables "Instance2" with the target position 10,000.
2. The right side of the time diagram shows how "Instance2" is already enabled while "Instance1" is still running. The motion of "Instance1" is stopped and aborted by the "Test" signal. "Instance2" directly approaches its target position 10,000, although the position 6,000 has not yet been reached.

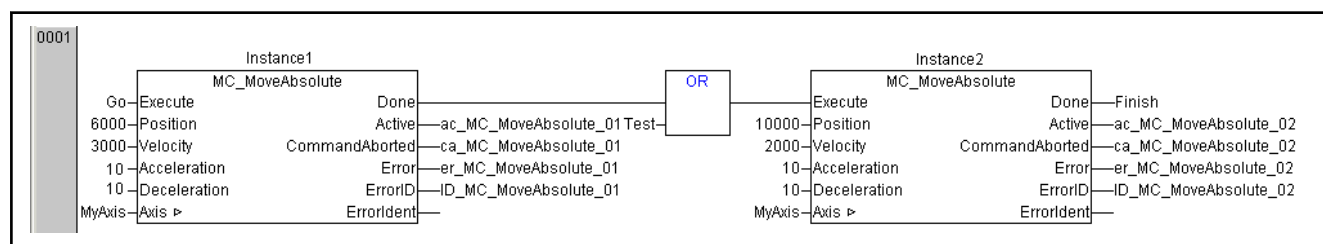


Fig. 7-12: Two instances of MC_MoveAbsolute

³⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

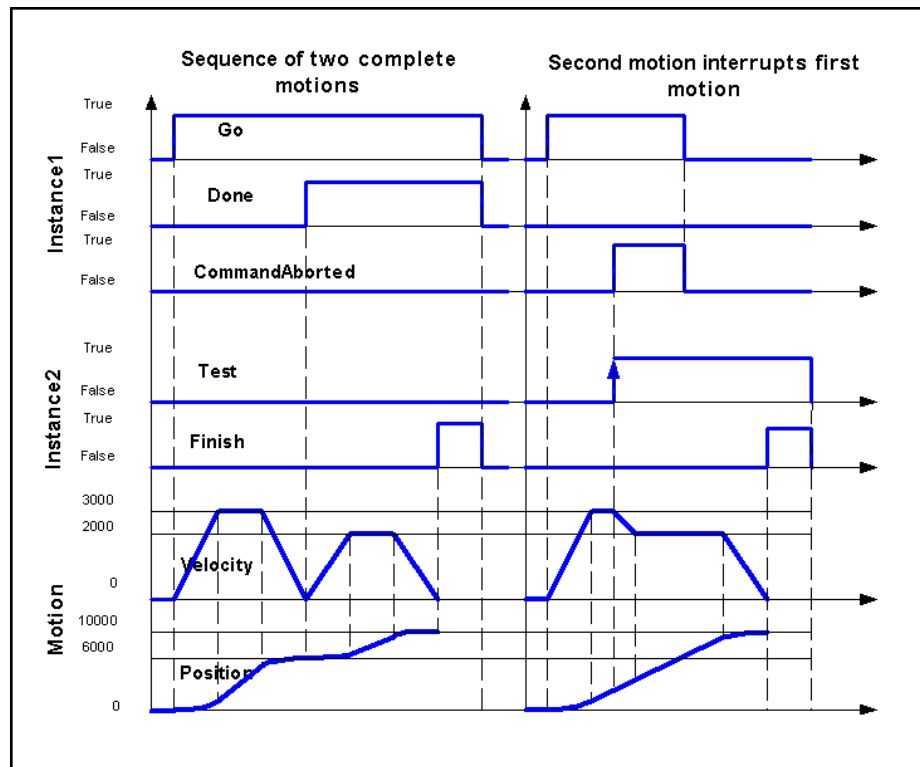


Fig. 7-13: Time diagram: Two instances of MC_MoveAbsolute

- Parameters used** The function block determines the new target position by applying the command position and approaching it.
- Special function:** The function blocks "MC_MoveAbsolute", "MC_MoveAdditive" and "MC_MoveRelative" accept a velocity specification of $V_{com} = 0$.
When these function blocks ($V_{com} = 0$) are called, the axis is stopped. Positioning is only interrupted. The target position remains in the axis.
If MC_MoveAdditive ($V_{com} \neq 0$) is called subsequently, the remaining distance is traversed. The target position is only overwritten when another Motion function block is called.
- Troubleshooting** The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

Program:

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.4.8 MC_MoveAdditive

- Brief description** The function block adds the "distance" to the "target position" and approaches it.



If this function block requires power added before via an instance of the [MC_Power](#), page 367, (for real axes).

Library	Range
RMB_PLCopenFieldBus	Motion_FBs

Tab. 7-17: Library assignment

Interface description

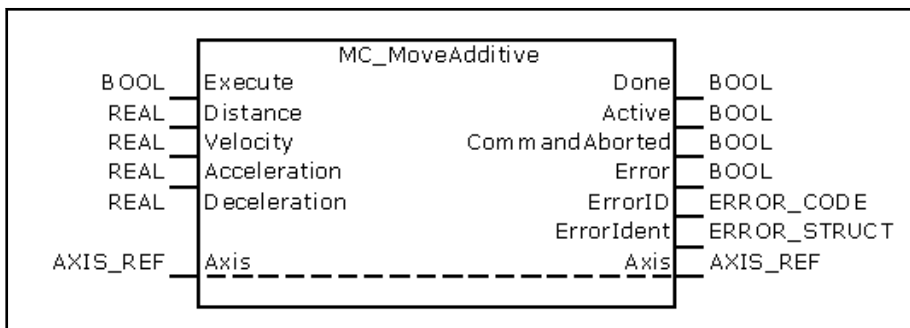


Fig. 7-14: MC_MoveAdditive function block

	Name	Type	Comment ⁴⁾
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
	Distance	REAL	Relative distance for the motion (+/-).
	Velocity	REAL	Maximum velocity value (always +) (does not necessarily have to be reached).
	Acceleration	REAL	Acceleration (always +).
	Deceleration	REAL	Deceleration (always +).
VAR_OUTPUT	Done	BOOL	Desired position reached
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 7-18: Interface of the MC_MoveAdditive function block

Implementation example:

The figure below shows two examples of two combined function block instances while the axis is in "Discrete Motion State".



Two different function block types are used:

- Instance1: MC_MoveAbsolute
- Instance2: MC_MoveAdditive.

1. The left part of the time diagram shows a case in which the second function block is started after the first one. The first function block reaches the distance 6,000 (at a velocity of 0). The "Done" output causes the second function block to travel a distance of up to 10,000.

⁴⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

2. The right part of the time diagram shows a case in which the second function block is already started while the first function block is still active. In this case, the first motion is stopped and aborted at constant velocity by the "Test" signal. The second function block adds its distance of 4,000 to the originally commanded position of 6,000 and moves the axis to the resulting position 10,000 at a velocity specified at the second function block.

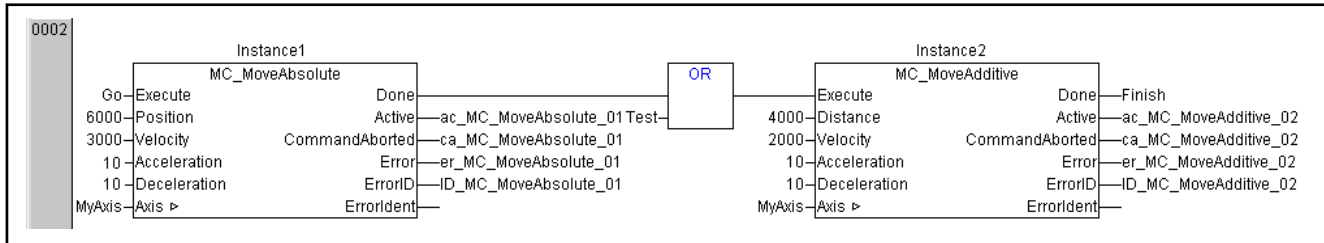


Fig. 7-15: Instances of MC_MoveAbsolute and MC_MoveAdditive

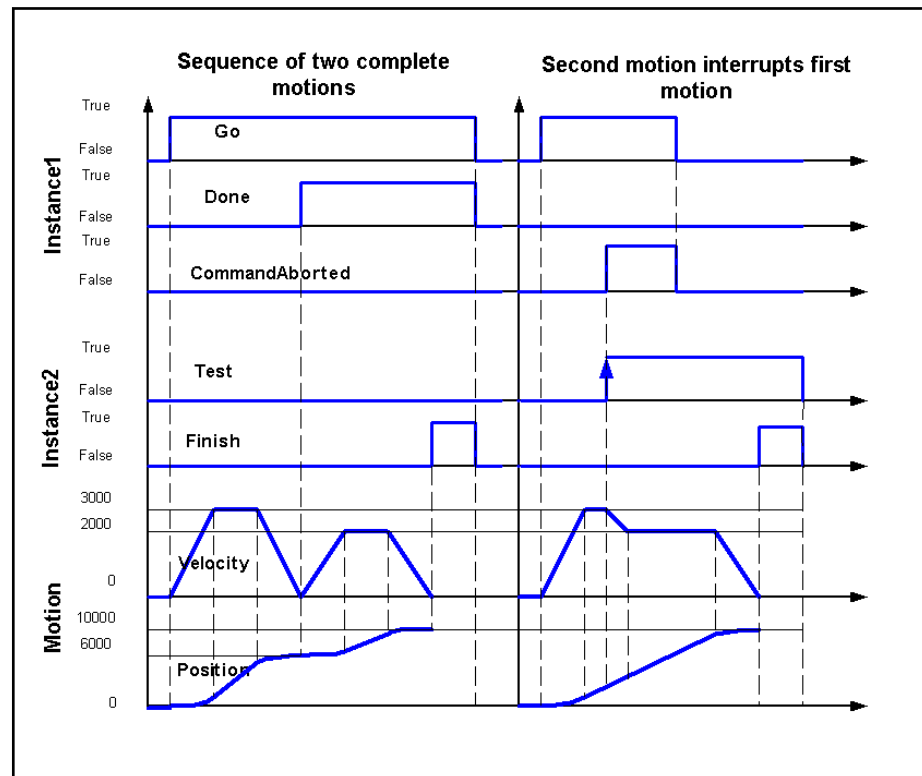


Fig. 7-16: Time diagram: Instances of MC_MoveAbsolute and MC_MoveAdditive

Parameters used The function block determines the new target position by adding the distance to the previous target position and approaching it.



The distance is specified relative to the actual position.

Special function: The function blocks "MC_MoveAbsolute", "MC_MoveAdditive" and "MC_MoveRelative" accept a velocity specification of $V_{com} = 0$.

When these function blocks ($V_{com} = 0$) are called, the axis is stopped. Positioning is only interrupted. The target position remains in the axis.

If MC_MoveAdditive ($V_{com} \neq 0$) is called subsequently, the remaining distance is traversed. The target position is only overwritten when another Motion function block is called.

Troubleshooting The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

Program:

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F RELATED TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.4.9 MC_MoveRelative

Brief description The function block determines the new "target position" by adding the "distance" to the previous "actual position" and approaching it.



If this function block is used, power must have been added before via an instance of the [MC_Power](#), page 367, (for each real axes).

Library	Range
RMB_PLCopenFieldBus	Motion_FBs

Tab. 7-19: Library assignment

Interface description

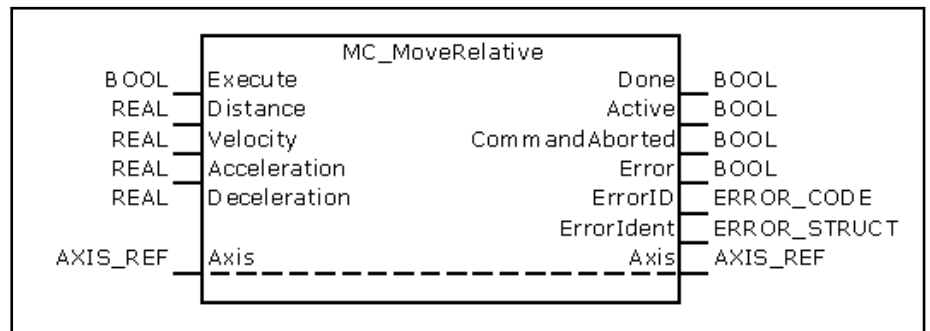


Fig. 7-17: MC_MoveRelative function block

	Name	Type	Comment ⁵⁾
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
	Distance	REAL	Relative distance for the motion (+/-).
	Velocity	REAL	Maximum velocity value (always +) (does not necessarily have to be reached).
	Acceleration	REAL	Acceleration (always +).
	Deceleration	REAL	Deceleration (always +).
VAR_OUTPUT	Done	BOOL	Desired position reached
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command

⁵⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

	Name	Type	Comment ⁵⁾
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 7-20: Interface of the MC_MoveRelative function block

Signal-time diagram

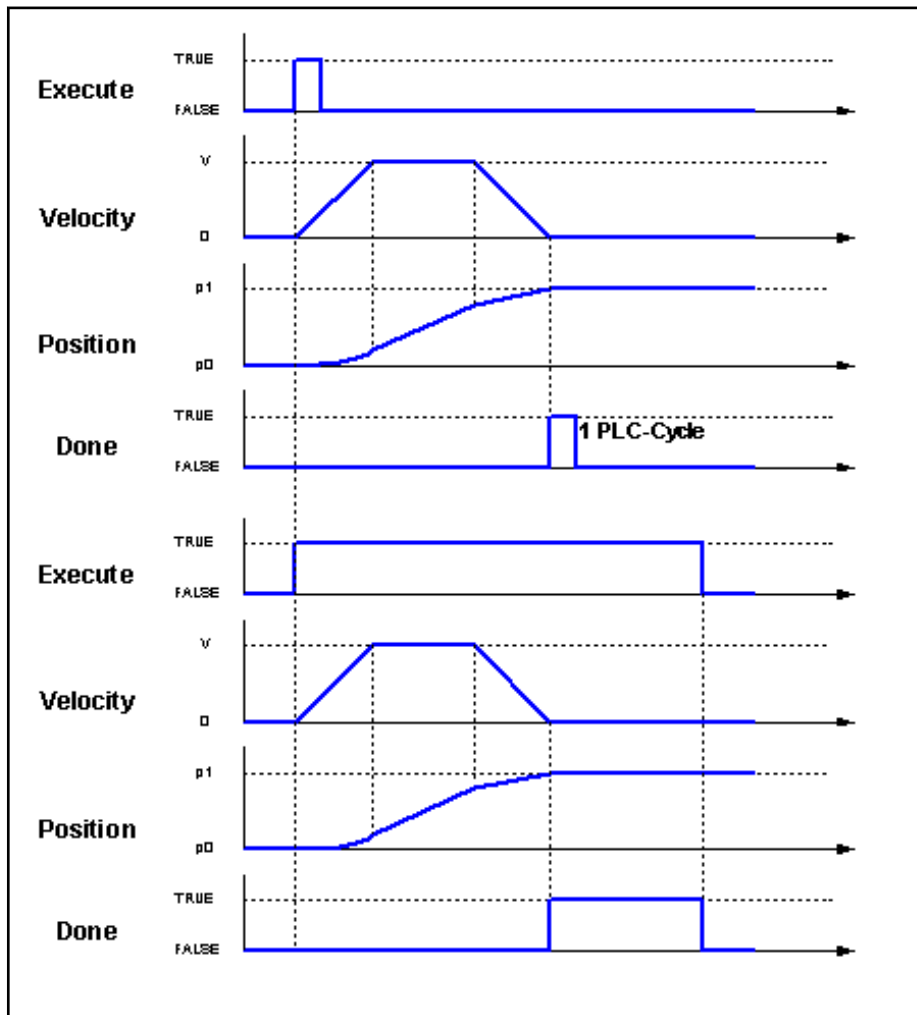


Fig. 7-18: Time diagrams for MC_MoveRelative

Implementation Example

The following example shows the combination of two MC_MoveRelative function block instances.

1. The left part of the time diagram shows a case in which "Instance2" is called after "Instance1" was processed. After "Instance1" has reached the desired distance 6,000 (velocity 0), its "Done" output uses "Instance2" to cause a further motion by 4,000 to reach the total distance of 10,000.
2. The right part shows how "Instance2" is already enabled by the "Test" signal while "Instance1" is still running. The first motion is aborted during the constant velocity of "Instance1". "Instance1" adds its distance of

⁵⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

4,000 to the current position (3,250). The axis approaches the resulting position of 7,250.

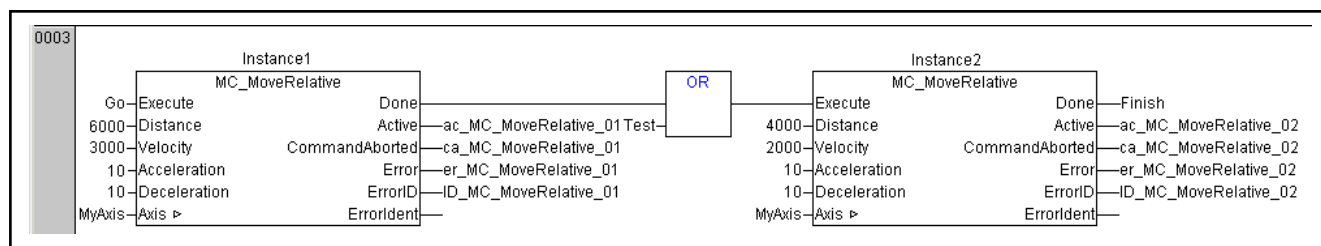


Fig. 7-19: Two instances of MC_MoveRelative

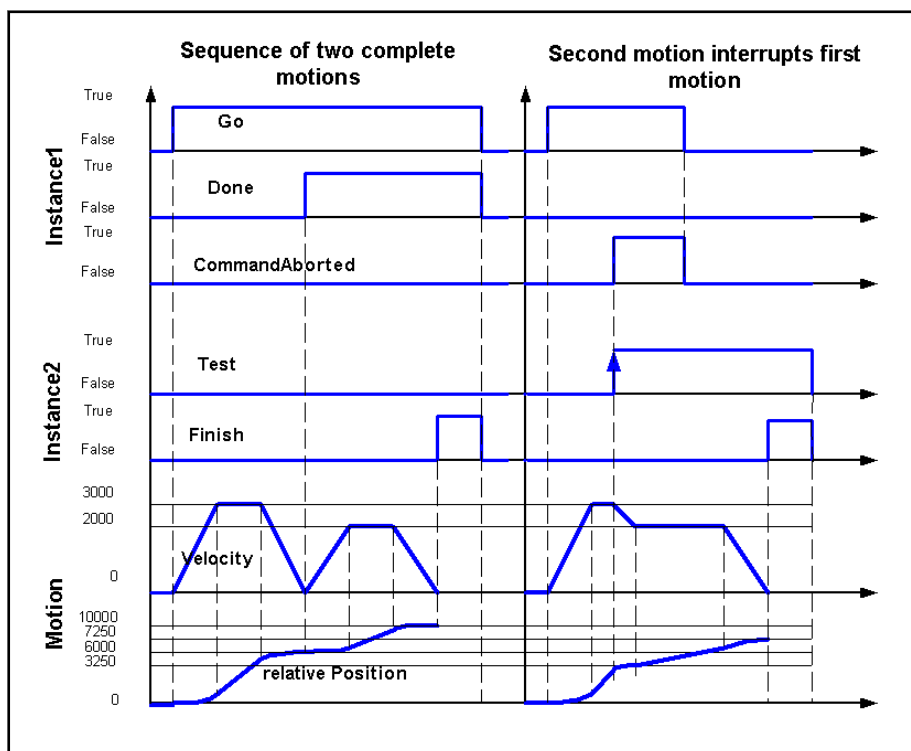


Fig. 7-20: Time diagram: Two instances of MC_MoveRelative

Parameters used The function block determines the new "target position" by adding the "distance" to the previous "actual position" and approaching it.



The distance is specified relative to the actual position.

Special function: The function blocks "MC_MoveAbsolute", "MC_MoveAdditive" and "MC_MoveRelative" accept a velocity specification of Vcom = 0.

When these function blocks (Vcom = 0) are called, the axis is stopped. Positioning is only interrupted. The target position remains in the axis.

If MC_MoveAdditive (Vcom ≠ 0) is called subsequently, the remaining distance is traversed. The target position is only overwritten when another Motion function block is called.

Troubleshooting The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

Program:

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
```

```
ErrorIdent.Table := F RELATED TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.4.10 MC_MoveVelocity

Brief description This function block controls an "endless" motion at the respectively defined velocity. To end the motion, the function block instance has to be interrupted by another function block instance which issues a new command.



If this function block is used, power must have been added before via an instance of the [MC_Power](#), [page 367](#), (for each real axes).

The "InVelocity" signal is reset if the function block instance is interrupted by another function block instance (CommandAborted) or if the edge of its "Execute" input is falling.



If "Execute" is disabled, the drive keeps running on the same velocity.

Library	Range
RMB_PLCopOpenFieldBus	Motion_FBs

Tab. 7-21: Library assignment

Interface description

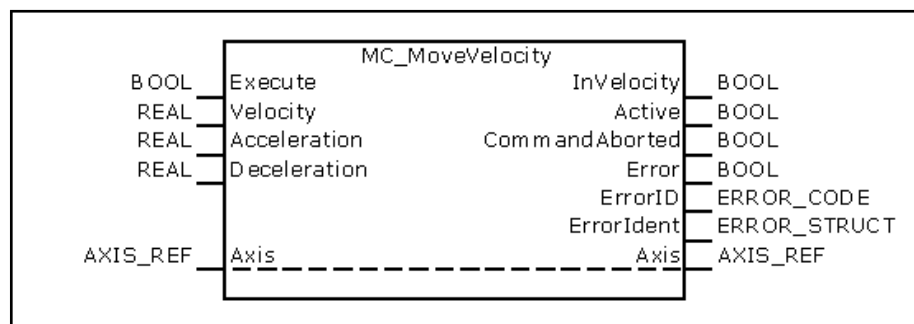


Fig. 7-21: MC_MoveVelocity function block

	Name	Type	Comment ⁶⁾
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
	Velocity	REAL	Maximum velocity value (does not necessarily have to be reached).
	Acceleration	REAL	Acceleration (always +).
	Deceleration	REAL	Deceleration (always +).
VAR_OUTPUT	InVelocity	BOOL	Velocity reached (for the first time)
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command

⁶⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

	Name	Type	Comment ⁶⁾
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 7-22: Interface of the MC_MoveVelocity function block

Signal-time diagram

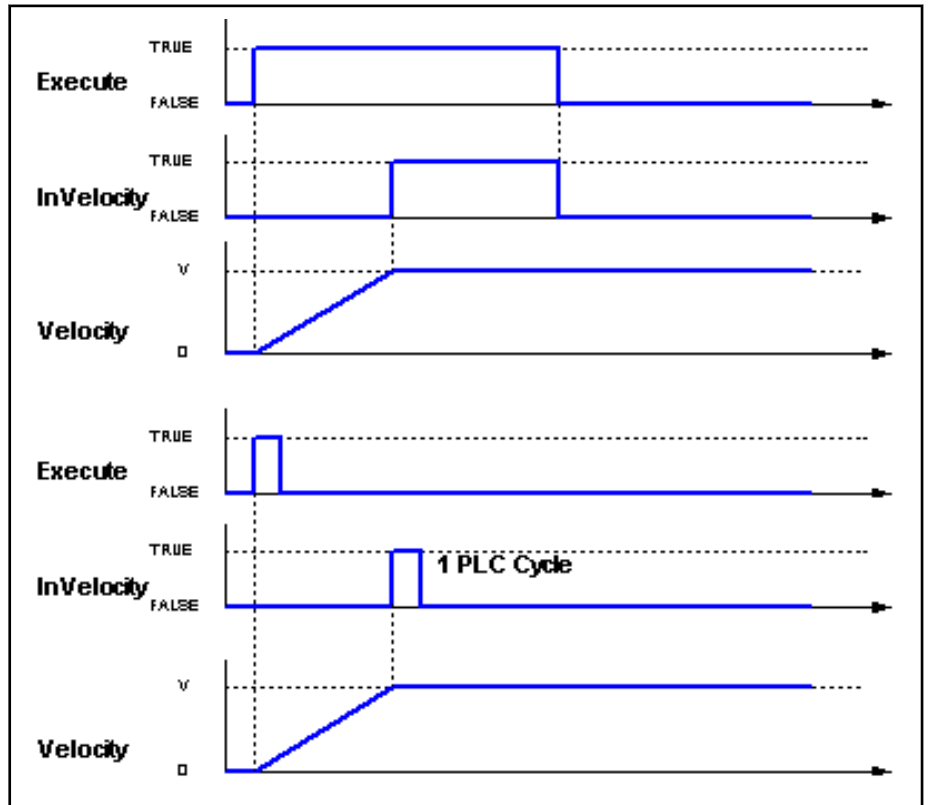


Fig. 7-22: Time diagram of MC_MoveVelocity (without aborted command)

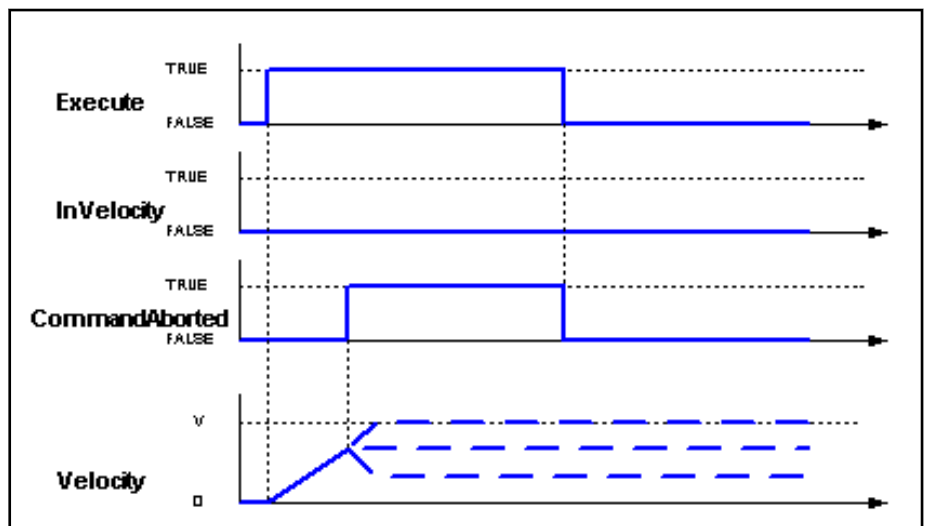


Fig. 7-23: Time diagram of MC_MoveVelocity (with aborted command)

⁶⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

Implementation example: The following figure shows two examples of two combined instances of the MC_MoveVelocity function block:

1. The left part of the time diagram shows a case in which the second instance is called after the job of the first instance is complete. Once the first function block has reached the required velocity of 3,000, the "Instance1.InVelocity" output **AND** the "Next" signal cause the second function block to continue the motion at a velocity of 2,000.
2. The right side of the time diagram shows a case where the first function block has not yet reached the velocity when the second function block is started. The following is shown: The first motion is restarted by "Go" at the "Instance1.Execute" input. While the first function block is still accelerating to reach a velocity of 3,000, it is interrupted by the "Test" signal which starts the second function block. The second function block is now running and decelerates to a velocity of 2,000.

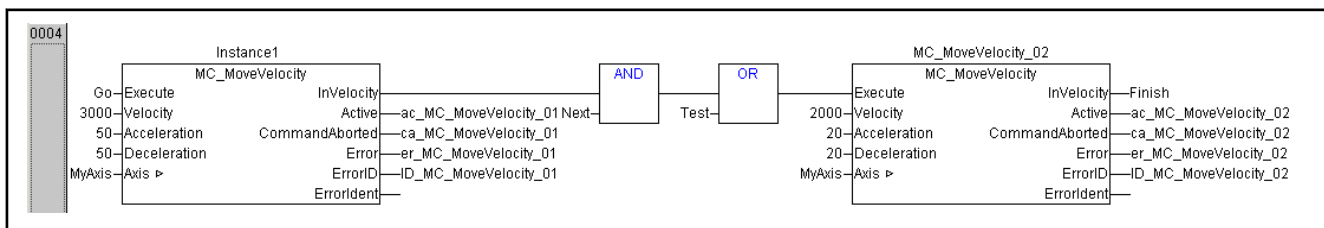


Fig. 7-24: Two instances of MC_MoveVelocity

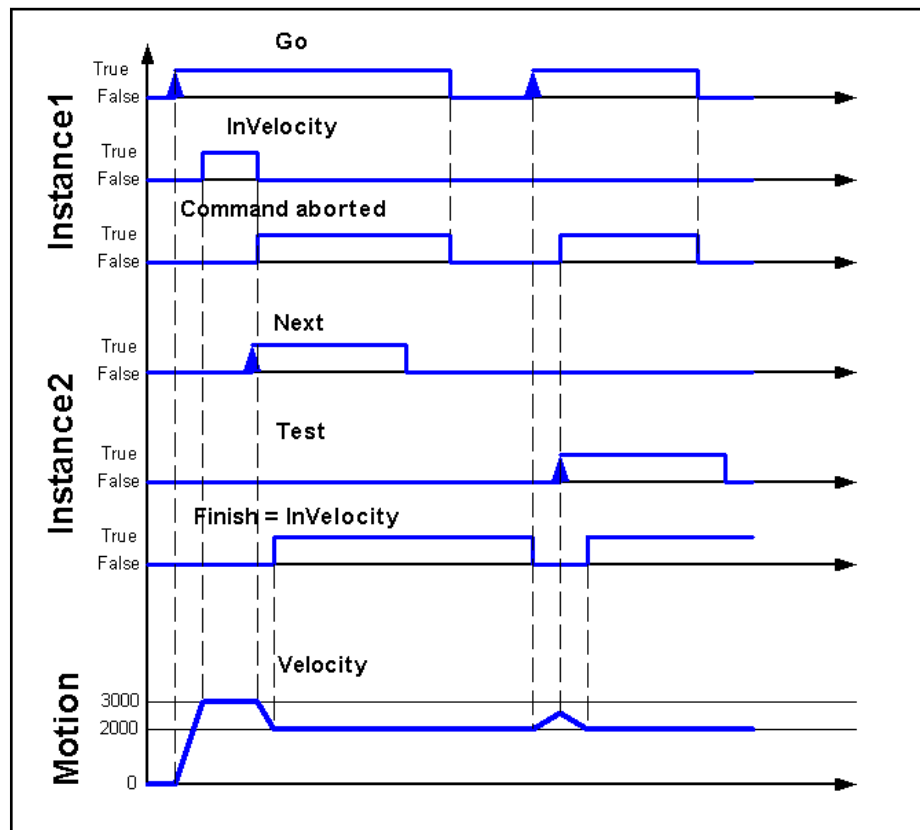


Fig. 7-25: Time diagram: Two instances of MC_MoveVelocity

Troubleshooting The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

Program:

```

Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F RELATED TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;

```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.5 Function blocks for parameter access (Parameter_FBs)

7.5.1 Parameter_FBs, general information

The RMB_PLCopenFieldBus.library for IndraDrives uses the parameter channel of cyclic communication (in cyclic I/O exchange) for the MB_Read/WriteXXX function blocks.

The parameterization of the drive via the field bus has implemented a configurable parameter channel in the cyclic channel of the IndraDrive.

When controlling the drive by function blocks of the "RMB_PLCopenFieldBus.library" library, the parameter channel has to be configured to "ParamCH IDN" in the drive node:

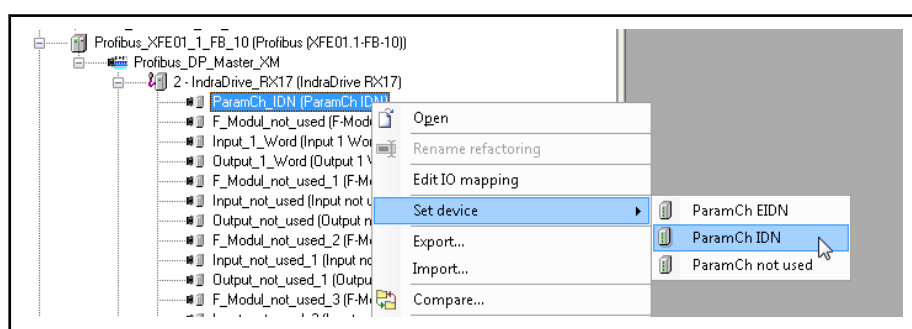


Fig. 7-26: Configuring the parameter channel

The parameter channel is always at the start of the cyclic data channel. The parameter channel length is configured in the master of the control

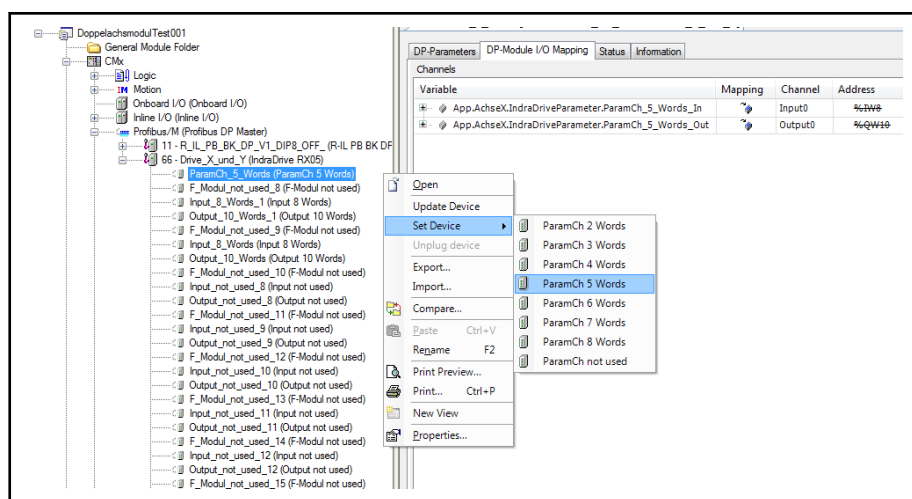


Fig. 7-27: Parameter channel in the cyclic data channel

	Function
MB_ReadListParameter, page 389,	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.
MB_ReadListSegParameterPtr, page 392,	Returns the values of a section 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.
MB_ReadMaxRealValue, page 394	The function block returns the maximum value allowed of a parameter specified by the user. The return value is represented as REAL number.
MB_ReadMaxValue, page 396	The function block returns the maximum value allowed of a parameter specified by the user. The return value is represented as DINT number.
MB_ReadMinRealValue, page 397	The function block returns the minimum value allowed of a parameter specified by the user. The return value is represented as REAL number.
MB_ReadMinValue, page 399	The function block returns the minimum value allowed of a parameter specified by the user. The return value is represented as DINT number.
MB_ReadRealParameter, page 402	Returns the value of a user-specific parameter. The return value is represented as REAL number.
MB_ReadParameter, page 400	Returns the value of a user-specific parameter. The return value is represented as DINT number.
MB_ReadSercosAttribute, page 403	Allows reading the Sercos attributes of a parameter. The return value is represented as a DWORD.
MB_ReadStringParameter, page 405,	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.
MB_WriteListParameter, page 406	Writes memory content of a defined length (NoOfBytes) provided from the initial address (ValueAdr) to a user-specific parameter.
MB_WriteListSegParameterPtr, page 409	Writes memory content of a defined length (NoOfBytes) provided from the initial address (ValueAdr) to a specified section of a user-specific parameter.
MB_WriteParameter, page 411	Writes a DINT value in a parameter specified by the user.
MB_WriteRealParameter, page 413	Writes a REAL value to a user-specific parameter.
MB_WriteStringParameter, page 414	Writes a character string with up to 255 characters to a user-specific parameter.
MC_ReadActualPosition, page 415	Reads the current position of the drive.
MC_ReadAxisError, page 417,	Retrieves the error state of the addressed axis. One instance of the function block is to be used for each axis.
MC_ReadStatus, page 418	Reads the current status of the axis and outputs it in a decoded way.

Tab. 7-23: Function blocks of the "Parameter_FB" folder

7.5.2 Access to data of a drive (IndraDrive)

Structure of and Access to Parameters from the PLC Program

The drive data can be accessed via read-only or sometimes write parameter access.

The S- and P-parameters can be accessed via the field bus.

The structure of all MotionControl parameters is identical:

S-, P- as parameter type identifier followed by the group number and a four digit parameter number.

P- and S-parameter declaration files for global constants are available for the user in this library.

Parameter constants

- [GVL_S_Param_ID, field bus, page 361,](#)
- [GVL_P_Param_ID, field bus, page 362,](#)

Parameter number: *S-parameters*

(*Constants to address parameters via parameter ident*)

```
VAR GLOBAL CONSTANT
  FP_S_0_0000 : MB_IDN := 0; (* Dummy parameter *)
  FP_S_0_0001 : MB_IDN := 1; (* NC cycle time TNcyc *)
  FP_S_0_0002 : MB_IDN := 2; (* SERCOS cycle time TScyc *)
  FP_S_0_0003 : MB_IDN := 3; (* Minimum AT transmit starting...
  (*.....*)
END_VAR
```

P-parameters

(*Constants to address parameters via parameter ident*)

```
VAR GLOBAL CONSTANT
  FP_P_0_0001 : MB_IDN := 32769; (* Switching frequency...
  FP_P_0_0004 : MB_IDN := 32772; (* Speed loop smoothing...
  FP_P_0_0007 : MB_IDN := 32775; (* Display text of diagnostic...
  FP_P_0_0008 : MB_IDN := 32776; (* Activation EStop function *)
  (*.....*)
END_VAR
```

As an alternative to the constant, the number can also be applied at the input "ParameterNumber" of the respective function block.

Parameter offsets:

Parameter group	Decimal	Hexadecimal
S-0-0000	0	16#0
P-0-0000	32768	16#8000

Tab. 7-24: *Parameter offsets*

7.5.3 Special Features of Twin or Multi-Axis Modules

Twin or multi-axis modules and CDD link axes

The parameter channel is only available for the first axis in case of twin axis or multi-axis modules and CDD link axes.

The slave axes can be selected via a subindex of the first axis.

In the "AXIS_REF" data structure of the RMB_PLCOpenFieldBus library, the user can select this subindex:

Declaration of the multi-axis module:

```
<AxisName>: <LibraryNamespace>.AXIS_REF:= ...;
```

```
Drive_X: FieldBusDrives.AXIS_REF:= ...;
```

Specifying the axis subindex of a multi-axis module or axis in the CCD link:

Drive_X.IndraDriveParameter.Subindex := Axis_Index;

Access to different axes of a multi-axis module

```

PROGRAM prParameterLesenST
VAR
    Drive_X:      FieldBusDrives.AXIS_REF:=
                    (AxisNo:= OBJECTS.FieldBusAxis 11,
                     CntrlNo:= CONTROLS.LOCAL_CNTRL);
    ReadParameter: FieldBusDrives.MB_ReadRealParameter;
    bReadEnable:   BOOL; // Read parameter request
    bSlaveAxis:    BOOL; // Reading/writing of/to slave axis
    rValue_X:      REAL; // Read X-axis value
    rValue_Y:      REAL; // Read value of Y-axis which is slave axis of X-axis
    bReadDone:     BOOL; // Function block reports reading successful
    rTmpValue:     REAL; // Read value
END_VAR

//bSlaveAxis = TRUE -> Reading of second axis module of a multiple axis module
IF NOT bSlaveAxis THEN
    // Subindex of first axis module of a multiple axis module
    Drive_X.IndraDriveParameter.Subindex := 0;
ELSE
    // Subindex of second axis module of a multiple axis module
    Drive_X.IndraDriveParameter.Subindex := 1;
END_IF
ReadParameter(
    Enable:= bReadEnable,
    ParameterNumber:= FieldBusDrives.FP_S_0_0051,
    // In case of multiple axis modules, always reference (AXIS_REF) of first axis
    Axis:= Drive_X,
    Done=> bReadDone,
    ...
    Value=> rTmpValue);
// Analyzing the parameter content depending on the subindex
IF bReadDone
    // First axis module of the multiple axis module
    AND Drive_X.IndraDriveParameter.Subindex = 0
    THEN
        rValue_X := rTmpValue;
    ELSIF bReadDone
        // Second axis module of the multiple axis module
        AND Drive_X.IndraDriveParameter.Subindex = 1
        THEN
            rValue_Y := rTmpValue;
        END_IF
    END_IF

```

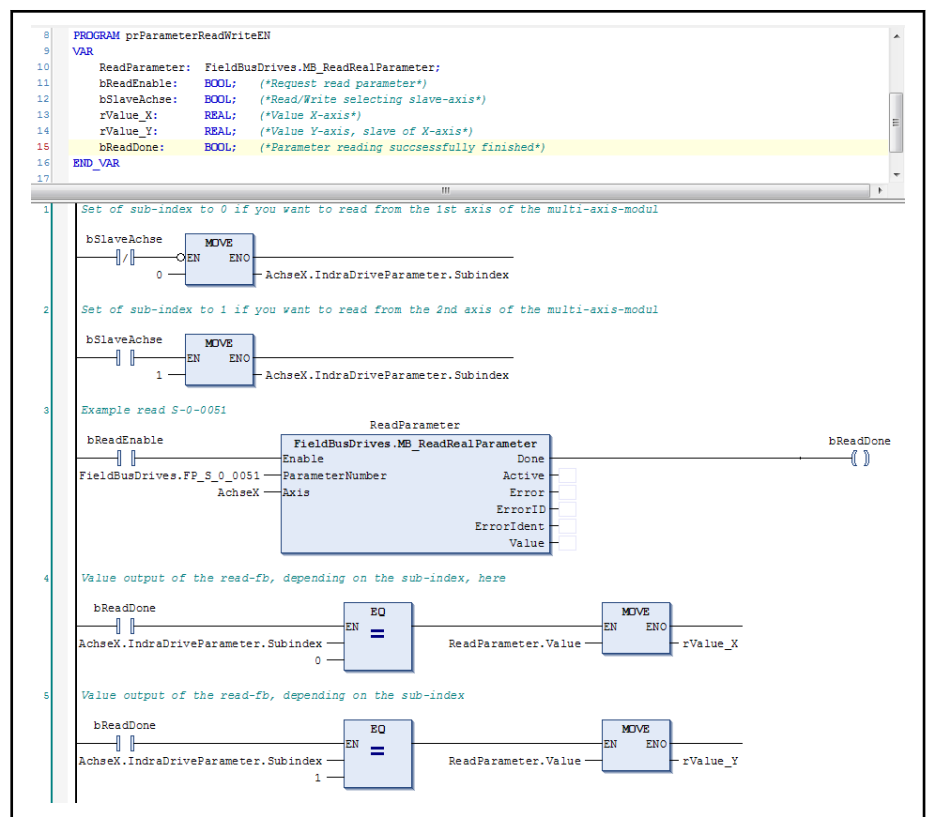


Fig. 7-28: Program sample parameter reading of twin or multi-axis module

For all PLCOpen parameter blocks, the subindex (default = 0) has to be specified in the AXIS_REF of the multi-axis module (default = 0).

For double-axis type or CDD link applies that only one axis can be accessed.

If parameters of several axes have to be read or written, reading and writing has to take place subsequently.

7.5.4 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	Range
RMB_PLCOpenFieldBus	Parameter_FBs

Tab. 7-25: Library assignment

Interface description

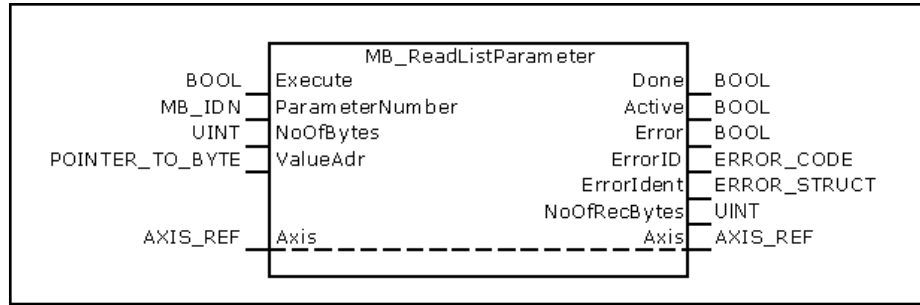


Fig. 7-29: MB_ReadListParameter FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it
VAR_INPUT	Execute	BOOL	Reading the parameter is triggered by the positive edge at "Execute"
	Parameter number	MB_IDN	Number of the parameter (e.g. FP_S_0_0000, also refer to Parameter constants, page 387)
	NoOfBytes	UINT	Maximum number of data to be read in bytes
	ValueAdr	POINT- ER_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	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	NoOf RecBytes	UINT	Number of bytes actually read

Tab. 7-26: Interface of the MB_ReadListParameter function block

Signal-time diagram

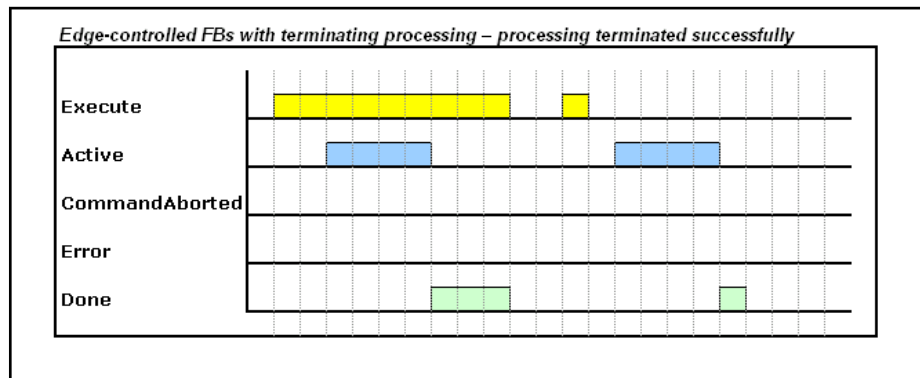


Fig. 7-30: Default signal-time diagram

Program example: MB_ReadList-Parameter

The program example shows the reading of a list parameter (P-0-0072) with 1,024 values (DINT) and the saving to the ReadListArray.

Program example: declaration section

```

PROGRAM ReadListParameter
VAR IN OUT
  RA1: ^AXIS_REF; (* Fieldbus axis RA1 *)
END_VAR

VAR
  ReadListArray: ARRAY [0..1023] OF DINT; (* Data array for the received bytes *)
  FB_ReadListParameter: MB_ReadListParameter;
  bExecute_RLP: BOOL; (* Start FB *)
  diParameterNo_RLP: MB_IDN := FP_P_0_0072; (* CAM parameter P-0-0072 *)
  uiNoOfBytes_RLP: UINT := 4096; (* DINT := 4 Bytes *)
  bDone_RLP: BOOL; (* TRUE := FB ready *)
  bActive_RLP: BOOL; (* FB works *)
  bError_RLP: BOOL; (* Error occurred *)
  enErrorID_RLP: ERROR_CODE; (* Error class *)
  stErrorIdent_RLP: ERROR_STRUCT; (* Error definition *)
  uiNoOfRecBytes_RLP: UINT; (* Number of received bytes *)
END_VAR

```

Program example in the function block diagram

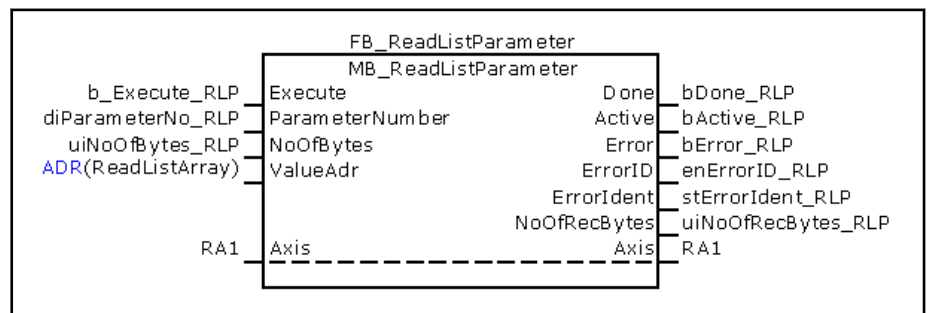


Fig. 7-31: FBD program example: ReadListParameter

Program example in structured text

```

FB_ReadListParameter(
  Execute:= bExecute_RLP, (* Start FB *)
  ParameterNumber:= diParameterNo_RLP, (* CAM parameter P-0-0072 *)
  NoOfBytes:= uiNoOfBytes_RLP, (* DINT := 4 Bytes *)
  ValueAdr:= ADR(ReadListArray), (* Address of ReadListArray *)
  Axis:= RA1, (* Fieldbus axis *)
  Done=> bDone_RLP, (* TRUE := FB ready *)
  Active=> bActive_RLP, (* FB works *)
  Error=> bError_RLP, (* Error occurred *)
  ErrorId=> enErrorID_RLP, (* Error class *)
  ErrorIdent=> stErrorIdent_RLP, (* Error definition *)
  NoOfRecBytes=> uiNoOfRecBytes_RLP; (* Number of received bytes *)
)

```



The MB_ReadListParameter function block only runs together with the IndraDrive drives.

Troubleshooting

The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```

Error := TRUE;
ErrorID := ACCESS_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#F60B; (*Command execution not possible (transfer still active)*)
ErrorIdent.Additional2 := 16#0000;

```

or

F_RELATED_TABLE

```

Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;

```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.5.5 MB_ReadListSegParameterPtr

Brief description The function block returns the value of a range of a user-specific parameter. A pointer addressing a memory space of no more than 64 kB as defined by the user is transferred to the block (memory in the PLC). An offset (location from where the data should be read related to the beginning in the parameter) and the desired use data length (NoOfBytes) is transferred (NoOfBytes ≤ maximum number of parameter bytes that can be read - offset). The return value is the number of bytes actually read.



1. The offset is to be selected in such a way that the first data element to be read can be read completely.
2. The data type of the elements of the parameter to be read and the data type of the elements of the ARRAY in the PLC have to match.

Library	Range
RMB_PLCPopenFieldBus	Parameter_FBs

Tab. 7-27: Library assignment

Interface description

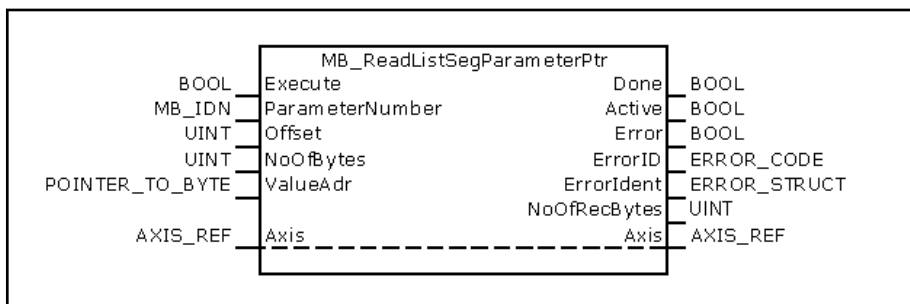


Fig. 7-32: MB_ReadListSegParameterPtr FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it
VAR_INPUT	Execute	BOOL	Reading the parameter is triggered by the positive edge at "Execute"
	Parameter number	MB_IDN	Number of the parameter (e.g. FP_S_0_0000, also refer to Access to data of a drive (IndraDrive) , page 387)
	Offset	UINT	Offset in bytes related to the beginning of the parameter from where it is to be read
	NoOfBytes	UINT	Maximum number of data to be read in bytes

	Name	Type	Comment
	ValueAdr	POINT-ER_TO_BYTE	It points to the initial address of the memory space in the PLC 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	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	NoOf RecBytes	UINT	Number of bytes actually read

Tab. 7-28: Interface of the MB_ReadListSegParameterPtr FB

Signal-time diagram

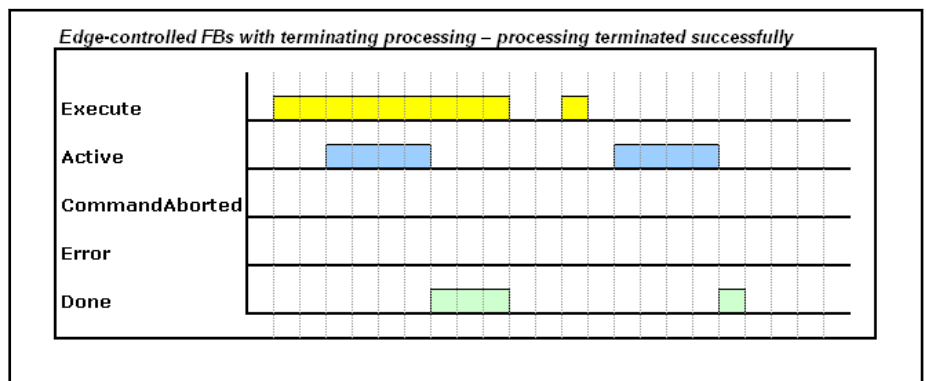


Fig. 7-33: Default signal-time diagram

Program example: MB_ReadList-SegParameterPtr

The program example shows partially the reading of a list parameter (P-0-0072) from byte 128 with a length of 512 bytes and the saving to the ReadListArray.

Program example: declaration section

```

PROGRAM ReadListSegParameterPtr
VAR IN_OUT
  RA1: ^AXIS_REF; // Fieldbus axis RA1
END_VAR
VAR
  ReadListArray: ARRAY [0..1023] OF DINT; // Data array for the received bytes

  FB ReadListSegParameterPtr: MB_ReadListSegParameterPtr;
  bExecute_RLSPP: BOOL; // Start FB
  diParameterNo_RLSPP: MB_IDN := FP_P_0_0072; // CAM parameter P-0-0072
  uiOffset_RLSPP: UINT := 128; // beginning with byte 128
  uiNoOfBytes_RLSPP: UINT := 512; // 512 bytes (128 DINT)
  bDone_RLSPP: BOOL; // TRUE := FB ready
  bActive_RLSPP: BOOL; //
  bError_RLSPP: BOOL; // Error occurred
  enErrorID_RLSPP: ERROR_CODE; // Error class
  stErrorIdent_RLSPP: ERROR_STRUCT; // Error definition
  uiNoOfRecBytes_RLSPP: UINT; // Number of received bytes
END_VAR

```

Program example in the function block diagram

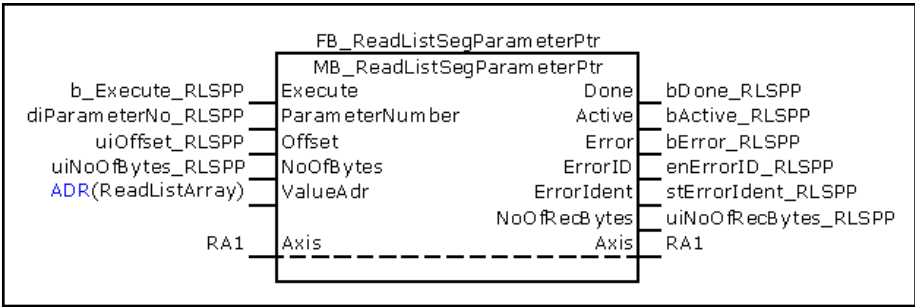


Fig. 7-34: FBD program example: MB_ReadListSegParameterPtr

Program example in structured text

```
FB_ReadListSegParameterPtr(  
  Execute := bExecute_RLSPP, // Start FB  
  ParameterNumber := diParameterNo_RLSPP, // Parameter Cam data  
  Offset := uiOffset_RLSPP, // beginning with Byte 128  
  NoOfBytes := uiNoOfBytes_RLSPP, // 512 bytes (128 DINT)  
  ValueAdr := ADR(ReadListArray), // Address of ReadListArray  
  Axis := Antriebl, // Fieldbus axis RA1  
  Done=> bDone_RLSPP, // TRUE := FB ready  
  Active=> bActive_RLSPP, //  
  Error=> bError_RLSPP, // Error occurred  
  ErrorId=> enErrorID_RLSPP, // Error class  
  ErrorIdent=> stErrorIdent_RLSPP: // Error definition  
  NoOfRecBytes=> uiNoOfRecBytes_RLSPP); // Number of received bytes
```

 The MB_ReadListSegParameterPtr function block only runs together with the IndraDrive drives.

Troubleshooting The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```
Error := TRUE;  
ErrorID := ACCESS_ERROR;  
ErrorIdent.Table := F_RELATED_TABLE;  
ErrorIdent.Additional1 := 16#F60B; (*Command execution not possible (transfer still active)*)  
ErrorIdent.Additional2 := 16#0000;
```

or
F_RELATED_TABLE

```
Error := TRUE;  
ErrorID := RESOURCE_ERROR;  
ErrorIdent.Table := F_RELATED_TABLE;  
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)  
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.5.6 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".

Library	Range
RMB_PlcOpenFieldBus	Parameter_FBs

Tab. 7-29: Library assignment

Interface description

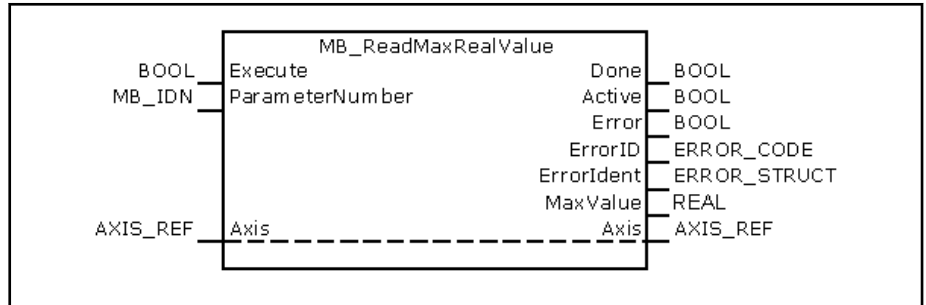


Fig. 7-35: MB_ReadMaxRealValue FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
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 (refer to Parameter constants, page 387)
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
	MaxValue	REAL	If "Done" = TRUE, "MaxValue" indicates the maximum input value allowed of the parameter selected.

Tab. 7-30: Interface of the MB_ReadMaxRealValue function block

Signal-time diagram

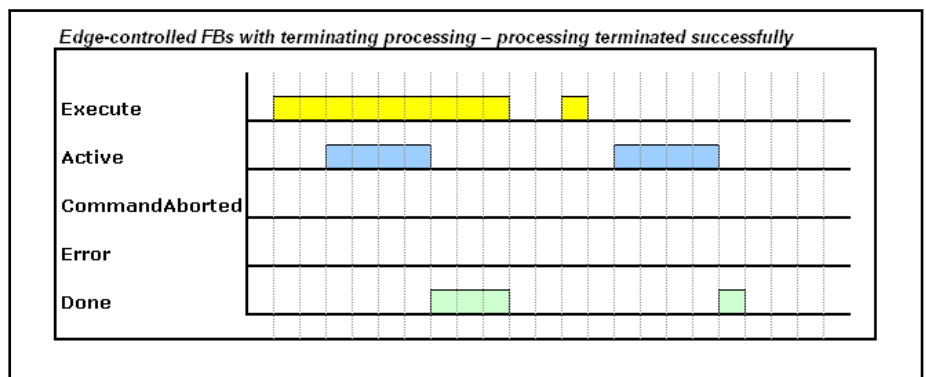


Fig. 7-36: Default signal-time diagram



The MB_ReadMaxRealValue function block from the RMB_PLcOpenFieldBus.Library only runs together with the IndraDrive drives.

Troubleshooting

The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := ACCESS_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#F60B; (*Command execution not possible (transfer still active)*)
ErrorIdent.Additional2 := 16#0000;
```

or

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.5.7 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".

Library	Range
RMB_PlcOpenFieldBus	Parameter_FBs

Tab. 7-31: Library assignment

Interface description

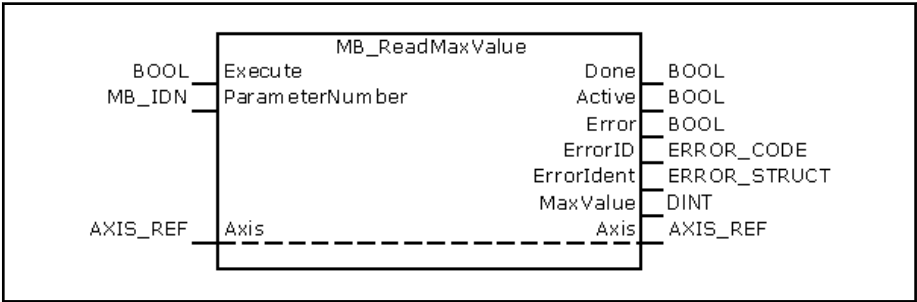


Fig. 7-37: MB_ReadMaxValue FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
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 (refer to Parameter constants, page 387)
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
	MaxValue	DINT	If "Done" = TRUE, "MaxValue" indicates the maximum input value allowed of the parameter selected.

Tab. 7-32: Interface of the MB_ReadMaxValue FB

Brief description

Signal-time diagram

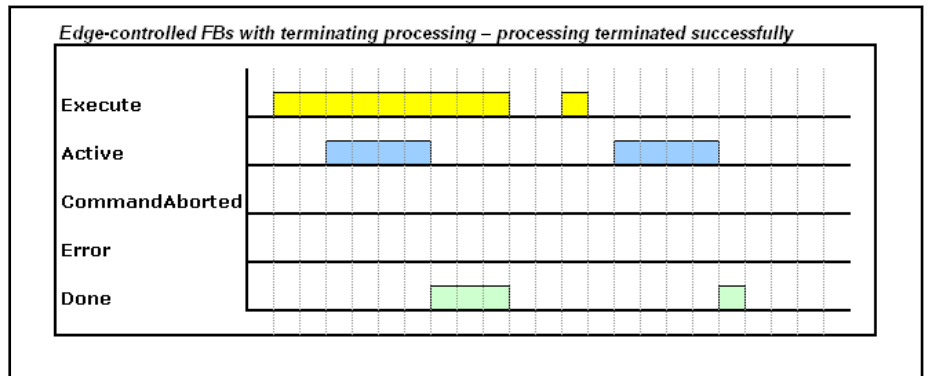


Fig. 7-38: Default signal-time diagram



The MB_ReadMaxValue function block from the RMB_PLCOpenFieldBus.Library only runs together with the IndraDrive drives.

Troubleshooting

The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := ACCESS_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#F60B; (*Command execution not possible (transfer still active)*)
ErrorIdent.Additional2 := 16#0000;
```

or

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.5.8 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".

Library	Range
RMB_PLCoOpenFieldBus	Parameter_FBs

Tab. 7-33: Library assignment

Interface description

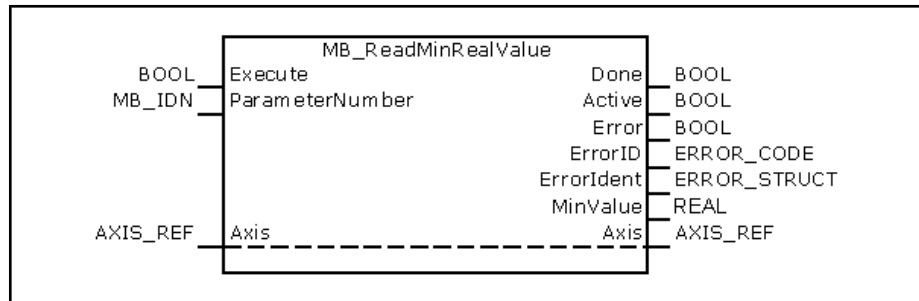


Fig. 7-39: MB_ReadMinRealValue FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
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 (refer to Parameter constants, page 387)
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
	MinValue	REAL	If "Done" = TRUE, "MinValue" indicates the minimum input value allowed of the parameter selected.

Tab. 7-34: Interface of the MB_ReadMinRealValue function block

Signal-time diagram

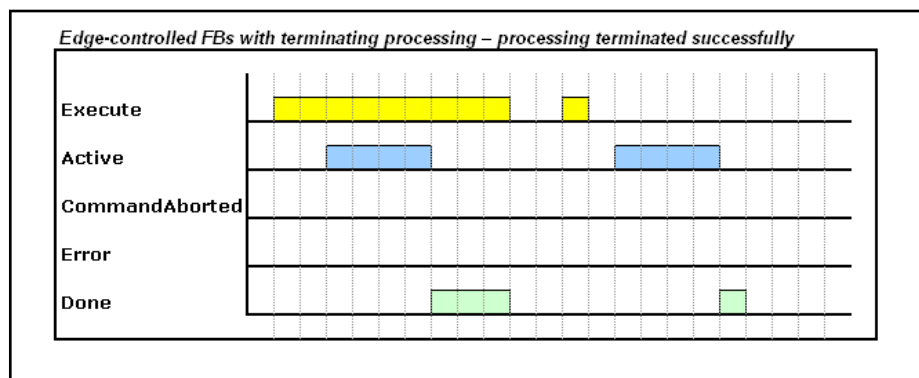


Fig. 7-40: Default signal-time diagram



The MB_ReadMinRealValue function block from the RMB_PLCoOpenFieldBus.Library only runs together with the IndraDrive drives.

Troubleshooting The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := ACCESS_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#F60B; (*Command execution not possible (transfer still active)*)
ErrorIdent.Additional2 := 16#0000;
```

or

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.5.9 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".

Library	Range
RMB_PlcOpenFieldBus	Parameter_FBs

Tab. 7-35: Library assignment

Interface description

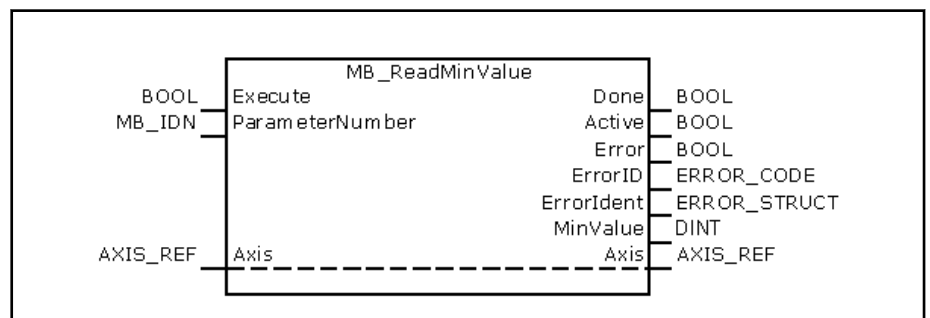


Fig. 7-41: MB_ReadMinValue FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
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 (refer to Parameter constants , page 387)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data after preprocessing running

	Name	Type	Comment
	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
	MinValue	DINT	If "Done" = TRUE, "MinValue" indicates the minimum input value allowed of the parameter selected.

Tab. 7-36: Interface of the MB_ReadMinValue FB

Signal-time diagram

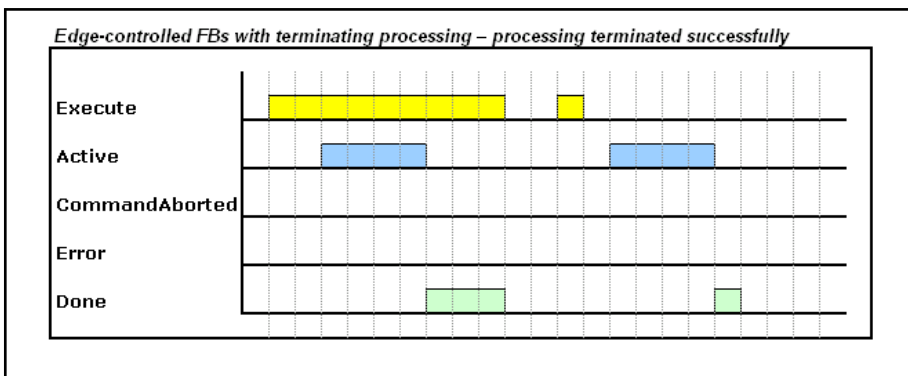


Fig. 7-42: Default signal-time diagram



The MB_ReadMinValue function block from the RMB_PLCOpenFieldBus.Library only runs together with the IndraDrive drives.

Troubleshooting

The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := ACCESS_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#F60B; (*Command execution not possible (transfer still active)*)
ErrorIdent.Additional2 := 16#0000;
```

or

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.5.10 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 the "Enable".

Library	Range
RMB_PlcOpenFieldBus	Parameter_FBs

Tab. 7-37: Library assignment

Interface description

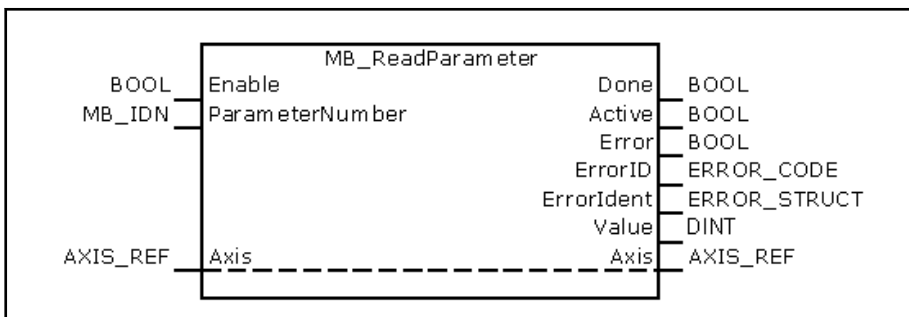


Fig. 7-43: MB_ReadParameter FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the actual axis and/or the addressed object.
VAR_INPUT	Enable	BOOL	Continuous reading of the parameter as long as "Enable" is TRUE
	Parameter number	MB_IDN	Number of the parameter (e.g. FP_S_0_0000, also refer to Parameter constants, page 387)
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
	Value	DINT	If "Done"=TRUE, "Value" displays the value of the selected parameter as DINT.

Tab. 7-38: Interface of the MB_ReadParameter FB

Signal-time diagram

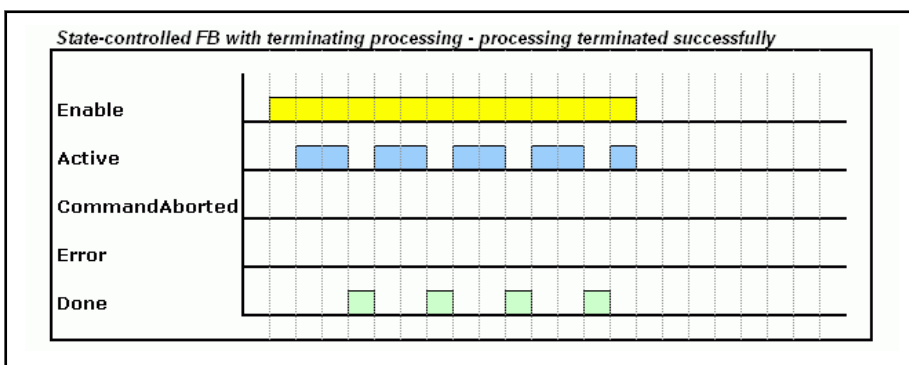


Fig. 7-44: Default signal-time diagram



The MB_ReadRealParameter function block only runs together with the IndraDrive drives.

Troubleshooting The function block generates error messages only at a positive edge or if TRUE is set at the "Enable" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := ACCESS_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#F60B; (*Command execution not possible (transfer still active)*)
ErrorIdent.Additional2 := 16#0000;
```

or

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.5.11 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".

Library	Range
RMB_PLCopOpenFieldBus	Parameter_FBs

Tab. 7-39: Library assignment

Interface description

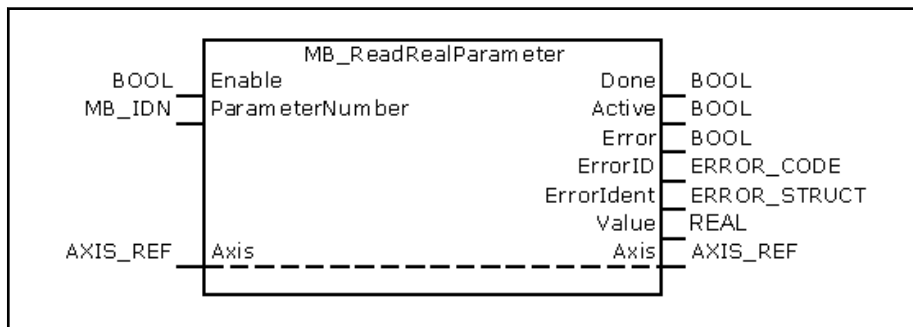


Fig. 7-45: MB_ReadRealParameter FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it
VAR_INPUT	Enable	BOOL	Continuous reading of the parameter as long as "Enable" is TRUE
	Parameter number	MB_IDN	Number of the parameter (e.g. FP_S_0_0000, also refer to Access to data of a drive (IndraDrive) , page 387)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data after preprocessing running

	Name	Type	Comment
	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
	Value	REAL	If "Done"=TRUE, "Value" displays the 4 byte value as REAL number

Tab. 7-40: Interface of the MB_ReadRealParameter FB

Signal-time diagram

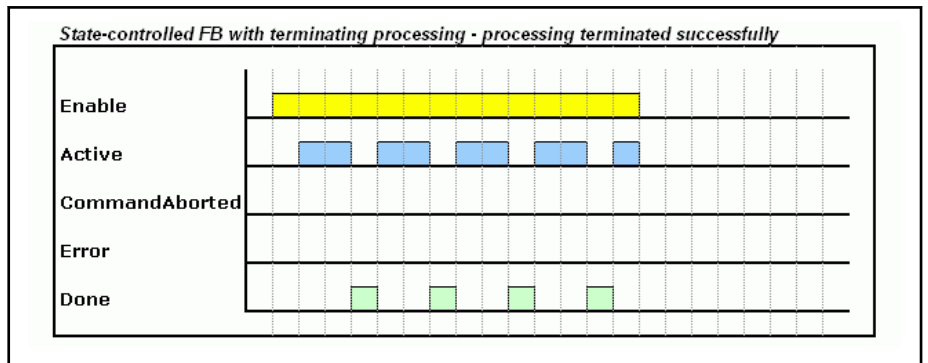


Fig. 7-46: Default signal-time diagram



The MB_ReadRealParameter function block only runs together with the IndraDrive drives.

Troubleshooting

The function block generates error messages only at a positive edge or if TRUE is set at the "Enable" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := ACCESS_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#F60B; (*Command execution not possible (transfer still active)*)
ErrorIdent.Additional2 := 16#0000;
```

or

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.5.12 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".

Library	Range
RMB_PLCoenFieldBus	Parameter_FBs

Tab. 7-41: Library assignment

Interface description

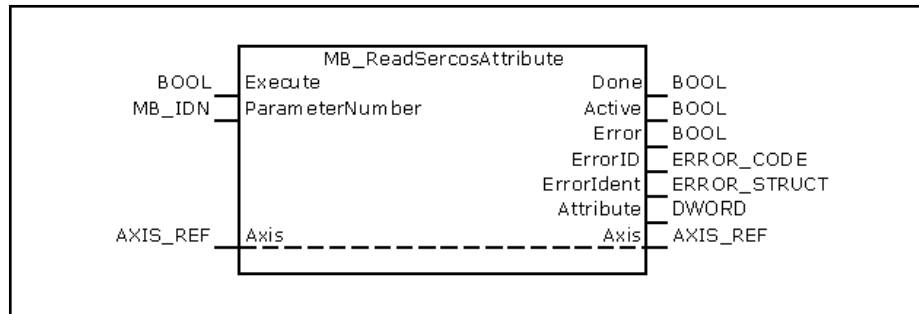


Fig. 7-47: MB_ReadSercosAttribute FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
VAR_INPUT	Execute	BOOL	Reading the parameter attribute at transition 0-1
	Parameter-Number	MB_IDN	Number of the parameter (e.g. FP_S_0_0000, also refer to Access to data of a drive (IndraDrive) , page 387)
VAR_OUTPUT	Done	BOOL	If "Done"=TRUE, "Attribute" indicates the attribute of the selected parameter
	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
	Attributes	DWORD	If "Done"=TRUE, "Attribute" indicates the attribute of the selected parameter

Tab. 7-42: Interface of the MB_ReadSercosAttribute function block

Decoding the attribute

For decoding purposes, the function block "IL_SercosAttribute" is provided in the "ML_Base.library".

Signal-time diagram

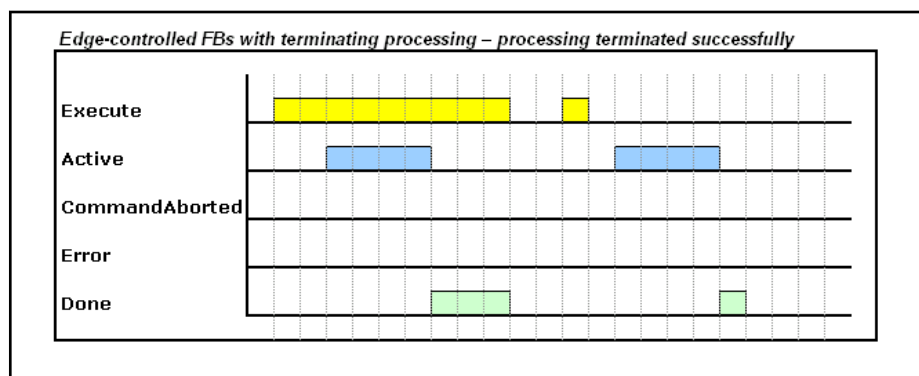


Fig. 7-48: Default signal-time diagram



The MB_ReadSercosAttribute function block only runs together with the IndraDrive drives.

Troubleshooting The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := ACCESS_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#F60B; (*Command execution not possible (transfer still active)*)
ErrorIdent.Additional2 := 16#0000;
```

or

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.5.13 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.

Library	Range
RMB_PLCopenFieldBus	Parameter_FBs

Tab. 7-43: Library assignment

Interface description

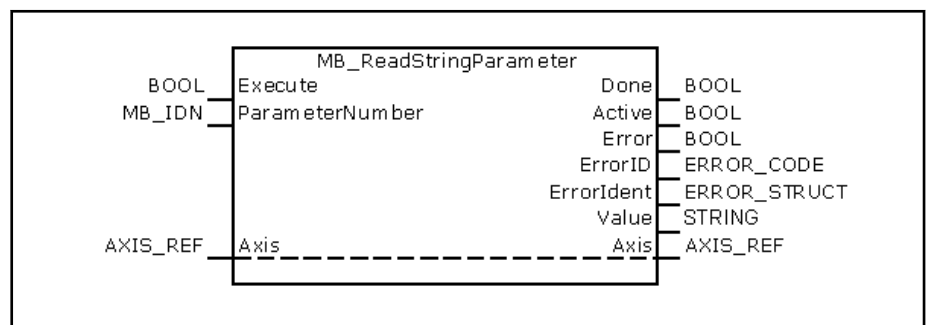


Fig. 7-49: MB_ReadStringParameter FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it
VAR_INPUT	Execute	BOOL	It starts reading the text of the parameter with a rising edge
	Parameter number	MB_IDN	Number of the parameter (e.g. FP_S_0_0000, also refer to Access to data of a drive (IndraDrive) , page 387)
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

	Name	Type	Comment
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Value	STRING	If "Done"=TRUE, "Value" displays the value of the selected parameter as character string with a maximum of 255 characters.

Tab. 7-44: Interface of the MB_ReadStringParameter FB

Signal-time diagram

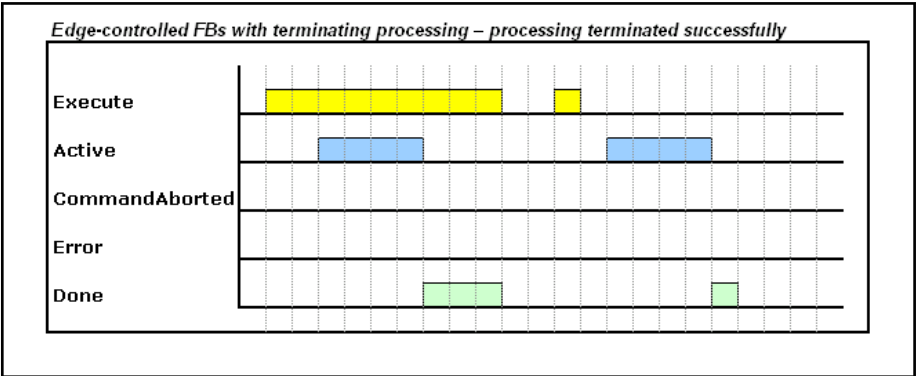


Fig. 7-50: Default signal-time diagram



The MB_ReadStringParameter function block only runs together with the IndraDrive drives.

Troubleshooting The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := ACCESS_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#F60B; (*Command execution not possible (transfer still active)*)
ErrorIdent.Additional2 := 16#0000;
```

or

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.5.14 MB_WriteListParameter

Brief description This function block writes memory content of a defined length (NoOfBytes) provided with the initial address (ValueAdr) from the PLC to a user-specific parameter.



The specified number of bytes must exactly match the data volume to be transferred!
If this number is too high, data waste is transferred. If it is too small, data is missing in the target parameter!



The data type of the elements of the parameter to be written and the data type of the elements of the ARRAY have to match.

Library	Range
RMB_PLCopenFieldBus	Parameter_FBs

Tab. 7-45: Library assignment

Interface description

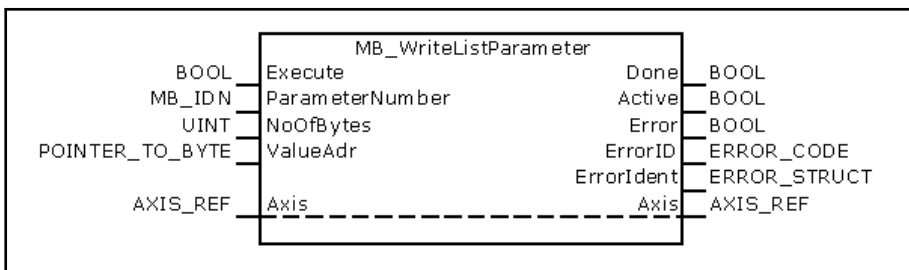


Fig. 7-51: MB_WriteListParameter FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it
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 (e.g. FP_S_0_0000, also refer to Parameter constants, page 387)
	NoOfBytes	UINT	Number of bytes to be written
	ValueAdr	POINTER_TO_BYTE	Pointer to the initial address of the memory in which the data to be written is saved.
VAR_OUTPUT	Done	BOOL	The value was successfully written 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

Tab. 7-46: Interface of the MB_WriteListParameter function block

Signal-time diagram

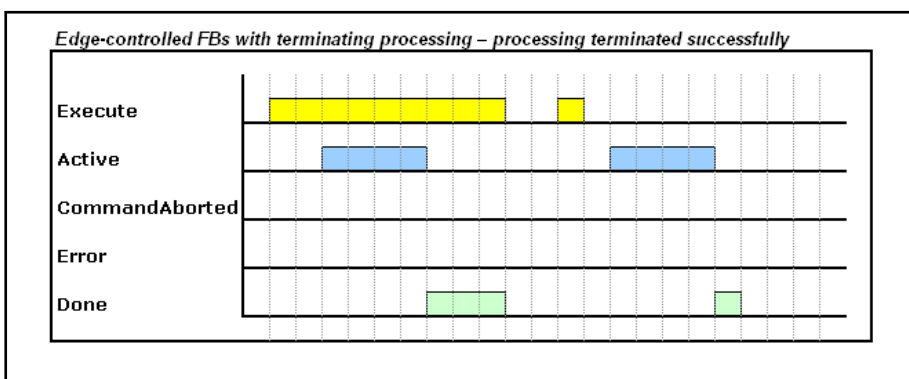


Fig. 7-52: Default signal-time diagram

Program example: MB_WriteList-Parameter

This program example shows the writing of the list parameter (P-0-0072) with 1,024 values (DINT) from the WriteListArray.

Program example: declaration section

```

PROGRAM WriteListParameter
VAR IN OUT
    RA1:                AXIS_REF;                // Fieldbus axis RA1
END_VAR
VAR
    WriteListArray:     ARRAY [0..1023] OF DINT; // Data source

FB WriteListParameter: MB_WriteListParameter;
bExecute_WLP:          BOOL;                    // Start FB
diParameterNo_WLP:    MB_IDN:= FP_P_0_0072;    // CAM parameter P-0-0072
uiNoOfBytes_WLP:      UINT:= 4096;             // DINT := 4 Bytes
bDone_WLP:            BOOL;                    // TRUE := FB ready
bActive_WLP:          BOOL;                    // FB works
bError_WLP:           BOOL;                    // Error occurred
enErrorID_WLP:        ERROR_CODE;              // Error class
stErrorIdent_WLP:     ERROR_STRUCT;            // Error definition
END_VAR

```

Program example in the function block diagram

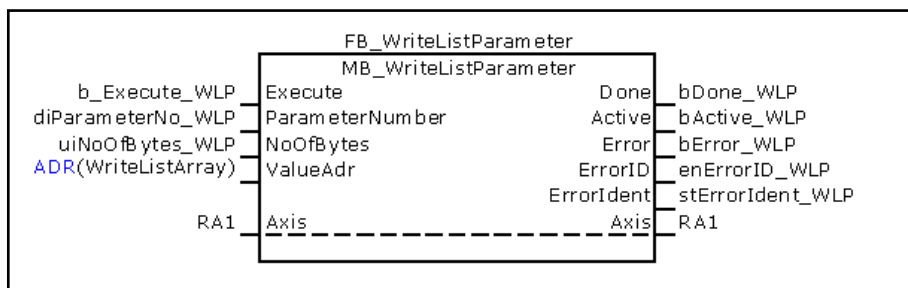


Fig. 7-53: FBD program example: WriteListParameter

Program example in structured text

```

FB_WriteListParameter(
    Execute:= bExecute_WLP,                // Start FB
    ParameterNumber:= diParameterNo_WLP,  // CAM parameter P-0-0072
    NoOfBytes:= uiNoOfBytes_WLP,          // DINT := 4 Bytes
    ValueAdr:= ADR(WriteListArray),       // Address of WriteListArray
    Axis:= RA1,                           // Fieldbus axis
    Done=> bDone_WLP,                     // TRUE := FB ready
    Active=> bActive_WLP,                 // FB works
    Error=> bError_WLP,                   // Error occurred
    ErrorId=> enErrorID_WLP,              // Error class
    ErrorIdent=> stErrorIdent_WLP);       // Error definition

```



The MB_WriteListParameter function block only runs together with the IndraDrive drives.

Troubleshooting

The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```

Error := TRUE;
ErrorID := ACCESS_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#F60B; (*Command execution not possible (transfer still active)*)
ErrorIdent.Additional2 := 16#0000;

```

or

F_RELATED_TABLE

```

Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;

```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.5.15 MB_WriteListSegParameterPtr

Brief description This function block writes memory content of a defined length (NoOfBytes) provided with the initial address (ValueAdr) in the PLC to a user-specific parameter displaced by one offset.



The specified number of bytes must exactly match the data volume to be transferred!

If this number is too high, data waste is transferred. If it is too small, data is missing in the target parameter!



1. The offset is to be selected in such a way that the first data element to be written can be written completely.
2. The data type of the elements of the parameter to be written and the data type of the elements of the ARRAY in the PLC have to match.

Library	Range
RMB_PLCopenFieldBus	Parameter_FBs

Tab. 7-47: Library assignment

Interface description

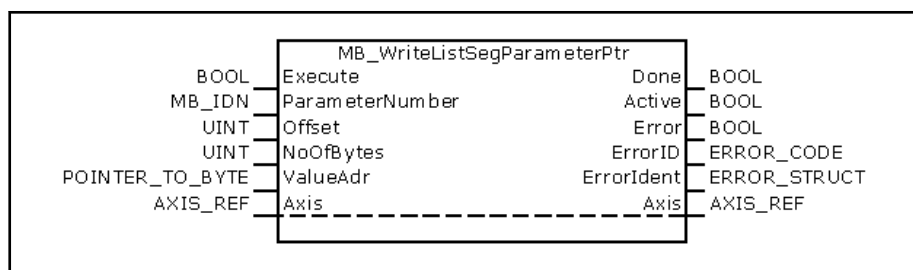


Fig. 7-54: MB_WriteListSegParameterPtr FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it
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 (e.g. FP_S_0_0000, also refer to Parameter constants , page 387)
	Offset	UINT	Offset in bytes related to the beginning of the parameter from where it is to be written
	NoOfBytes	UINT	Number of bytes to be written
	ValueAdr	POINTER_TO_BYTE	Pointer to the initial address of the memory in which the data to be written is saved.

	Name	Type	Comment
VAR_OUTPUT	Done	BOOL	The value was successfully written 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

Tab. 7-48: Interface of the MB_WriteListSegParameterPtr FB

Signal-time diagram

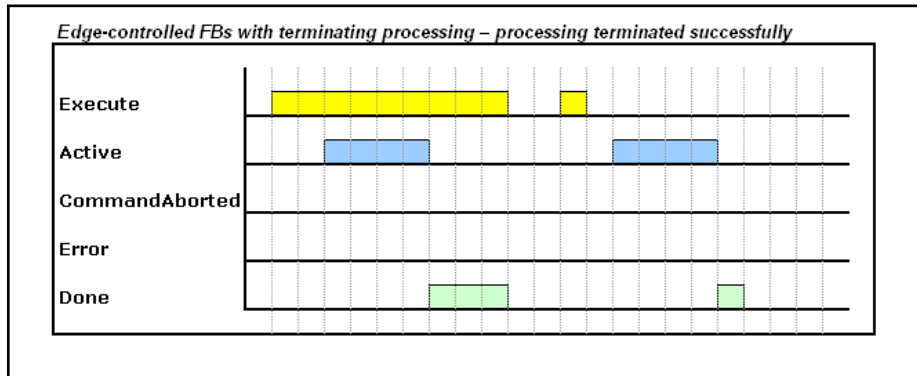


Fig. 7-55: Default signal-time diagram

Program example: MB_WriteListSegParameterPtr

This program example shows the writing of 512 bytes to the list parameter (P-0-0072) starting from byte 128 from the WriteListArray.

Program example: declaration section

```

PROGRAM WriteListSegParameterPtr
VAR IN_OUT
    RA1:                AXIS_REF;                // Fieldbus axis RA1
END_VAR
VAR
    WriteListArray:     ARRAY [0..1023] OF DINT;  // Data source

FB_WriteListSegParameterPtr: MB_WriteListSegParameterPtr;
bExecute_WLSPP:           BOOL;                  // Start FB
diParameterNo_WLSPP:     MB_IDN:= FP_P_0_0072;  // CAM parameter P-0-0072
uiOffset_WLSPP:          UINT:= 128;            // beginning with Byte 128
uiNoOfBytes_WLSPP:       UINT:= 512;            // 512 bytes (128 DINT)
bDone_WLSPP:             BOOL;                  // TRUE := FB ready
bActive_WLSPP:           BOOL;                  // FB works
bError_WLSPP:            BOOL;                  // Error occurred
enErrorID_WLSPP:         ERROR_CODE;            // Error class
stErrorIdent_WLSPP:      ERROR_STRUCT;          // Error definition
END_VAR

```

Program example in the function block diagram

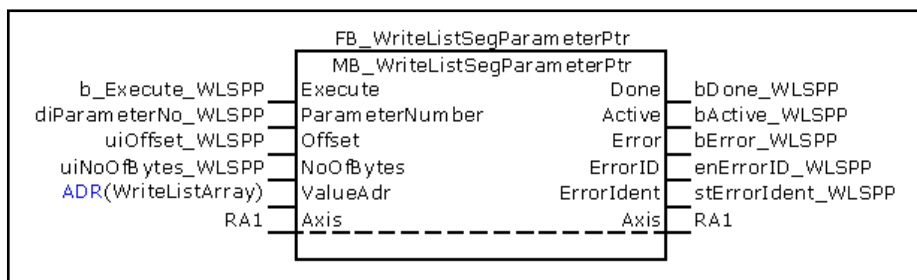


Fig. 7-56: FBD program example: WriteListSegParameterPtr

Program example in structured text

```

FB_WriteListSegParameterPtr(
  Execute:=      bExecute_WLSPP,           // Start FB
  ParameterNumber:= diParameterNo_WLSPP,   // CAM parameter P-0-0072
  Offset:=       uiOffset_WLSPP,           // beginning with Byte 128
  NoOfBytes:=    uiNoOfBytes_WLSPP,        // 512 bytes (128 DINT)
  ValueAdr:=     ADR(WriteListArray),      // Address of WriteListArray
  Axis:=         RA1,                      // Drive address
  Done=>         bDone_WLSPP,              // TRUE := FB ready
  Active=>       bActive_WLSPP,            // FB works
  Error=>        bError_WLSPP,             // Error occurred
  ErrorID=>      enErrorID_WLSPP,          // Error class
  ErrorIdent=>   stErrorIdent_WLSPP);      // Error definition

```



The MB_WriteListSegParameterPtr function block only runs together with the IndraDrive drives.

Troubleshooting

The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```

Error := TRUE;
ErrorID := ACCESS_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#F60B; (*Command execution not possible (transfer still active)*)
ErrorIdent.Additional2 := 16#0000;

```

or

F_RELATED_TABLE

```

Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;

```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.5.16 MB_WriteParameter

Brief description The function block writes a value in a parameter specified by the user.

Library	Range
RMB_PLCopenFieldBus	Parameter_FBs

Tab. 7-49: Library assignment

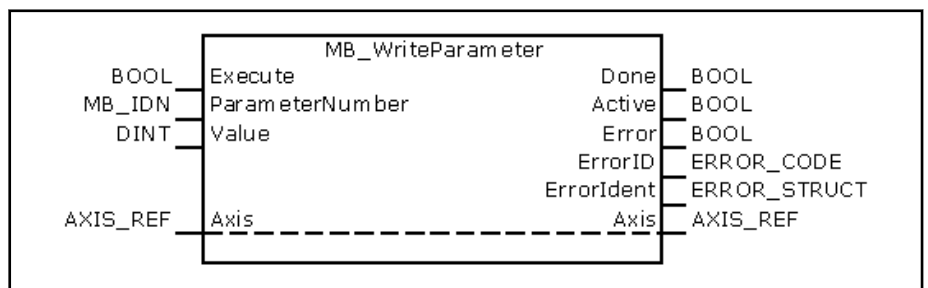
Interface description

Fig. 7-57: MB_WriteParameter FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it
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 (e.g. FP_S_0_0000, also refer to Parameter constants, page 387)
	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

Tab. 7-50: Interface of the MB_WriteParameter FB

Signal-time diagram

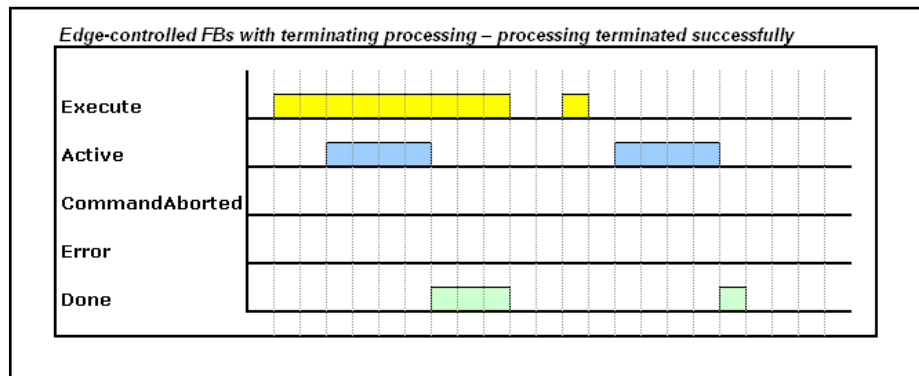


Fig. 7-58: Default signal-time diagram



The MB_WriteParameter function block only runs together with the IndraDrive drives.

Troubleshooting

The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := ACCESS_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#F60B; (*Command execution not possible (transfer still active)*)
ErrorIdent.Additional2 := 16#0000;
```

or

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.5.17 MB_WriteRealParameter

Brief description This function block writes a value to a user-specific parameter.

Library	Range
RMB_PLCPopenFieldBus	Parameter_FBs

Tab. 7-51: Library assignment

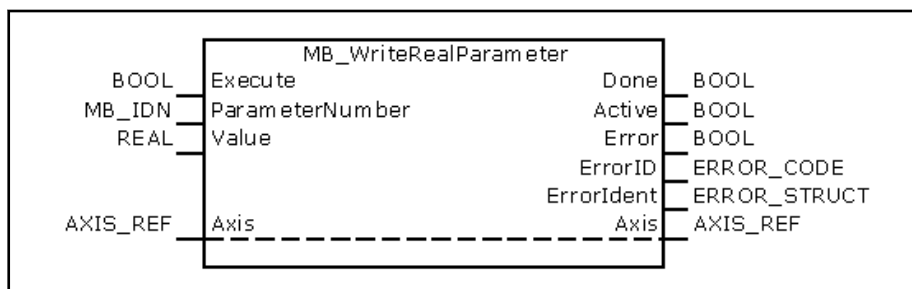


Fig. 7-59: MB_WriteRealParameter FB

Interface description

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it
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 (e.g. FP_S_0_0000, also refer to Parameter constants, page 387)
	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

Tab. 7-52: Interface of the MB_WriteRealParameter FB

Signal-time diagram

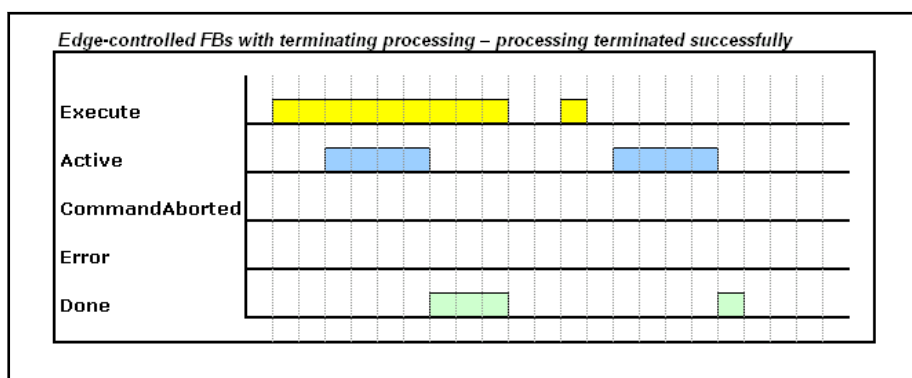


Fig. 7-60: Default signal-time diagram



The MB_WriteRealParameter function block only runs together with the IndraDrive drives.

Troubleshooting The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := ACCESS_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#F60B; (*Command execution not possible (transfer still active)*)
ErrorIdent.Additional2 := 16#0000;
```

or

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.5.18 MB_WriteStringParameter

Brief description The function block writes a character string with a maximum of 255 characters to a user-specific parameter.

Library	Range
RMB_PLCopenFieldBus	Parameter_FBs

Tab. 7-53: Library assignment

Interface description

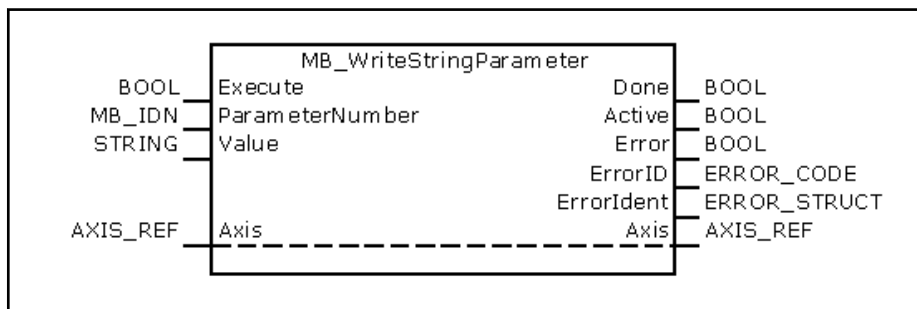


Fig. 7-61: MB_WriteStringParameter FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the desired control and the object controlled by it
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 (e.g. FP_S_0_0000, also refer to Parameter constants, page 387)
	Value	STRING	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

	Name	Type	Comment
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 7-54: Interface of the MB_WriteStringParameter FB

Signal-time diagram

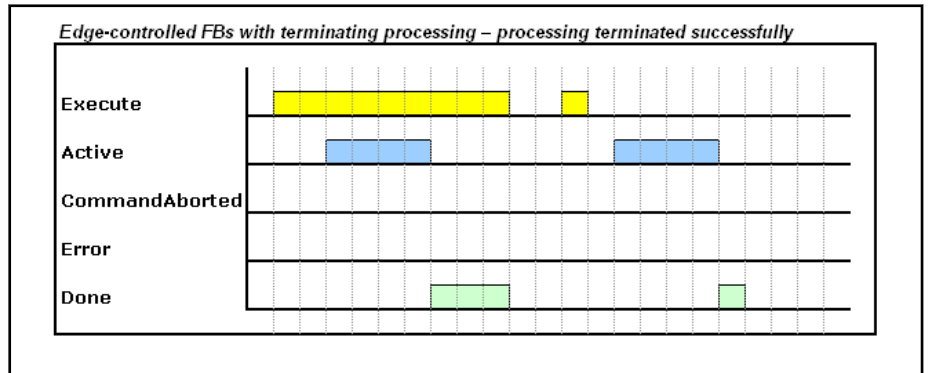


Fig. 7-62: Default signal-time diagram



The MB_WriteStringParameter function block only runs together with the IndraDrive drives.

Troubleshooting

The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := ACCESS_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#F60B; (*Command execution not possible (transfer still active)*)
ErrorIdent.Additional2 := 16#0000;
```

or

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.5.19 MC_ReadActualPosition

Brief description

The function block reads the current position of the drive.



A modification of the axis at the input of the function block requires another 0/1 transition at "Enable".

Library	Range
RMB_PLCopenFieldBus	Parameter_FBs

Tab. 7-55: Library assignment

Interface description

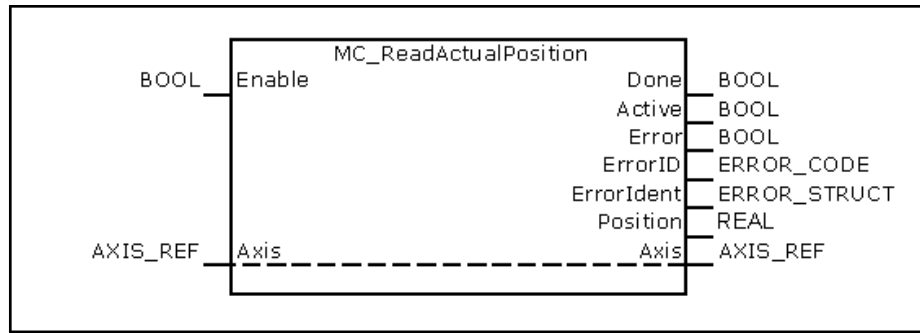


Fig. 7-63: MC_ReadActualPosition FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
VAR_INPUT	Enable	BOOL	Continuous reading of the position value as long as "Enable" is TRUE.
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Position	REAL	Current absolute position of the drive ⁷⁾

Tab. 7-56: Interface of the MC_ReadActualPosition FB

Signal-time diagram

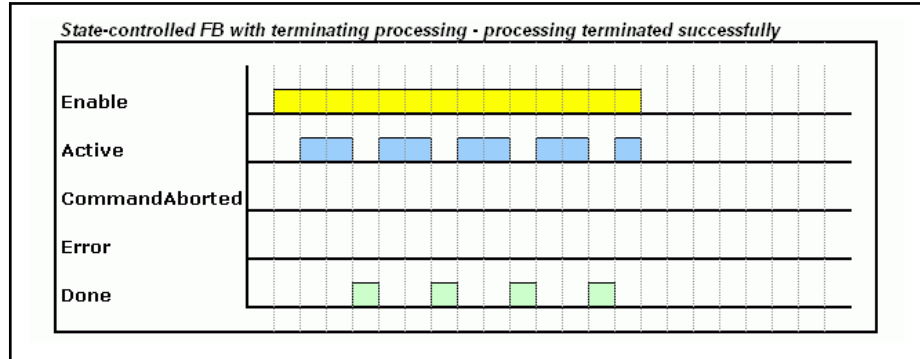


Fig. 7-64: Default signal-time diagram



The MC_ReadActualPosition function block only runs together with the IndraDrive drives.

Parameters used

The function block reads the current position of the drive "actual position value" S-0-0051 or the parameter specified in the drive on the input description of the field bus process data .

S-0-0051, the actual position value encoder 1 displays the current position of the motor encoder.

Troubleshooting

The MC_ReadActualPosition function block generates only error messages at a positive edge or if TRUE is set at the "Enable" input.

⁷⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

If there are configuration errors at the function block, the following is displayed

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

Otherwise, the task is given to the drive and it might react with an own error message.

7.5.20 MC_ReadAxisError

Brief description The function block queries the error state of the addressed axis. One instance of the function block is to be used for each axis.



A modification of the axis at the input of the function block requires another 0/1 transition at "Enable".

Library	Range
RMB_PLCopenFieldBus	Parameter_FBs

Tab. 7-57: Library assignment

Interface description

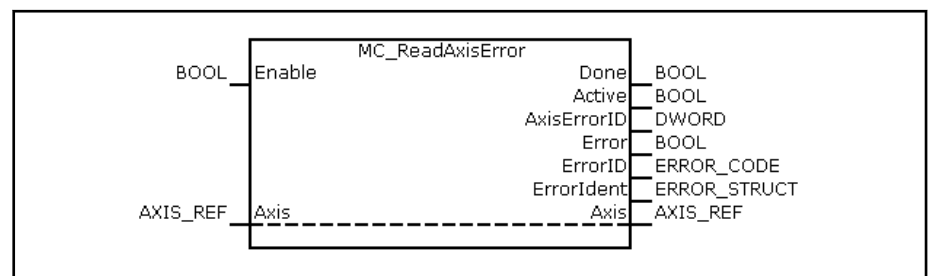


Fig. 7-65: MC_ReadAxisError FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
VAR_INPUT	Enable	BOOL	Continuous reading of the error information as long as "Enable" is TRUE.
VAR_OUTPUT	Done	BOOL	Value available
	Active	BOOL	Data processing runs when preprocessing is complete
	AxisErrorID	DWORD	It indicates the error of the drive of the axis
	Error	BOOL	It indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Short note to the error cause of the function block call, not of the axis error!
	ErrorIdent	ERROR_STRUCT	Detailed information on the error of the function block call, not the axis error!

Tab. 7-58: Interface of the MC_ReadActualPosition FB

Signal-time diagram

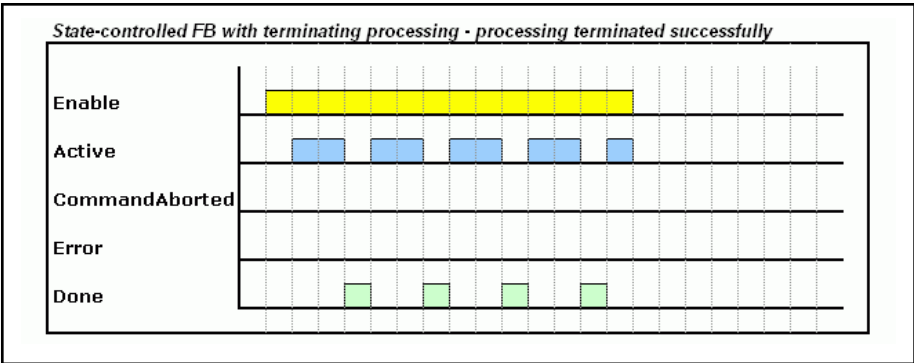


Fig. 7-66: Default signal-time diagram

 The MC_ReadAxisError function blocks only runs together with the IndraDrive drives.

- Parameters used

This function block queries the error status (S-0-0390, Diagnostic number) of the axis addressed.

This parameter stores the diagnostic number shown in the display. Thus, the control can generate individual diagnostics using this number (e.g. in languages in which the diagnostic texts are not saved in the drive).
- Troubleshooting

The MC_ReadAxisError function block generates error messages only at a positive edge or if TRUE is set at the "Enable" input.

If there are configuration errors at the function block, the following is displayed
F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

7.5.21 MC_ReadStatus

Brief description This function block reads the current status of the axis and outputs this in a decoded way.

-  A modification of the axis at the input of the function block requires another 0/1 transition at "Enable".
-  Due to the extension of the function block, it was modified in a way which is incompatible with the previous version!

Library	Range
RMB_PLCPopenFieldBus	Parameter_FBs

Tab. 7-59: Library assignment

Interface description

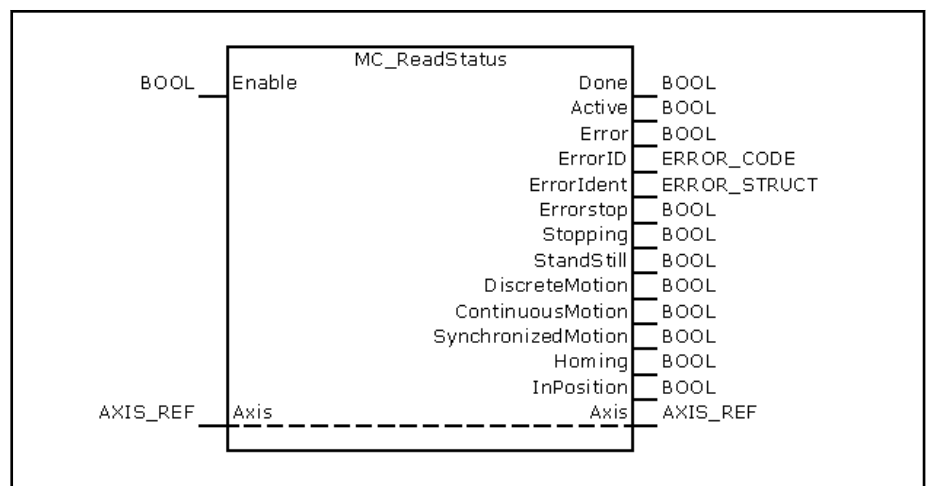


Fig. 7-67: MC_ReadStatus FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
VAR_INPUT	Enable	BOOL	Continuous reading of the axis state as long as "Enable" is TRUE.
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	It indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Errorstop	BOOL	Drive/axis in 'Errorstop' state
	Stopping	BOOL	Drive/axis in "Stopping" state
	StandStill	BOOL	Drive/axis in 'Standstill' state
	DiscreteMotion	BOOL	Drive/axis in 'DiscreteMotion' state
	ContinuousMotion	BOOL	Drive/axis in "ContinuousMotion" state
	Synchronized-Motion	BOOL	Drive/axis in 'SynchronizedMotion' state
	Homing	BOOL	Drive in 'Homing' state
	InPosition	BOOL	Drive in 'InPosition' state

Tab. 7-60: Interface of the MC_ReadStatus FB

Signal-time diagram

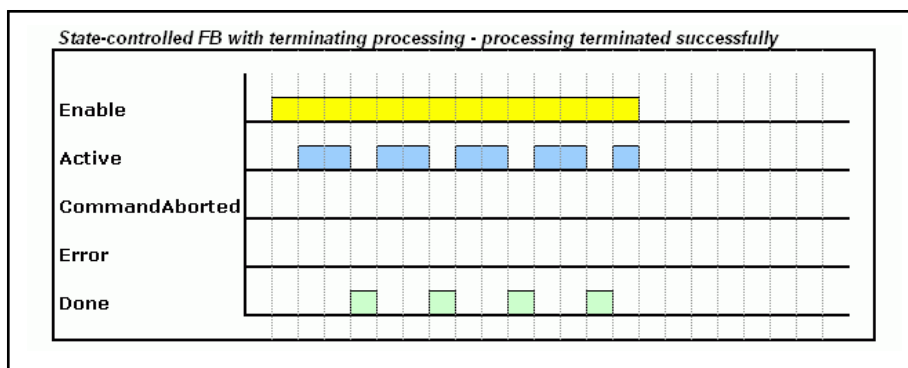


Fig. 7-68: Default signal-time diagram



The MC_ReadStatus function block only runs together with the IndraDrive drives.

Parameters used This function block reads the current axis status (S-0-0144, signal status word or the parameter specified in the drive on the input description of the field bus process data and outputs it in a decoded way.

Troubleshooting The function block generates error messages only at a positive edge or if TRUE is set at the "Enable" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

7.6 System_FBs

There are the following function blocks in the "System_FBs" folder

- MP_Drive
- MP_DriveParameter and
- MP_FB_IndraDrv_AcyclicCommunication.

The function blocks and their methods establish the communication between the control and the respective field bus drive.



These elements are not to be used, since side effects including malfunctions in the motion of the axes cannot be excluded!

7.7 PLC controls drives via field bus connection (field bus axes)

7.7.1 General information

IndraDrives can be controlled by an IndraControl via a field bus, depending on the drive interface.



The drives are connected to the control via a field bus.
The coupling via a Sercos II or Sercos III bus is not part of the description.

The activation is executed from the PLC program using MotionLogic function blocks according to the PLCopen concept.

Before operating such a field bus axis at the field bus, the following requirements have to be met:

- An IndraWorks project is created
- A control has been added to the project
- A field bus interface (onboard or function module) is set as master

Depending on the used library, declare the drives in the application:

- For RMB_PLCopenFieldBus.library, refer to [chapter 7.7.3 "Drive settings for the PLC communication" on page 422](#)
- For RMB_PLCopenEtherCAT_SoE.library, refer to ["Import Drive_XY drive" on page 433](#)

The PLC programs use MotionLogic function blocks, functions and data types summarized in several [Libraries, page 423](#).

These libraries are integrated together with the PLC program(s) into the IndraWorks project.

Additionally to the controlling of the field bus drive via the PLC program, values have to be exchanged with the drive saved in its parameters.

7.7.2 Creating field bus axes in the project

System and drive settings are required in the IndraWorks project at the PLC to ensure a correct communication between the drives and the PLC. The field bus axes are regular field bus devices at the field bus.

Drag the required drives from the library to the field bus master object.

Optionally, add a drive via the context menu of the master object: **Add ► Slave ► Drives**.

In this case, the new drive is added as last drive below the field bus master object.

Required settings for a PROFIBUS device

The drive has several sub devices in a Profibus device.

The drives (device files) in their original state are not provided with the required settings to communicate together with the PLCopen function blocks from the RMB_PLCopenFieldBus.Library via Profibus.

For IndraDrive, the following settings are required:

- ParamCh_IDN,
- F-Module_not_used, (if the module is not used)
- Input_8_Words,
- Output_10_Words.

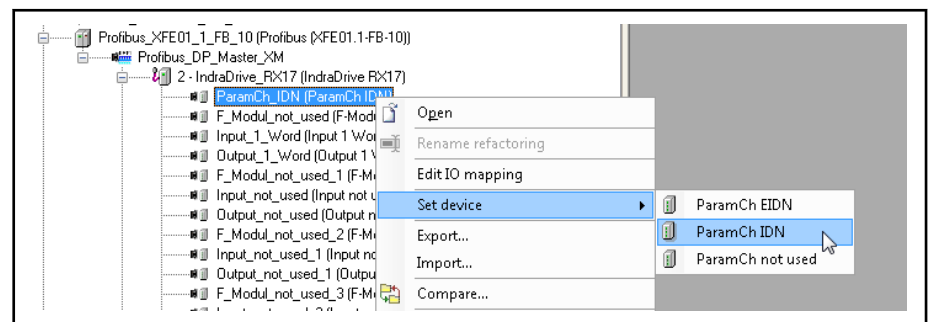


Fig. 7-69: IndraDrive, setting the parameter channel on the sub devices

Required settings for an EtherCAT device

Even the drives are not configured in their original state for such a communication with a PLC via Profibus.

Therefore, drive settings have to be made for the PLC communication.

For EtherCAT devices, the following settings are required:

- Configuring the first secondary operation mode, see [chapter 8.2.1 "Configuring the operation modes" \(e.g. with FWA MPB20\)](#) on page 430
- Configuration of signal status word and signal control word, see [chapter 8.2.2 "Configuration of signal status word and signal control word"](#) on page 430
- Configuring process data, see ["Configuring process data"](#) on page 432
- Configuring cyclic data, see [chapter 8.2.6 "Configuring cyclic data"](#) on page 437

7.7.3 Drive settings for the PLC communication

General information

To activate field bus drives with the PLCopen function blocks, some basic settings are required regarding the communication of the PLCopen functions blocks with the field bus drives.



The settings described do only refer to the communication between the drive and the PLC and do NOT describe the general drive commissioning.

The required settings can be selected via the "IndraWorks Engineering" or via "IndraWorks DS".



Currently, it **cannot** be parameterized via the Profibus connection of the PLC.

From the PC, on which the IndraWorks Engineering is running, the communication to parameterization has to be established via RS232 serial or via the existing Profibus DP master.

In the following, the parameterization is described via serial communication.

IndraDrive communication settings



The following settings do not apply to the operation of drives via EtherCAT.

For EtherCAT, select the following configurations, see [chapter 8.2 "IndraDrive – Required configurations"](#) on page 429

General information

In the following, the IndraDrive settings required for the communication are described.

Setting bus device address

Set the bus device address of the drive in the node "Master communication with regard to the set station address.

Length of parameter channel

By entering the parameter P-0-4083 directly using the parameter editor, the length of the parameter channel is set to 10 bytes.

A newer IndraDrive firmware (from MPH-03VRS) automatically sets to 5 words (ParamCH_5_Words).

Setting process data and profile type

The profile type "Freely configurable mode" and the parameters of the cyclic data exchange with the PLC are set in the following:

Process data inputs (AT)

- P-0-4078, Field bus status word
- S-0-0051, Actual position value encoder 1
- S-0-0040, Actual velocity value
- S-0-0390, Diagnostic number
- S-0-0144, Signal status word

Process data outputs (MDT)

- P-0-4077, Field bus control word
- S-0-0282, Positioning command value
- S-0-0259, Positioning velocity
- S-0-0260, Positioning velocity
- S-0-0359, Positioning deceleration
- S-0-0145, Signal control word

Signal control word Bit 0: S-0-0346: Positioning control word, bit number 4

Signal status word Bit 0: S-0-0343: Message interpolator stopped, bit number 0

Operation mode settings As primary operation mode, the operating mode "Drive-controlled positioning" has to be set.

"Velocity control" has to be assigned as first secondary operation mode.

The second and following secondary operation modes are not relevant. These modes are not selected by the PLCopen function blocks.



The MB_Home function block only references if the drive is parameterized for referencing.

7.7.4 Including required libraries

The PLCopen concept was implemented for IndraWorks Engineering for field bus types and is supported by the following libraries:

Sercos II and Sercos III

- ML_PLCopen.library, function blocks of PLCopen for Sercos drives.
- RIL_CommonTypes.library, system-comprehensive data types, addressing, error management.
- ML_Base.library, supplementing data types to the ML* system,
- Supplementing libraries, e.g. RIL_Uutilities.library.

PROFIBUS DP

- RMB_PLCopenFieldBus.library, function blocks and data types for Profibus, Profinet and Ethernet drives.
- RIL_CommonTypes.library, error management.
- Supplementing libraries, e.g. RIL_Uutilities.library.

EtherCAT

- RMB_PLCopenEtherCAT_SoE.library, function blocks and data types for EtherCAT drives.

It was tried to maintain a uniform concept for field bus types. This also affects identical names for functionally identical function blocks and data types.

Adding required libraries The RMB_PLCopenFieldBus.Library and RMB_PLCopenEtherCAT_SoE.Library libraries can be added via the library manager **Add library...**

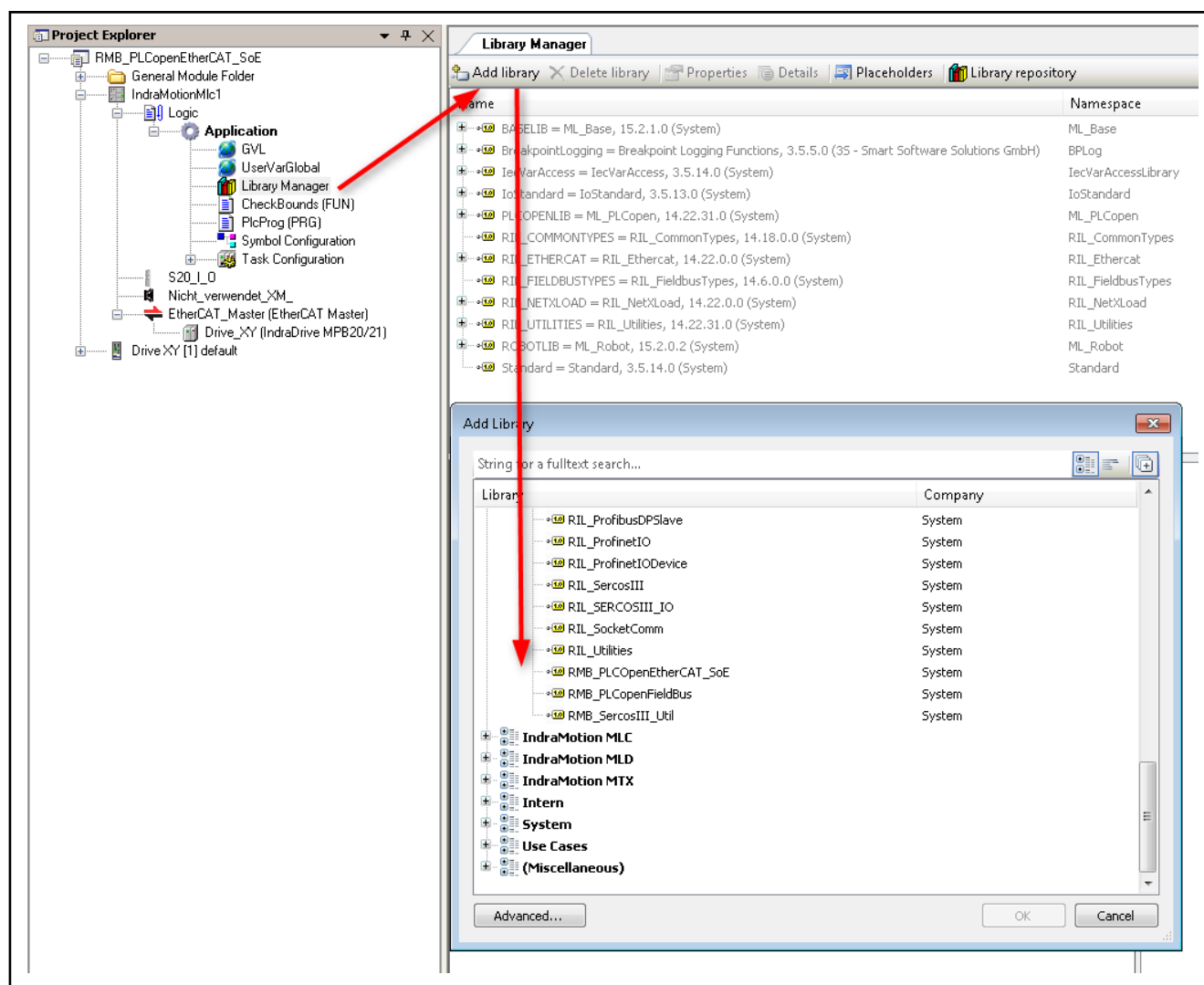


Fig. 7-70: Example: Library Manager

The elements of the RMB_PLcOpenFieldBus.library and RMB_PLcOpenEtherCAT_SoE.Library have to be used by specifying the namespace.

7.7.5 Addressing field bus drives in the PLC project

AXIS_REF, axis addresses of field bus drives

The structure **AXIS_REF** (RIL_CommonTypes) contains address information on the desired control and on the object controlled by this control (axis number).

In contrast to the **AXIS_REF** structure (RIL_CommonTypes) supporting the addressing of Sercos-based IndraDrives, the **AXIS_REF** structure (RMB_PLcOpenFieldbus) is extended by two additional elements (MP_Drive and MP_DriveParameter). These two elements are only used by the system.

The axis number can differ from the station number set at the drive.

The Global Variable List "MLCVarGlobal" with all Sercos-based axes is automatically created and maintained for Sercos-based axes.

The global variable list "GVL_FieldBusDrives" with all field bus drives has to be manually created and maintained for the field bus drives.

The two first elements of each drive "CntrlNo" and "AxisNo" are preassigned in this list. The elements "MP_Drive" and "MP_DriveParameter" remain unchanged. Therefore, they do not have to be included in the list.

Global Variable List "GVL_FieldBusDrives", example

Namespace specification

Standard namespace for the RMB_PLCopenFieldBus.library: "FieldBus-Drives".

Program:

```
VAR GLOBAL
Drive_X: FieldBusDrives.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=FieldBusAxis_11); // IndraDrive X - example
Drive_Y: FieldBusDrives.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=FieldBusAxis_12); // IndraDrive Y - example
Drive_Z: FieldBusDrives.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=FieldBusAxis_13); // IndraDrive Z - example
Drive_n: FieldBusDrives.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=FieldBusAxis_14); // IndraDrive n - example
END_VAR
```

Standard namespace for the RMB_PLCopenEtherCAT_SoE.library: "PLCopenEtherCAT_SoE".

Program:

```
VAR GLOBAL
Drive_X: PLCopenEtherCAT_SoE.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=1001); // IndraDrive X - example
Drive_Y: PLCopenEtherCAT_SoE.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=1002); // IndraDrive Y - example
Drive_Z: PLCopenEtherCAT_SoE.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=1003); // IndraDrive Z - example
Drive_n: PLCopenEtherCAT_SoE.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=1004); // IndraDrive n - example
END_VAR
```

Declaration AXIS_REF

Drive_A: FieldBusDrives.AXIS_REF :=

The data types are in the RMB_PLCopenFieldBus.library, folder "Datatypes", element "AXIS_REF".

Assignment CntrlNo

(CntrlNo: = LOCAL_CNTRL

The enumeration value LOCAL_CNTRL is contained in RIL_Common-Types.lib, folder "Data types of logical addresses", CONTROLS.

The field bus function blocks always refer to the local PLC, that is LOCAL_CNTRL.

Assignment AxisNo

Taking the example of the RMB_PLCopenFieldBus.library:

```
AxisNo: = FieldBusAxis_1);
```

The enumeration value FieldBusAxis_1 is in the RIL_CommonTypes.lib, folder "Data types of logical addresses", OBJECTS.

To avoid ambiguities, there is an individual number range stored for field bus axes

FieldBusAxis_1 (6001) up to FieldBusAxis_99 (6099)

For field bus drives, specifications on the control and the axis number are used for the error output (ErrorIdent).

A functionality is not combined with the assignment of the axis number.



It is recommended to compare the station number and axis number.

AxisData[], axis data

The AxisData array, known from the Sercos axes, is not supported for field bus axes.

7.7.6 I/O addressing – RMB_PLCPopenFieldbus

General information

On the Profibus, the field bus drives appear as ordinary I/O modules. An I/O address has to be assigned.

To facilitate the I/O address assignment, it is directly executed as variable mapping on the Profibus module of the IndraDrive.

I/O address assignment: IndraDrive

The parameter communication, the F-module (safety technology) and the cyclic communication at IndraDrives are always individual submodules of the device.

The communication "Drive to the PLC" consists of the following:

- Subdevice (ParamCh_5_Words), 5 words for acyclic parameter communication,
- Subdevice (F-module), optional use; not used here,
- Subdevices of the cyclic data exchange (8 words input or 10 word output)

Addresses are mapped at the respective subdevice channels of the drive.

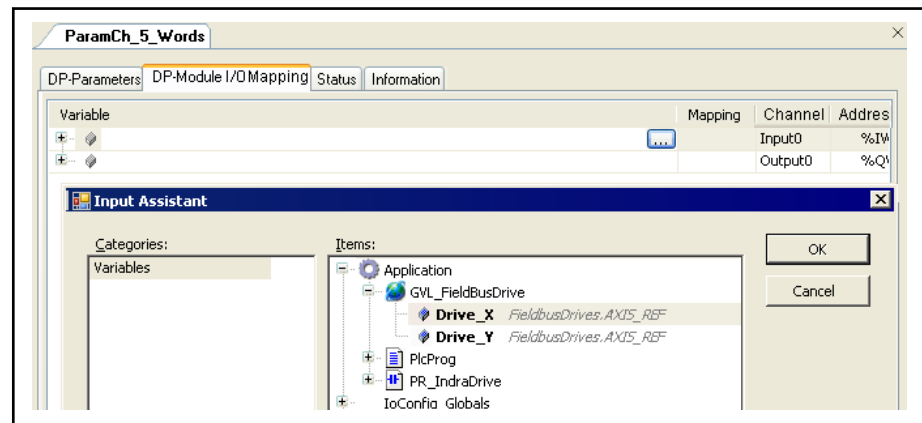


Fig. 7-71: Example: Input help for the mapping to select the drive

In AXIS_REF, the field bus drives are distinguished for the cyclic parameter exchange and "IndraDriveParameter" for the acyclic parameter exchange according to "IndraDrive"

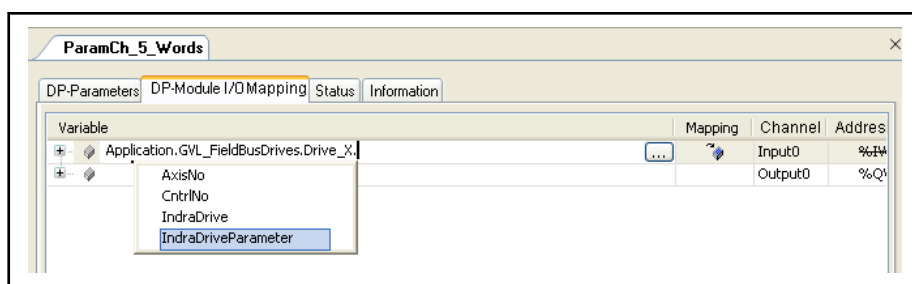


Fig. 7-72: Example: Mapping for the parameter exchange

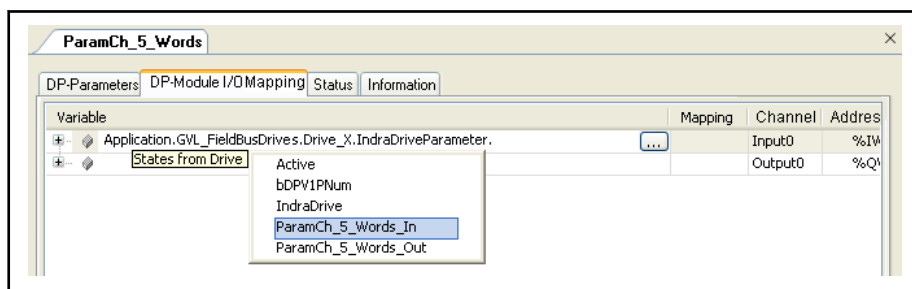


Fig. 7-73: Example: Example for the inputs of the parameter exchange

By direct mapping, the initial addresses for the parameter communication, the cyclic data exchange and - if available - an F-module with regard to the drive do not have to be considered.

7.7.7 I/O addressing – RMB_PLCOpenEtherCAT_SoE

To control the drive via function blocks of the RMB_PLCOpenEtherCAT_SoE.library, the following I/O assignments are required, see [chapter 8.2.6 "Configuring cyclic data" on page 437](#).

8 RMB_PLCopenEtherCAT_SoE.library

8.1 Introduction and overview



The basic function package PLC is sufficient for the library of the IndraLogic XLC/IndraMotion MLC 14VRS system.

Significance of the function block prefixes

The RMB_PLCopenEtherCAT_SoE library contains the function blocks communicating with the field bus axes.

- **Prefix MC_**
This prefix indicates that this function block is 100% PLCopen-compliant.
- **Prefix MB_**
This prefix indicates that this function block is executed according to the systematics of the PLCopen, deviates or supplements it or is not defined.
- **Prefix MP_**
This prefix is used for system-internally used function blocks.

The function blocks, functions, data types and constants are included in the following folders and Global Variable Lists:

- [Data types, page 438,](#)
- [Function blocks for single-axis control \(Motion_FBs\), page 439,](#)

8.2 IndraDrive – Required configurations

The operation of drives with function blocks of the RMB_PLCopenEtherCAT_SoE library, requires the following settings:

- Configuring the first secondary operation mode, see [chapter 8.2.1 "Configuring the operation modes \(e.g. with FWA MPB20\)" on page 430](#)
- Configuration of signal status word and signal control word, see [chapter 8.2.2 "Configuration of signal status word and signal control word" on page 430](#)
- Configuring process data, see ["Configuring process data" on page 432](#)
- Configuring cyclic data, see [chapter 8.2.6 "Configuring cyclic data" on page 437](#)

For configuration support of the drives, the RMB_PLCopenEtherCAT_SoE library contains two import files:

- **ParametersettingsIndraDrive.zip**
The parameter file contains the configurations for operation mode, signal status and control word and for cyclic data.
The file can be imported in existing drives, see [chapter 8.2.3 "Configuration using the parameter file" on page 431](#)
- **EtherCAT_EoS_IndraDriveSlave.zip**
The parameter file contains a pre-configured drive "Drive_XY" used as template to create new drives.
The file can be imported in the EtherCAT master, see [chapter 8.2.5 "IndraDrive slave – Import" on page 433](#)

The drive in the application has to be declared manually in both import files, see ["Declaration of the drive" on page 435](#)

8.2.1 Configuring the operation modes (e.g. with FWA MPB20)



The main operation mode is not changed.

If PLCopen function blocks control the drive, the first secondary operation mode is enabled automatically by the function blocks.

To configure the drive control of the first secondary operation mode, the following setting has to be selected: either "Drive-controlled positioning, encoder 1" or "Drive-controlled positioning lagless, encoder 1":

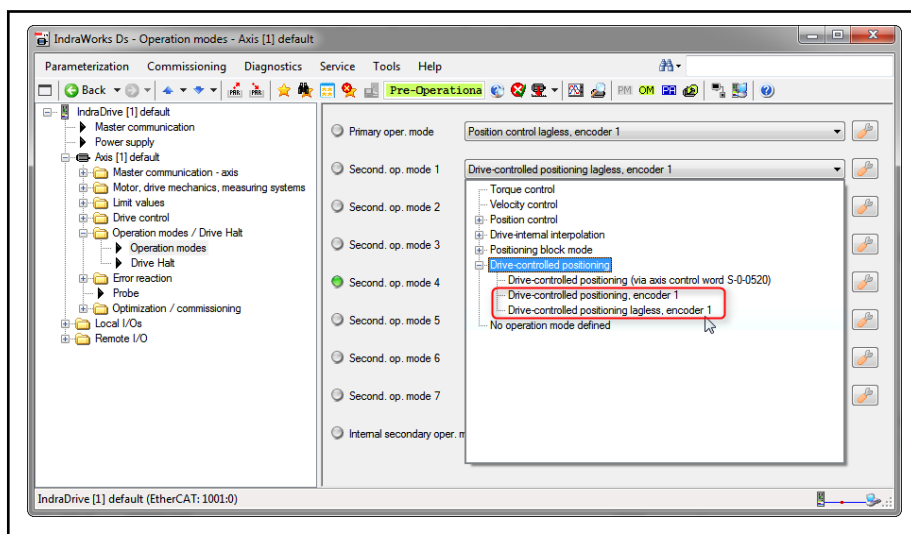


Fig. 8-1: Configuration of the first secondary operation mode - Drive control

8.2.2 Configuration of signal status word and signal control word

The configuration of the signal status word and the signal control word can either be selected via the parameter editor or via the drive wizard:

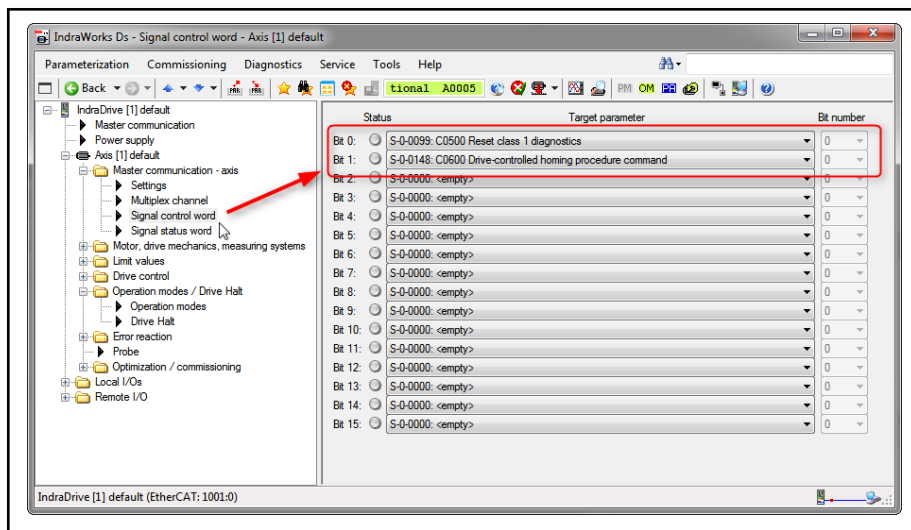


Fig. 8-2: Configuration of the "signal control word" via the drive wizard

Signal status word The following settings have to be selected in the signal status word:

- Assign bit 0 of parameter "S-0-0403" to parameter "S-0-0026" bit 0.
- Assign bit 0 of parameter "S-1-0419" to parameter "S-0-0026" bit 0.

Signal control word

The following settings have to be selected in the signal control word:

- Assign bit 0 of parameter "S-0-0099" to parameter "S-0-0027" bit 0.
- Assign bit 0 of parameter "S-0-0148" to parameter "S-0-0027" bit 1.

Configuration of signal status word and signal control word - Overview

Parameter name	Parameters				Parameters	Parameter name
Configuration list for signal status word	S-0-0026	Bit 0	↔	Bit 0	S-0-0403	Status of the actual position values
Configuration list for signal status word	S-0-0026	Bit 1	↔	Bit 0	S-0-0419	Acknowledgement of the positioning command value
Assignment list signal status word	S-0-0328	Bit 0	↔	Value = 0		
Assignment list signal status word	S-0-0328	Bit 1	↔	Value = 0		
Configuration list signal control word	S-0-0027	Bit 0	↔	Bit 0	S-0-0099	C0500 Reset class 1 diagnostics
Configuration list signal control word	S-0-0027	Bit 1	↔	Bit 0	S-0-0148	C0600 Drive-controlled homing procedure command
Assignment list signal control word	S-0-0329	Bit 0	↔	Value = 0		
Assignment list signal control word	S-0-0329	Bit 1	↔	Value = 0		

Tab. 8-1:

8.2.3 Configuration using the parameter file

The described configuration of the "first secondary operation mode" and the configuration of the "signal status word and signal control words" can optionally be selected using a parameter file.

A prepared parameter file is provided in the documentation for the "RMB_PLCCopenEtherCAT_SoE" library in the library manager:

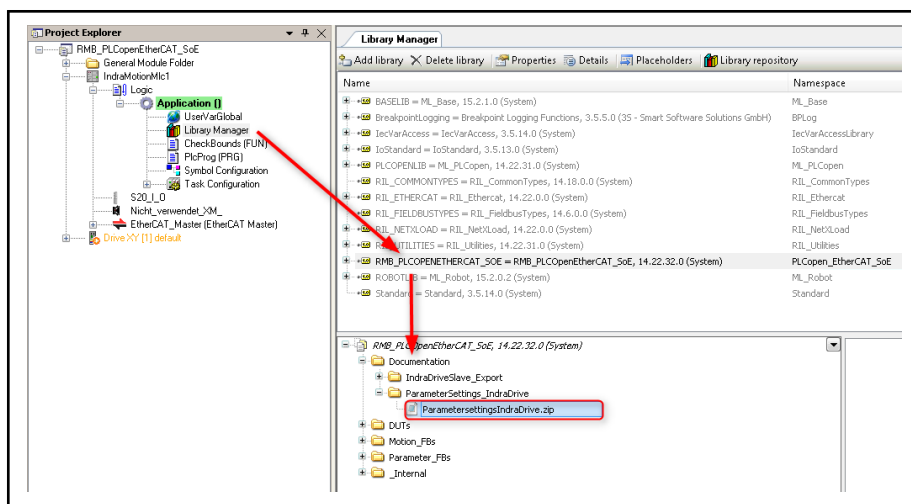


Fig. 8-3: Parameter file "ParameterSettingsIndraDrive.zip"



The parameter file is available in ".zip" file format.

To load the parameter file from the library manager to the drive, proceed as follows:



Prerequisite: To load the parameter to the drive, the drive has to be in online mode.

1. Double-click to open the parameter file with a suitable program such as 7 Zip or Notepad
2. Locally save the file with the extension ".par" (parameter file)
3. Select the drive node in the Project Explorer and open the context menu
4. Select the context menu command **Parameter ► Load**
5. Select the saved parameter file in the dialog and confirm the dialog with "Open"

The "Load parameter" dialog opens

6. Select the drive in which the parameter is to be loaded and confirm the dialog with load

The parameters are loaded to the drive

8.2.4 EtherCAT master – Configuring the drive

Configuring process data

The cyclic data required for communication are configured in "IndraDrive MPxXX EtherCAT Slave" of the EtherCAT master.

Double-click on the generic device editor of the affected drive in the Project Explorer:

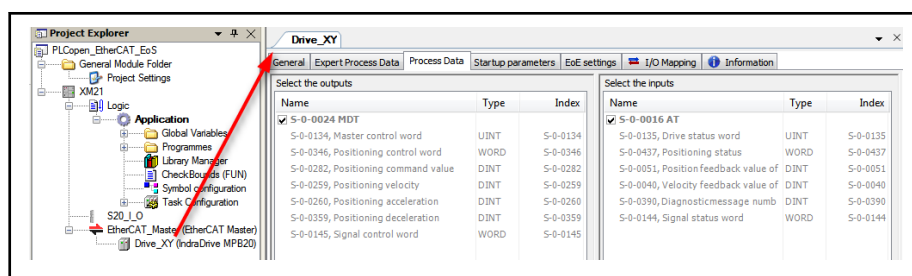


Fig. 8-4: Configure drive/tab "Process data"

Configuring the outputs

The following parameters have to be used in parameter S-0-0328 MDT:

Name	Type	Index
S-0-0134	Master control word	UINT
S-0-0346	Positioning control word	WORD
S-0-0282	Positioning command value	DINT
S-0-0259	Positioning velocity	DINT
S-0-0260	Positioning acceleration	DINT
S-0-0359	Positioning deceleration	DINT
S-0-0145	Signal control word	WORD

Tab. 8-2:

Configuring the inputs

The following parameters have to be used in parameter S-0-0016 AT:

Name	Type	Index
S-0-0135	Drive status word	UINT
S-0-0437	Positioning status	WORD
S-0-0051	Positioning feedback value of encoder 1	DINT
S-0-0040	Velocity feedback value of encoder 1	DINT
S-0-0390	Diagnostic message number	DINT
S-0-0144	Signal status word	WORD

Tab. 8-3:

8.2.5 IndraDrive slave – Import

Import "Drive_XY" drive

To facilitate the configuration of the drive, the "RMB_PLCoopenEtherCAT_SoE" library provides an import file containing a preconfigured drive.

As a result of the import, a drive with the designation "Drive_XY" is created below the EtherCAT master. The drive contains all required configurations to be operated with function blocks of the RMB_PLCoopenEtherCAT_SoE library. The import can be used for test purposes or as template to create new drives.

The import file "EtherCAT_EoS_IndraDriveSlave.zip" is provided in the documentation for the "RMB_PLCoopenEtherCAT_SoE" library in the library manager:

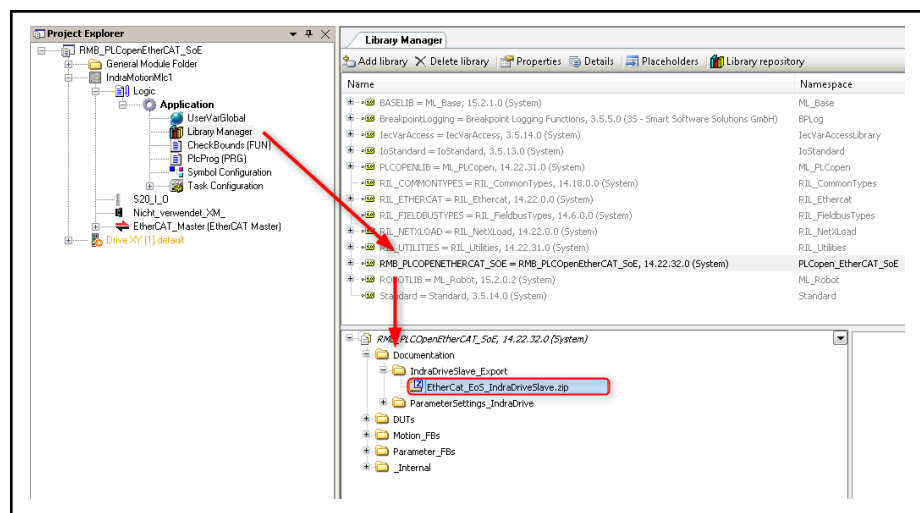


Fig. 8-5: Import file "EtherCAT_EoS_IndraDriveSlave.zip"



The parameter file is available in ".zip" file format.

To load the import file from the library manager to the EtherCAT master, proceed as follows:



Prerequisite: Log out the control to execute an import.

1. Double-click to open the import file with a suitable program such as 7 Zip
 2. Save import file locally (.iwx file)
 3. Select the EtherCAT master object in the Project Explorer and open the context menu
 4. Select the context menu command **Import...**
 5. Select the saved import file in the dialog and confirm the dialog with "Open"
- The "Import wizard" opens
6. Press "Finish" to confirm the dialog

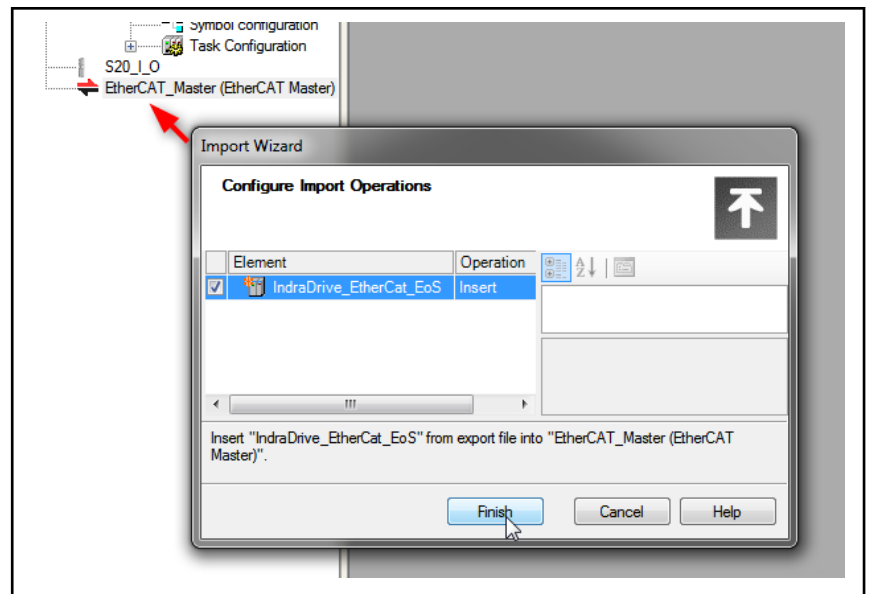


Fig. 8-6:

The drive "Drive_XY" is displayed below the EtherCAT master

Now the drive "Drive_XY" has to be declared in a Global Variable List (GVL) in the PLC application.

Declaration of the drive

1. Open the application object in the Project Explorer.
2. Add a Global Variable List to the application object by selecting the application.

Execute the context menu **Add ► Global Variable List...**



If a Global Variable List is available, any amendments can also be executed in this list.

3. Insert this code in the declaration part:

Program:

```
VAR_GLOBAL
  Drive_XY : PLCopen_EtherCAT_SoE.AXIS_REF :=
    (AxisNo := OBJECTS.FieldBusAxis_1,
     CntrlNo := CONTROLS.LOCAL_CNTRL);
END_VAR
```

The declaration is completed.

Rename Drive_XY

After the drive has been declared in the Global Variable List and after a compilation run has been completed, the drive name can be changed project-comprehensively by using the "Refactoring" command in the declaration part of the IndraLogic editor:

1. Set the cursor in the declaration of the drive "Drive_XY".
Open the context menu and select the command: **Rename Refactoring ► 'Drive_XY'**

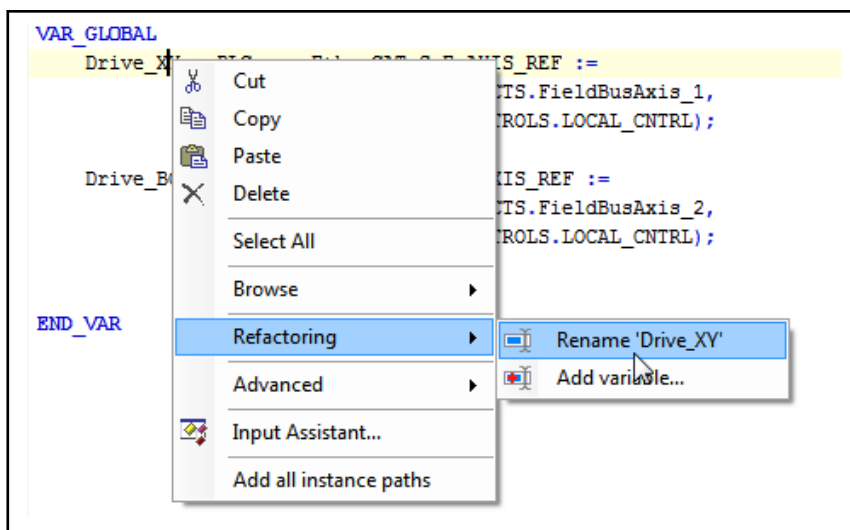


Fig. 8-7:

The dialog "Rename" opens.

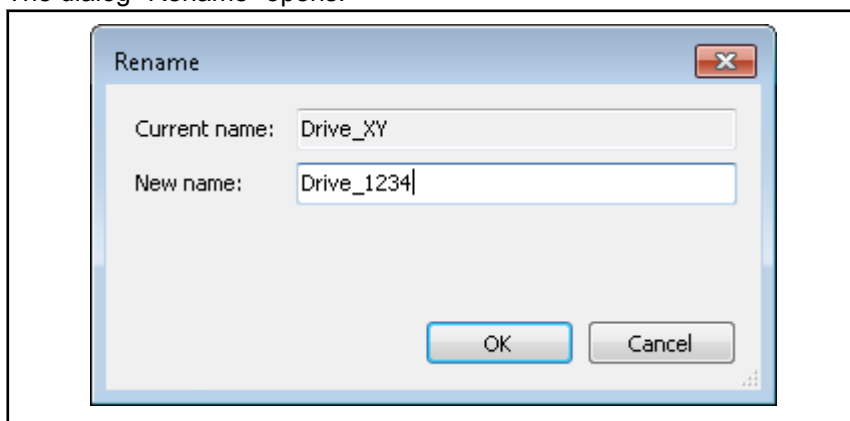


Fig. 8-8:

2. Enter a new name and confirm the dialog with "OK".

The dialog "Refactoring" opens.

The dialog shows all locations of the previous name in the project.

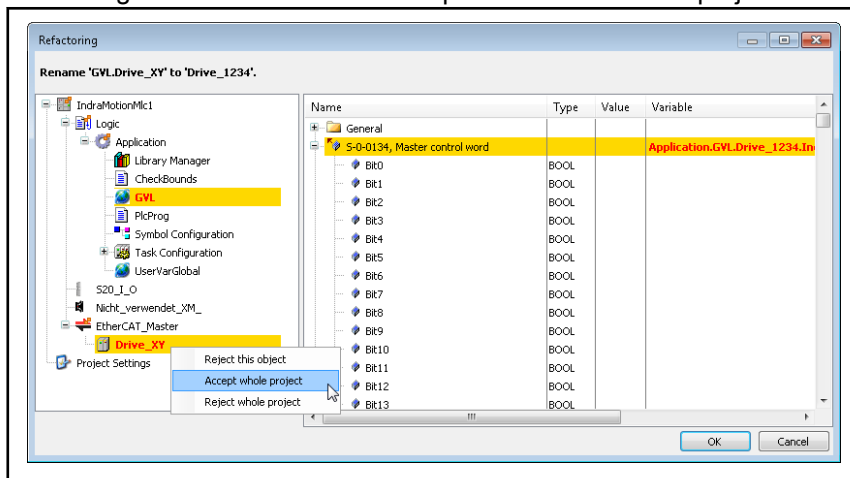


Fig. 8-9:

3. Click in the left window and open the context menu with left-click.
Select the option "Accept whole project" as shown in the previous screenshot.
4. Confirm the dialog with "OK"
The drive name is renamed in the project.

8.2.6 Configuring cyclic data

To configure the cyclic data, the input and output parameters ("S-0-0016 AT" and "S-0-0328 MDT") have to be connected to the IndraDrive function block via I/O mapping.

Double-click on the generic device editor of the affected drive by double-clicking in the Project Explorer and call the tab "I/O mapping".

Signal control word

The parameter "S-0-0145 Signal control word" has to be configured as follows:

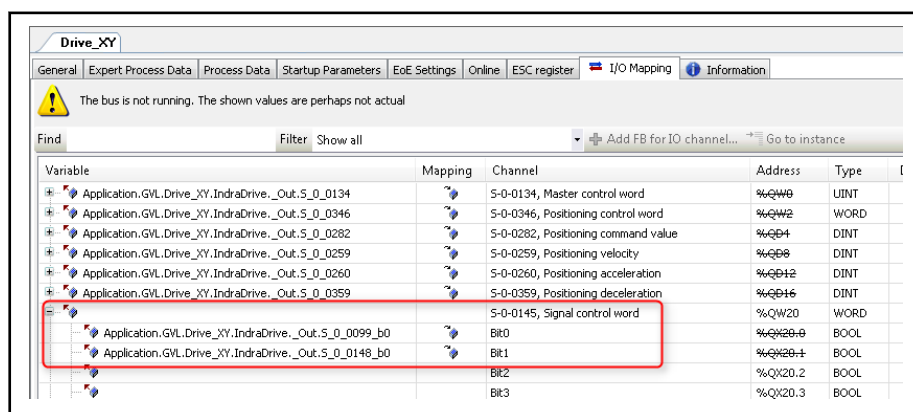


Fig. 8-10:

Signal status word

The parameter "S-0-0144 Signal status word" has to be configured as follows:

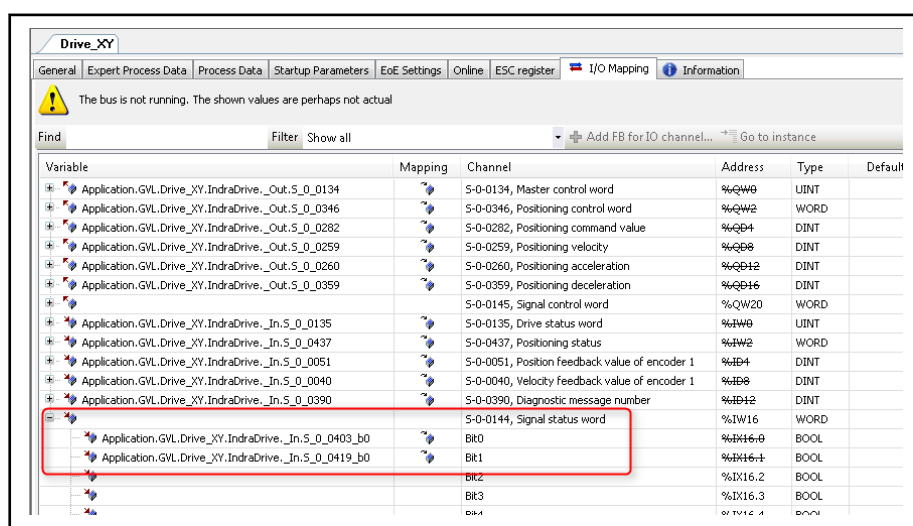


Fig. 8-11:

8.3 Data types

8.3.1 Data types, general

The folder contains data types that support working with the library elements of the "RMB_PLCopenEtherCAT_SoE.library".



Elements other than the elements described may not be used due to the side effects, including malfunctions in the motion sequence of the axes!

8.3.2 AXIS_REF

Library	RMB_PLCopenEtherCAT_SoE
Brief description	The structure contains information on the desired control and on the object controlled by this control (axis).



Information is transferred via **VAR_IN_OUT** to the respective PLC function block defined in the "RMB_PLCopenFieldBus.library".



Only for field bus drives/axes connected to the EtherCAT field bus.

Library	Range
RMB_PLCopenFieldBus	Data types

Tab. 8-4: Library assignment

Type definition

```
(* Structure of PLCopen element AXIS_REF. The AXIS_REF is a structure that contains *)
(* information on the corresponding Object. *)
(* It is used as a VAR_IN_OUT in all Fieldbus Motion Control Function Blocks. *)
TYPE AXIS_REF :
STRUCT
  CntrlNo: CONTROLS := LOCAL_CNTRL; (* control reference number *)
  AxisNo: OBJECTS := FieldBusAxis_1; (* object reference number *)
  IndraDrive: MP_Drive; (* Achsreferenznummer, Init: FieldBusAxis_1 *)
END_STRUCT
END_TYPE
```

Functional description

The structure **AXIS_REF** (RIL_CommonTypes) contains address information on the desired control and on the object controlled by this control (axis number).

In contrast to the **AXIS_REF** structure (RIL_CommonTypes) supporting the addressing of Sercos-based IndraDrives, the **AXIS_REF** structure (RMB_PLCopenFieldbus) is extended by two additional elements (**MP_Drive** and **MP_DriveParameter**). These two elements are only used by the system.

The axis number can differ from the station number set at the drive.

The Global Variable List "MLCVarGlobal" with all Sercos-based axes is automatically created and maintained for Sercos-based axes.

The global variable list "GVL_FieldBusDrives" with all field bus drives has to be manually created and maintained for the field bus drives.

The two first elements of each drive "CntrlNo" and "AxisNo" are preassigned in this list. The elements "MP_Drive" and "MP_DriveParameter" remain unchanged. Therefore, they do not have to be included in the list.



Address information is transferred as described in the following:

Sercos-based axes

- Information was transferred up to 13VRS via **VAR_IN_OUT**.
- In 14VRS and higher, information is transferred via **VAR_IN_OUT CONSTANT**.

Field bus axes

- Information is always transferred via **VAR_IN_OUT**.

Global Variable List "GVL_FieldBusDrives", example

```
VAR GLOBAL
Drive_X: PLCopen_EtherCAT_SoE.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=1001); // IndraDrive X - example
Drive_Y: PLCopen_EtherCAT_SoE.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=1002); // IndraDrive Y - example
END_VAR
```

Namespace specification PLCopen_EtherCAT_SoE

The namespace specification "PLCopen_EtherCAT_SoE." for a field bus drive has to be prefixed.

Declaration AXIS_REF

```
Drive_A: PLCopen_EtherCAT_SoE.AXIS_REF: =
```

The data types are in the RMB_PLCOpenEtherCAT_SoE.library, folder "Data types", element "AXIS_REF".

Assignment CntrlNo

```
(CntrlNo: = LOCAL_CNTRL
```

The enumeration value LOCAL_CNTRL is contained in RIL_Common-Types.lib, folder "Data types of logical addresses", CONTROLS.

The field bus function blocks always refer to the local PLC, that is **LOCAL_CNTRL**.

Assignment AxisNo

```
AxisNo: =1001..1002...);
```

.

For field bus drives, specifications on the control and the axis number are used for the error output (ErrorIdent).

A functionality is not combined with the assignment of the axis number.



It is recommended to compare the station number and the axis number.

8.4 Function blocks for single-axis control (Motion_FBs)

8.4.1 Motion_FBs, general information



The function blocks named in the following are used for the motion control.

State diagram

The following diagram defines the behavior of the MotionControl axes from the user's point of view as well as the reaction on exceptions in the programs.

In principle, motion commands are given and processed sequentially, even if the PLC is capable of parallel processing. These commands follow the state diagram of the MotionControl axes.

Each axis is in exactly one of the states defined (see the state diagram below). Each motion command represents a state transition for the axis concerned. The sequence of these transitions describes the entire behavior of the axis.

The chart focuses on the single axis.



The field bus drives only operate as single-axis function blocks!

The following function blocks with "management character" do not influence the state diagram:

- [MC_ReadActualPosition](#), page 460,
- [MC_ReadAxisError](#), page 461,
- [MC_ReadStatus](#), page 463,

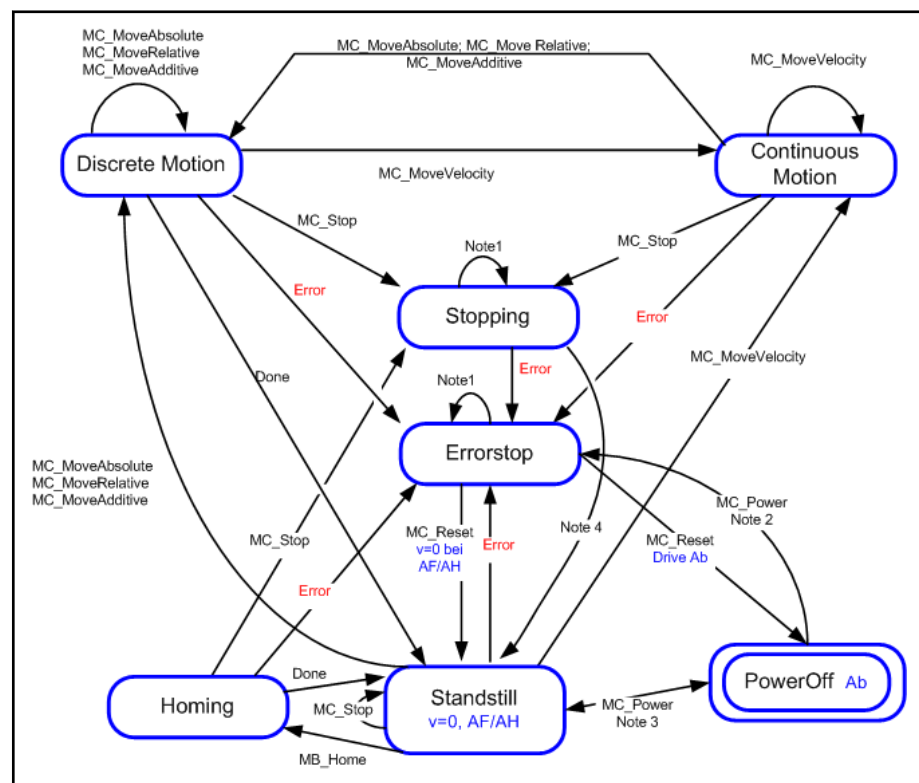


Fig. 8-12: MotionControl state diagram of a field bus axis

Legend

(Note 1) - All function blocks can be called without being processed except for MC_Reset and Error. They create the respective transitions to Standstill or ErrorStop.

(Note 2) - Power.Enable = TRUE and error in the axis.

(Note 3) - Power.Enable = TRUE and error-free axis.

(Note 4) - MC_Stop.Done AND NOT MC_Stop.Execute

An error in the drive results in a power switch off after a certain period, depending on the programmed error reaction. However, the state machine remains in "ErrorStop" and when the power is switched off, "MC_Reset" goes into power-off state.

	Function
MC_Power, page 441	Controls power on (on or off).
MC_Reset, page 442,	Implements the transition from the "ErrorStop" state to the "StandStill" state by resetting all axis-internal errors.
MC_Stop, page 444	Implements a controlled motion stop and transfers the axis to the "Stopping" state. This stops any function block processing. When standstill is reached, the "Done" output is set. Another command can only be started after the axis reached the "Stopping" state.
MC_Stop, page 445,	Implements a controlled motion stop and transfers the axis to the "Stopping" state. This stops any function block processing. When standstill is reached, the "Done" output is set. When "ExecuteLock" is TRUE, the axis is in the "Stopping" state and cannot be moved by another command.
MB_Home, page 447	Controls axes with incremental encoder when executing the "search home" sequence. The details of the sequence must be set via axis parameters. "Homing" ends in standstill.
MC_MoveAbsolute, page 448	The function block copies the absolute predetermined "position" to the "target position" and approaches it.
MC_MoveAdditive, page 451	The function block adds the "distance" to the "target position" and approaches it.
MC_MoveRelative, page 453	The function block determines the new "target position" by adding the "distance" to the previous "actual position" and approaches it.
MC_MoveVelocity, page 456	Controls an "endless" motion at the respectively defined velocity. To end the motion, the function block instance has to be interrupted by another function block instance which issues a new command.

Tab. 8-5: Function blocks of the "Motion_FBs" folder

8.4.2 MC_Power

Brief description

The function block controls the power adding (on or off).

For **each real axis, a separate instance** of this function block is required.

Enabling the function block is a prerequisite for every motion, acceleration or deceleration.

A special situation is given for the function block [MC_Stop, page 444](#). The position is kept active by the drive. Thus, the MC_Power function block cannot be switched off when MC_Stop is active.



Together with a virtual axis, the function block results in an error message.

Library	Range
RMB_PLCOpenFieldBus	Motion_FBs
RMB_PLCOpenEtherCAT_SoE	

Tab. 8-6: Library assignment

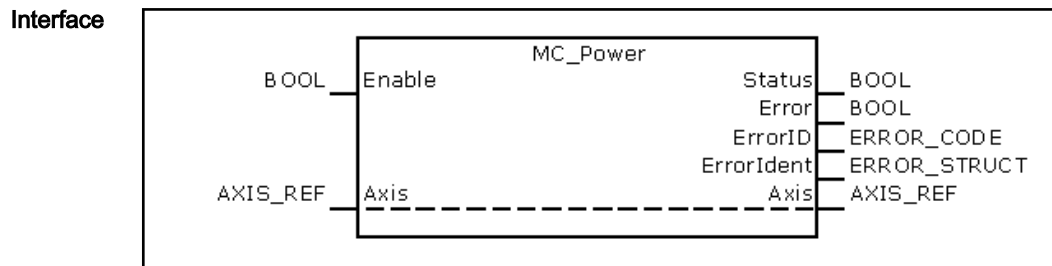


Fig. 8-13: MC_Power function block

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
VAR_INPUT	Enable	BOOL	When "Enable" is TRUE, power is added
VAR_OUTPUT	Status	BOOL	Actual state of power adding
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 8-7: Interface of the MC_Power function block

- Troubleshooting:**
- An error message is issued if the "Status" is still FALSE after a timeout at the set "Enable". This indicates that a hardware problem occurred when power was added.
 - An error message is also issued if it is tried to disconnect the power in an axis state in which this is not permitted.
 - If adding on power fails or malfunctions while the axis is moving, an "ErrorStop" is generated for the axis.

In case of a configuration error, the following is displayed

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

8.4.3 MC_Reset

Brief description This function block implements the transition from the "ErrorStop" state to the "StandStill" state by resetting all axis-internal errors.



The outputs of the function block instances that indicated the error(s) are not affected.

If there is no error present, triggering MC_Reset has no effect.

Library	Range
RMB_PLCOpenFieldBus	Motion_FBs
RMB_PLCOpenEtherCAT_SoE	

Tab. 8-8: Library assignment

Interface description

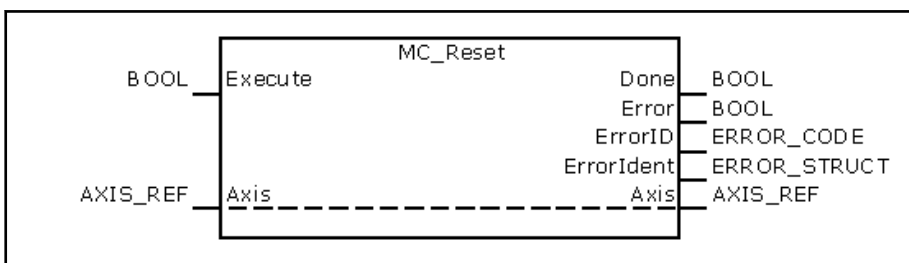


Fig. 8-14: MC_Reset function block

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
VAR_OUTPUT	Done	BOOL	Errors are reset
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 8-9: Interface of the MC_Reset function block

Signal-time diagram

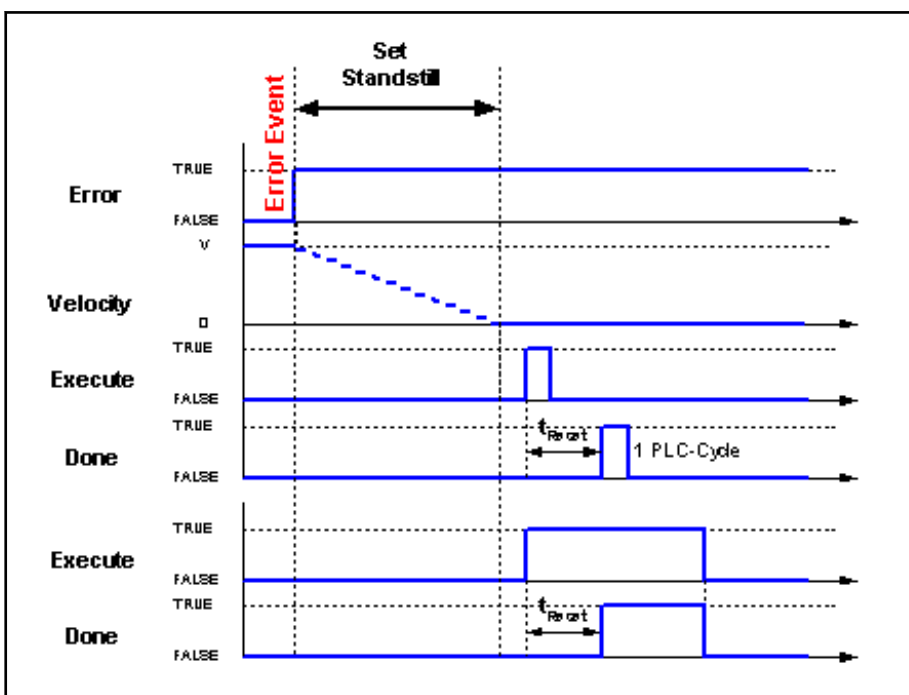


Fig. 8-15: Time diagram of MC_Reset

The function block uses the parameter "S-0-0099", C0500 Reset state class 1.

If there is no error present, triggering MC_Reset has no effect.

A clear error command "C5" is issued in any case at the drive.

Troubleshooting

The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

8.4.4 MC_Stop

Brief description

This function block implements a controlled motion stop and transfers the axis to the "Stopping" state. This stops any function block processing. When standstill is reached, the "Done" output is set.

The position is kept active.

 When "Execute" is TRUE, the axis remains in state "Stopping".

Another command can only be started after the axis reached the "Stopping" state.

In the "Stopping" state, it is not possible to switch off the axis power (MC_Power, page 441.). If "Enable" of the MC_Power instance is disabled, its "status" nevertheless remains switched on until "Execute" of the MC_Stop instance is also disabled.

 The "MC_Stop" function block is not supported for IndraDrive with active operating stop (P-0-0558 bit 0 = 1).

The function block never sets the "Done" output to TRUE.

Library	Range
RMB_PLCOpenFieldBus	Motion_FBs
RMB_PLCOpenEtherCAT_SoE	

Tab. 8-10: Library assignment

Interface description

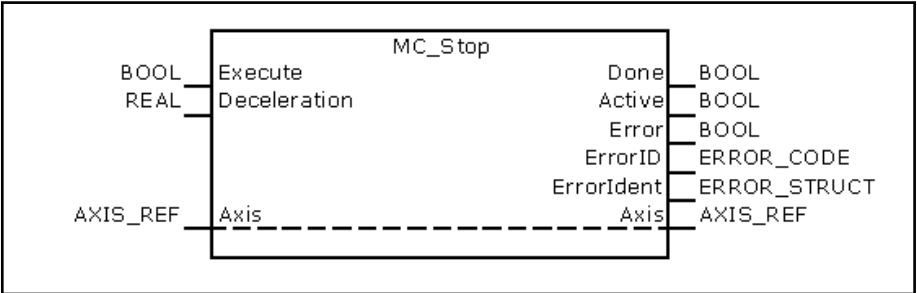


Fig. 8-16: MC_Stop function block

	Name	Type	Comment ¹⁾
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge. The function block cannot be aborted until the "Stopping" state is reached.
	Deceleration	REAL	Deceleration (always greater than 0, otherwise no "Stopping").
VAR_OUTPUT	Done	BOOL	Standstill reached
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 8-11: Interface of the MC_Stop function block

Signal-time diagram

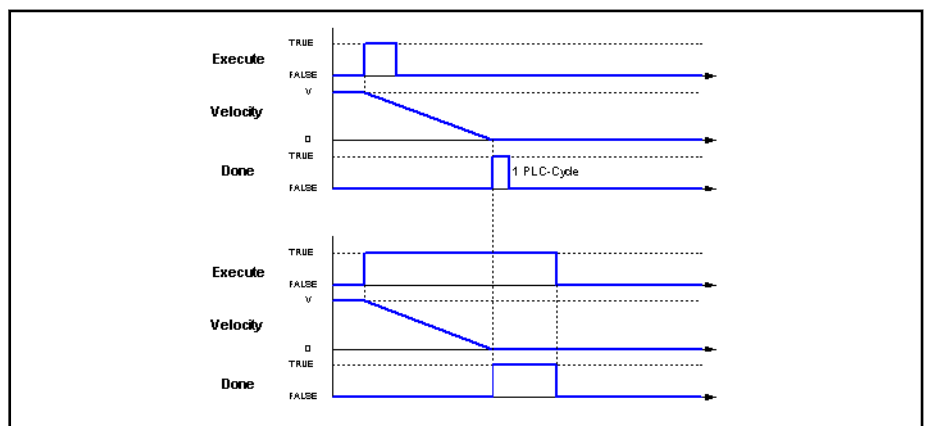


Fig. 8-17: Time diagram of MC_Stop

Troubleshooting

The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

8.4.5 MB_Stop

Brief description

This function block implements a controlled motion stop and transfers the axis to the "Stopping" state. This stops any function block processing.

When standstill is reached, the "Done" output is set.

The position is kept active.

¹⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.



When "ExecuteLock" is **TRUE**, the axis is in the "Stopping" state and cannot be moved by another command.

In contrast to "[MC_Stop](#), page 444," the function block can be interrupted by another function block as soon as "ExecuteLock" is **FALSE**.

In "Stopping" state, it is not possible to switch off the axis power ([MC_Power](#), page 441,). If "Enable" of the MC_Power instance is deactivated, its "Status" nevertheless remains switched on until the "ExecuteLock" signal of the "MB_Stop" instance is also deactivated.

Library	Range
RMB_PLCOpenFieldBus	Motion_PLCOpen
RMB_PLCOpenEtherCAT_SoE	

Tab. 8-12: Library assignment

Interface description

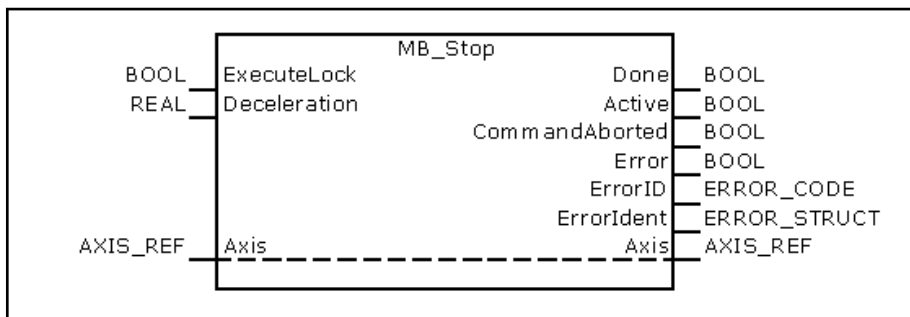


Fig. 8-18: MC_Stop FB

	Name	Type	Comment ²⁾
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
VAR_INPUT	ExecuteLock	BOOL	It starts the stopping of the axis with the rising edge. As long as "ExecuteLock" = TRUE, a motion cannot be started.
	Deceleration	REAL	Deceleration (always greater than 0, otherwise no "Stopping").
VAR_OUTPUT	Done	BOOL	Standstill is reached.
	Active	BOOL	Command is enabled. Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command is aborted by another command.
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 8-13: Interface of the MB_Stop FB

²⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

Signal-time diagram

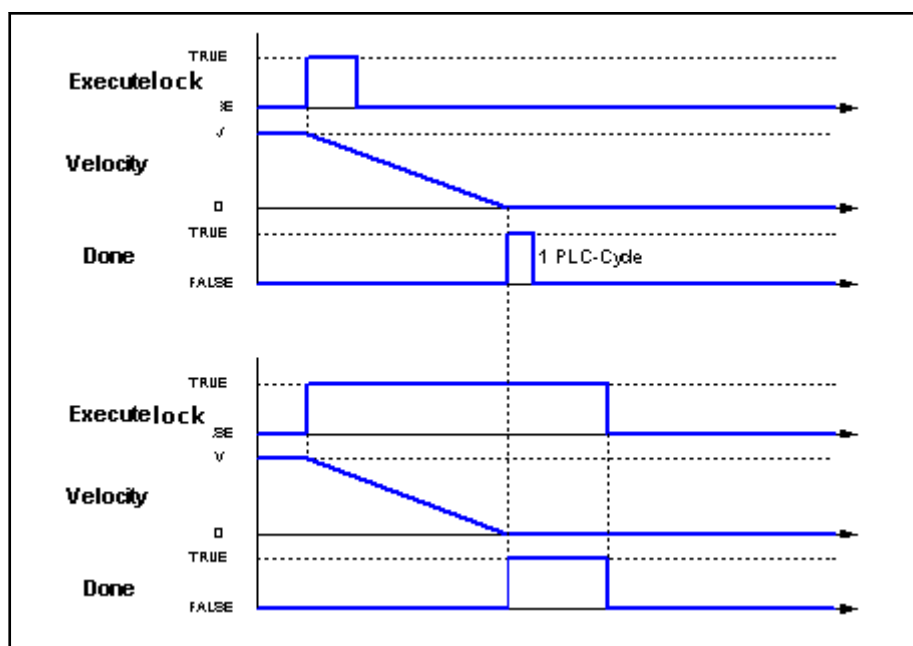


Fig. 8-19: Time diagram of MB_Stop

Troubleshooting

The function block generates error messages only at a positive edge or if TRUE is set at the "ExecuteLock" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

8.4.6 MB_Home

Brief description

The function block controls **axes with incremental encoder** when executing the "search home" sequence. The details of the sequence must be set via axis parameters. "Homing" ends in standstill.

Library	Range
RMB_PLCOpenFieldBus	Motion_FBs
RMB_PLCOpenEtherCAT_SoE	

Tab. 8-14: Library assignment

Interface description

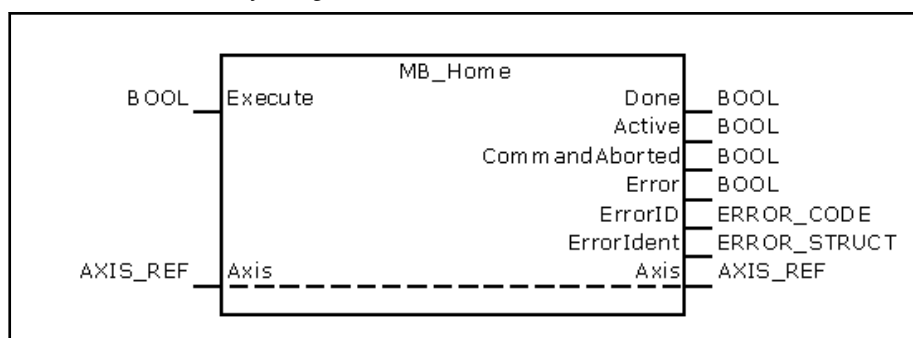


Fig. 8-20: MB_Home function block

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
VAR_OUTPUT	Done	BOOL	Homing completed, velocity 0
	Active	BOOL	Processing of data after preprocessing running
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 8-15: Interface of the MB_Home function block

Signal-time diagram

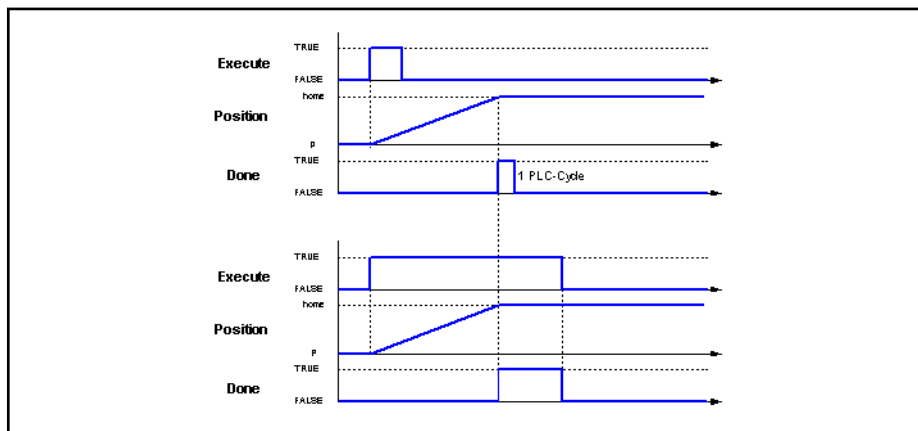


Fig. 8-21: Time diagram of MB_Home

Troubleshooting

The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

8.4.7 MC_MoveAbsolute

Brief description

The function block copies the absolutely specified "position" to the "target position" and approaches it.



If this function block requires power added before via an instance of the [MC_Power](#), page 441, (for real axes).

Library	Range
RMB_PLCOpenFieldBus	Motion_FBs
RMB_PLCOpenEtherCAT_SoE	

Tab. 8-16: Library assignment

Interface description

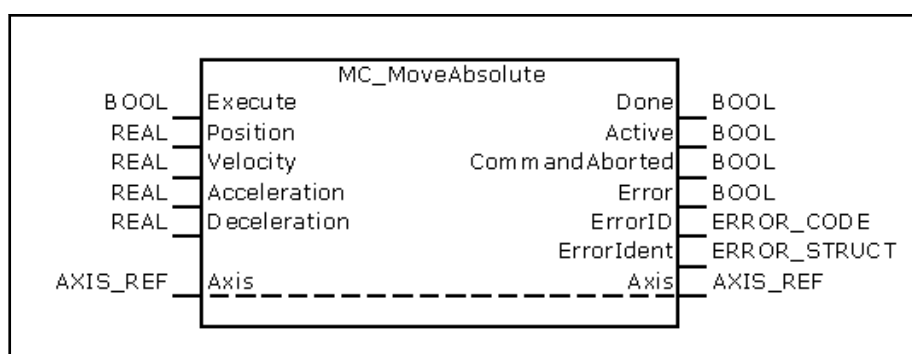


Fig. 8-22: MC_MoveAbsolute function block

	Name	Type	Comment ³⁾
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
	Position	REAL	Target position for the motion (+/-)
	Velocity	REAL	Maximum velocity value (always +) (does not necessarily have to be reached).
	Acceleration	REAL	Acceleration (always +)
	Deceleration	REAL	Deceleration (always +)
VAR_OUTPUT	Done	BOOL	Desired position reached
	Active	BOOL	Processing of data after preprocessing running
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 8-17: Interface of the MC_MoveAbsolute function block

Implementation example:

The following figure following shows two examples of cooperation of two "MC_MoveAbsolute" function block instances:

1. The left side of the time diagram shows the case in which Instance2 is called after Instance1. After Instance1 has implemented the position 6,000 (velocity 0), its "Done" output enables Instance2 with the target position 10,000.
2. The right side of the time diagram shows how Instance2 is already enabled while Instance1 is still running. The motion of Instance1 is stopped and aborted by the "Test" signal. Instance2 directly approaches its target position 10000 even though position 6000 has not yet been reached.

³⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

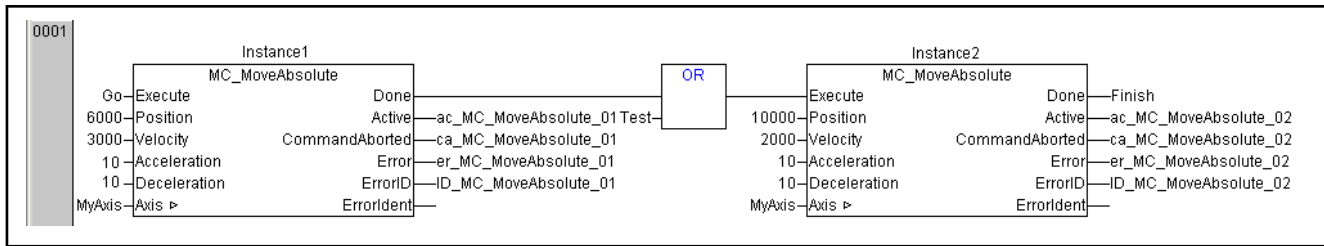


Fig. 8-23: Two instances of MC_MoveAbsolute

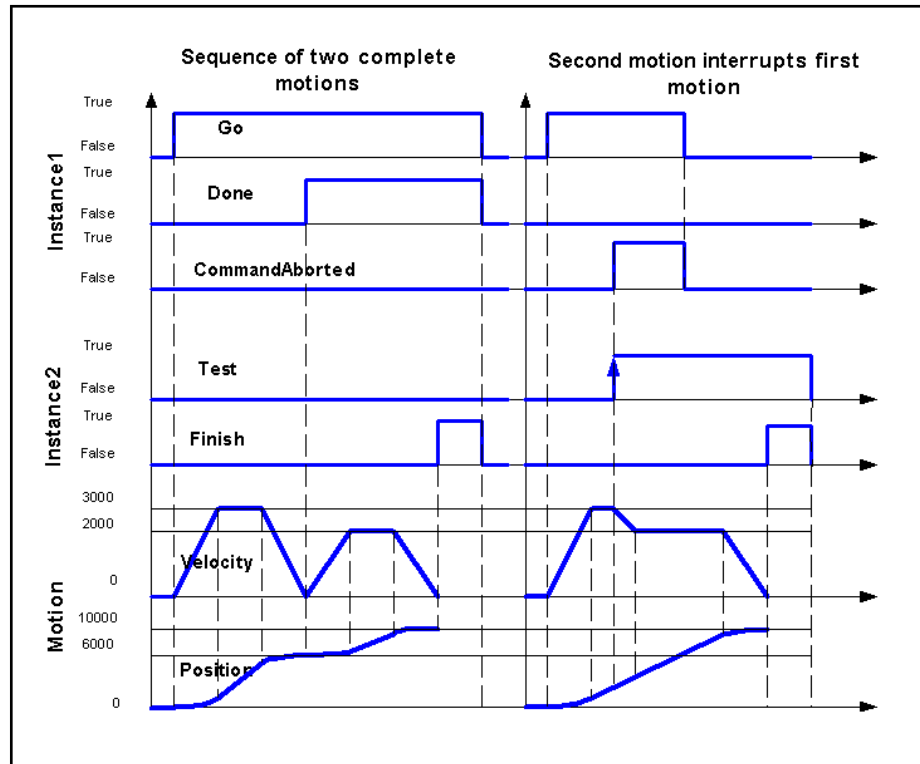


Fig. 8-24: Timing diagram: Two instances of MC_MoveAbsolute

- Parameters used** The function block determines the new target position by applying the command position and approaching it.
- Special function:** The function blocks "MC_MoveAbsolute", "MC_MoveAdditive" and "MC_MoveRelative" accept a velocity specification of $V_{com} = 0$.
When these function blocks ($V_{com} = 0$) are called, the axis is stopped. Positioning is only interrupted. The target position remains in the axis.
If MC_MoveAdditive ($V_{com} \neq 0$) is called subsequently, the remaining distance is traversed. The target position is only overwritten when another Motion function block is called.
- Troubleshooting** The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

Program:

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

8.4.8 MC_MoveAdditive

Brief description The function block adds the "distance" to the "target position" and approaches it.



If this function block requires power added before via an instance of the [MC_Power](#), page 441, (for real axes).

Library	Range
RMB_PLCOpenFieldBus	Motion_FBs
RMB_PLCOpenEtherCAT_SoE	

Tab. 8-18: Library assignment

Interface description

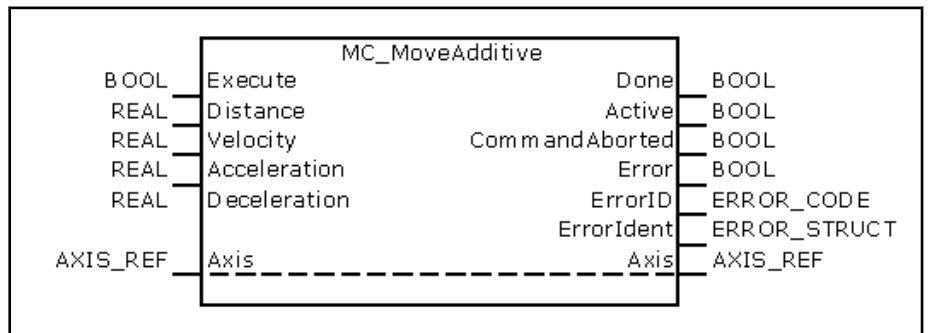


Fig. 8-25: MC_MoveAdditive function block

	Name	Type	Comment ⁴⁾
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
	Distance	REAL	Relative distance for the motion (+/-)
	Velocity	REAL	Maximum velocity value (always +) (does not necessarily have to be reached)
	Acceleration	REAL	Acceleration (always +)
	Deceleration	REAL	Deceleration (always +)
VAR_OUTPUT	Done	BOOL	Desired position reached
	Active	BOOL	Processing of data after preprocessing running
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 8-19: Interface of the MC_MoveAdditive function block

Implementation example: The figure below shows two examples of two combined function block instances while the axis is in "Discrete Motion State":

⁴⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.



Two different function block types are used:

- Instance1: MC_MoveAbsolute
- Instance2: MC_MoveAdditive.

1. The left part of the time diagram shows a case in which the second function block is started after the first one. The first function block reaches the distance 6,000 (at a velocity of 0). Subsequently, the "Done" output causes the second function block to travel the distance up to 10,000.
2. The right part of the time diagram shows a case in which the second function block is already started while the first function block is still active. In this case, the first motion is stopped and aborted by the "Test" signal at constant velocity. The second function block adds its distance of 4,000 to the originally commanded position of 6,000 and moves the axis to the resulting position 10,000 at a velocity specified at the second function block.

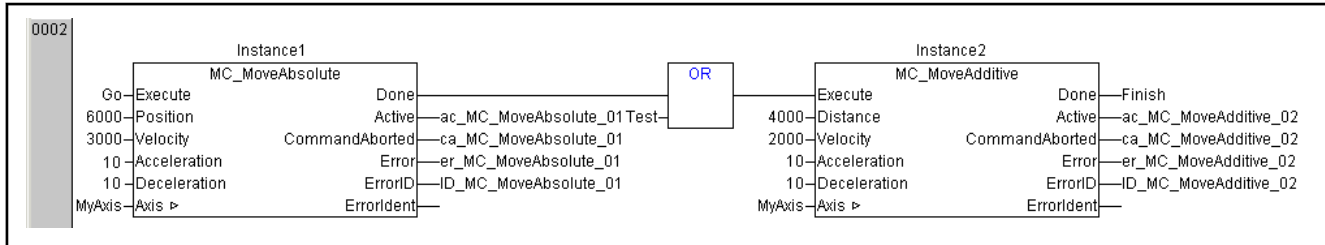


Fig. 8-26: Instances of MC_MoveAbsolute and MC_MoveAdditive

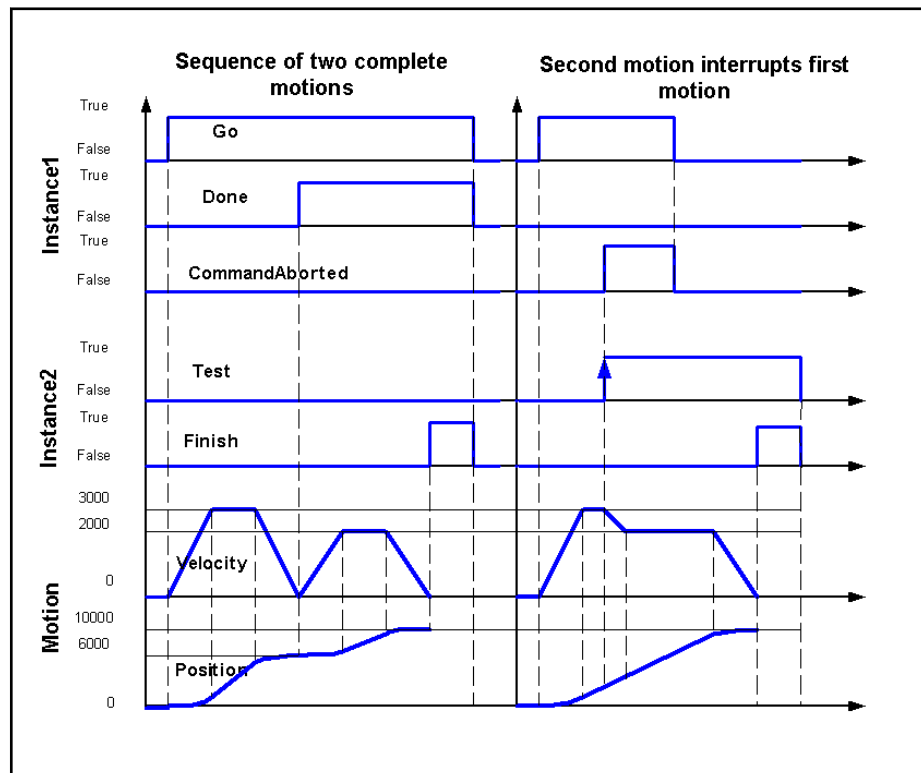


Fig. 8-27: Time diagram: Instances of MC_MoveAbsolute and MC_MoveAdditive

Parameters used

The function block determines the new target position by adding the distance to the previous target position and approaching it.



The distance is specified relative to the actual position.

Special function: The function blocks "MC_MoveAbsolute", "MC_MoveAdditive" and "MC_MoveRelative" accept a velocity specification of Vcom = 0.

When these function blocks (Vcom = 0) are called, the axis is stopped. Positioning is only interrupted. The target position remains in the axis.

If MC_MoveAdditive (Vcom ≠ 0) is called subsequently, the remaining distance is traversed. The target position is only overwritten when another Motion function block is called.

Troubleshooting The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

Program:

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F RELATED TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

8.4.9 MC_MoveRelative

Brief description The function block determines the new "target position" by adding the "distance" to the previous "actual position" and approaches it.



If this function block is used, power must have been added before via an instance of the [MC_Power, page 441](#), (for each real axes).

Library	Range
RMB_PLCOpenFieldBus	Motion_FBs

Tab. 8-20: Library assignment

Interface description

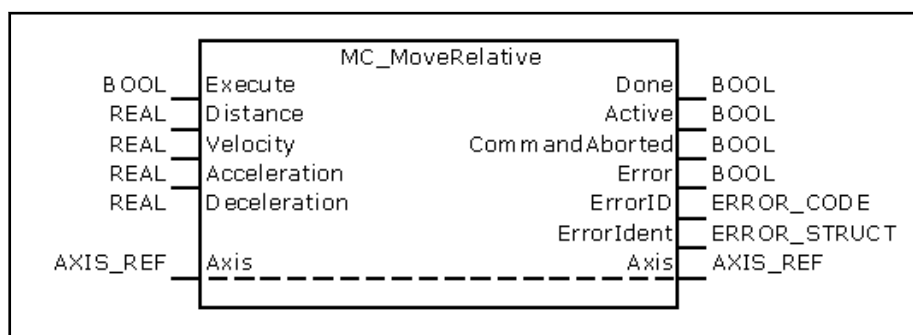


Fig. 8-28: MC_MoveRelative function block

	Name	Type	Comment ⁵⁾
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
	Distance	REAL	Relative distance for the motion (+/-)

⁵⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

	Name	Type	Comment ⁵⁾
	Velocity	REAL	Maximum velocity value (always +) (does not necessarily have to be reached)
	Acceleration	REAL	Acceleration (always +)
	Deceleration	REAL	Deceleration (always +)
VAR_OUTPUT	Done	BOOL	Desired position reached
	Active	BOOL	Processing of data after preprocessing running
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 8-21: Interface of the MC_MoveRelative function block

Signal-time diagram

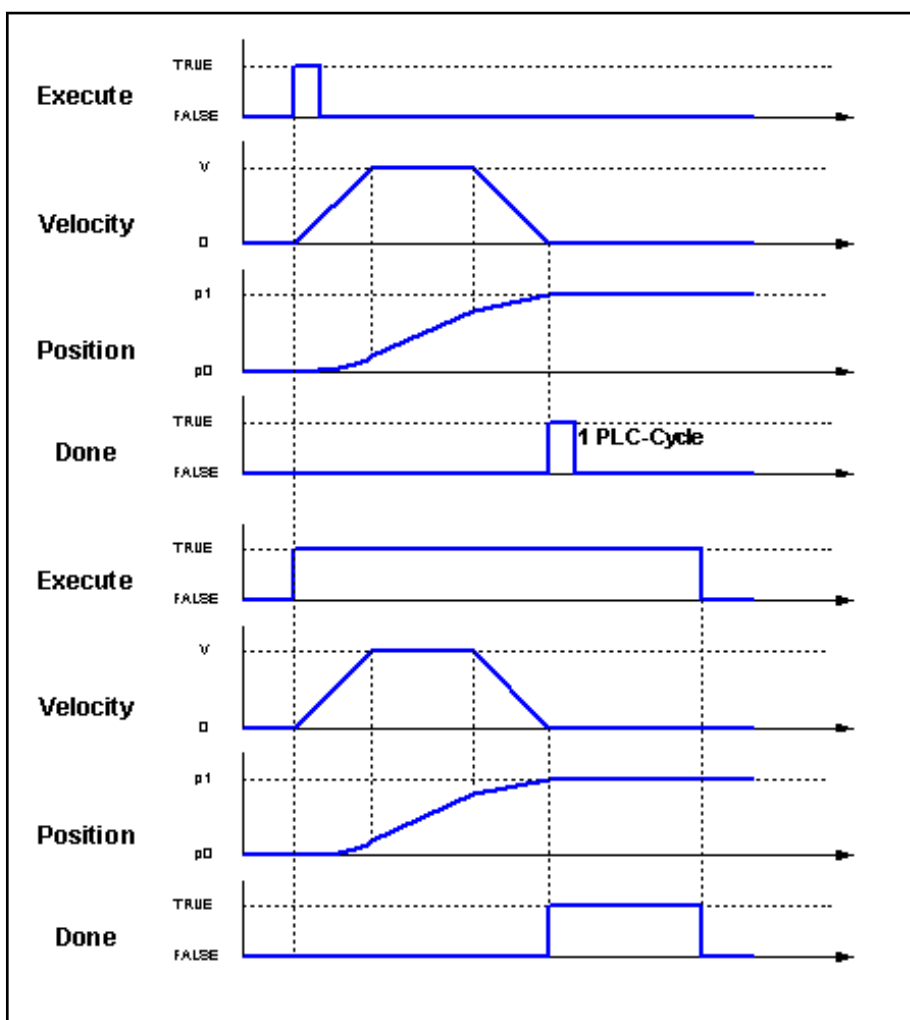


Fig. 8-29: Time diagrams for MC_MoveRelative

⁵⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

Implementation Example

The following example shows the combination of two MC_MoveRelative function block instances.

1. The left part of the time diagram shows a case in which Instance2 is called after Instance1 was processed. After Instance1 has reached the distance 6,000 (velocity 0), its "Done" output uses Instance2 to cause a further motion by 4,000 to reach the total distance of 10,000.
2. The right side shows how Instance2 is already activated by the "Test" signal while Instance1 is still running. The first motion is aborted during the constant velocity of Instance1. Instance1 adds its distance of 4000 to the current position (3250). The axis approaches the resulting position of 7,250.

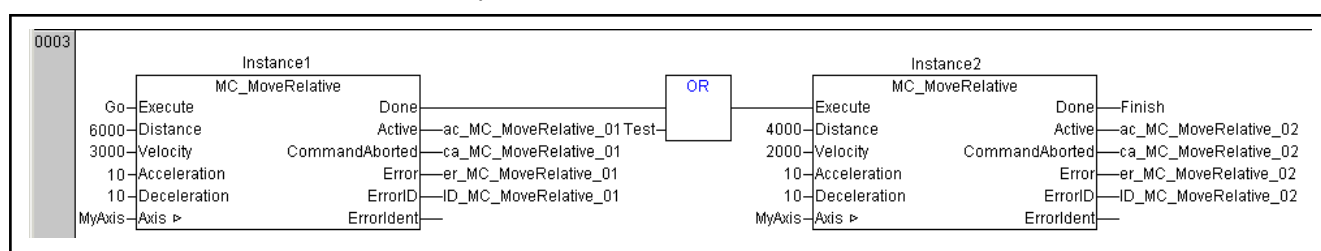


Fig. 8-30: Two instances of MC_MoveRelative

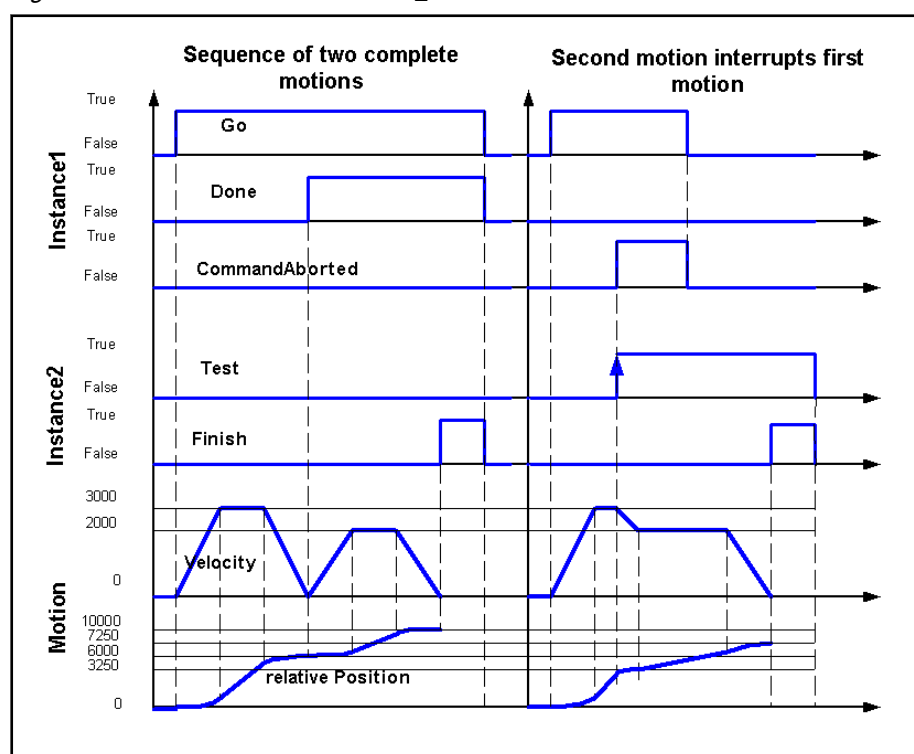


Fig. 8-31: Timing diagram: Two instances of MC_MoveRelative

Parameters used

The function block determines the new "target position" by adding the "distance" to the previous "actual position" and approaches it.



The distance is specified relative to the actual position.

Special function:

The function blocks "MC_MoveAbsolute", "MC_MoveAdditive" and "MC_MoveRelative" accept a velocity specification of Vcom = 0.

When these function blocks (Vcom = 0) are called, the axis is stopped. Positioning is only interrupted. The target position remains in the axis.

If MC_MoveAdditive (Vcom $\neq$ 0) is called subsequently, the remaining distance is traversed. The target position is only overwritten when another Motion function block is called.

Troubleshooting The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

Program:

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

8.4.10 MC_MoveVelocity

Brief description This function block controls an "endless" motion at the respectively defined velocity. To end the motion, the function block instance has to be interrupted by another function block instance which issues a new command.



If this function block is used, power must have been added before via an instance of the [MC_Power](#), [page 441](#), (for each real axes).

The "InVelocity" signal is reset if the function block instance is interrupted by another function block instance (CommandAborted) or if the edge of its "Execute" input is falling.



If "Execute" is disabled, the drive keeps running on the same velocity.

Library	Range
RMB_PLCoPenFieldBus	Motion_FBs

Tab. 8-22: Library assignment

Interface description

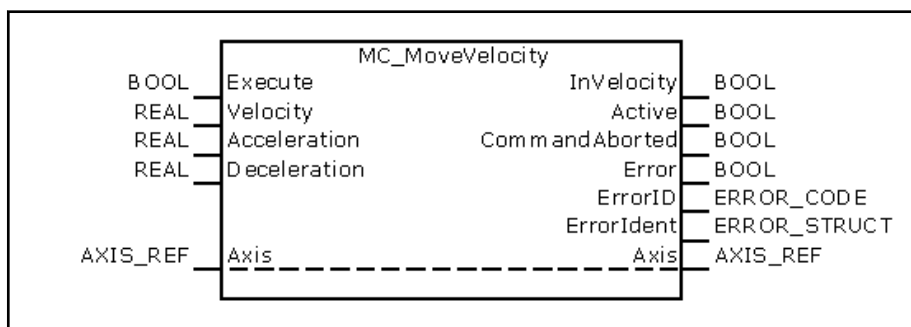


Fig. 8-32: MC_MoveVelocity function block

	Name	Type	Comment ⁶⁾
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
VAR_INPUT	Execute	BOOL	Starts the motion at a rising edge
	Velocity	REAL	Maximum velocity value (does not necessarily have to be reached)

⁶⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

	Name	Type	Comment ⁶⁾
	Acceleration	REAL	Acceleration (always +)
	Deceleration	REAL	Deceleration (always +)
VAR_OUTPUT	InVelocity	BOOL	Velocity reached (for the first time)
	Active	BOOL	Processing of data after preprocessing running
	CommandAborted	BOOL	Command aborted by subsequent command
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 8-23: Interface of the MC_MoveVelocity function block

Signal-time diagram

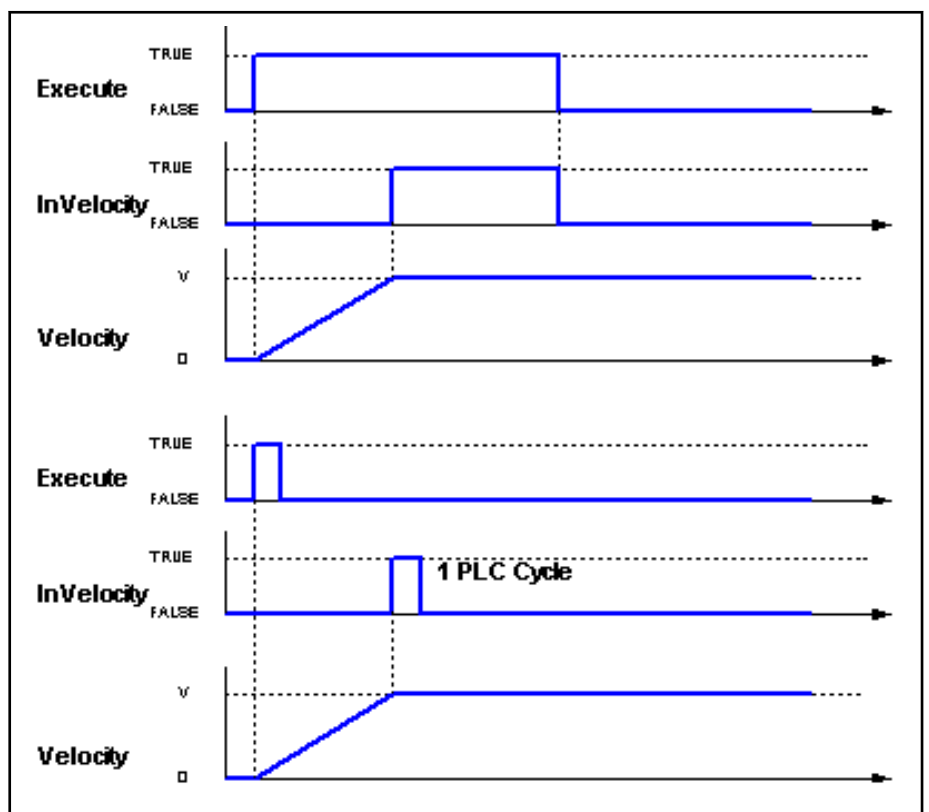


Fig. 8-33: Time diagram of MC_MoveVelocity (without aborted command)

⁶⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

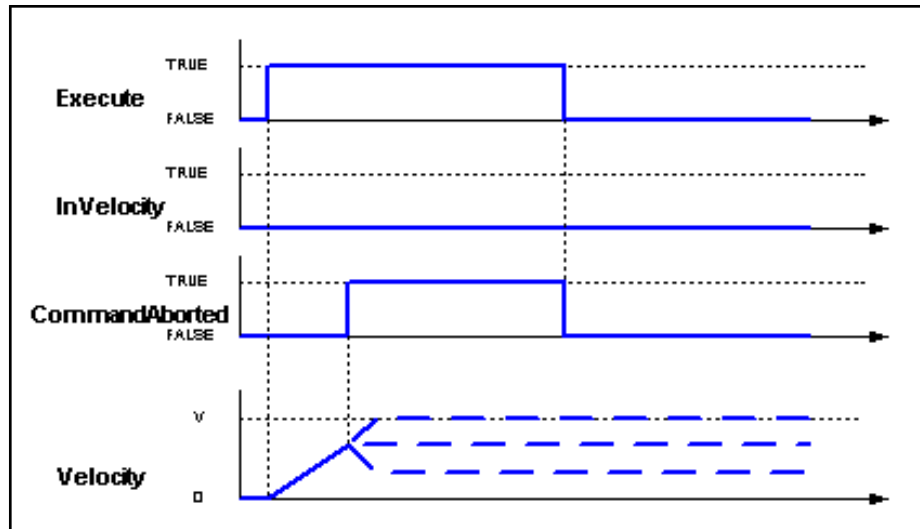


Fig. 8-34: Time diagram of MC_MoveVelocity (with aborted command)

Implementation example: The following figure shows two examples of two combined instances of the MC_MoveVelocity function block:

1. The left part of the time diagram shows a case in which the second instance is called after the job of the first instance is complete. Once the first function reached the required velocity of 3,000, the `Instance1.InVelocity` output **AND** the `Next` signal cause the second function block to continue the motion at a velocity of 2,000.
2. The right side of the time diagram shows a case where the first function block has not yet reached the velocity when the second function block is started. The following is shown: The first motion is restarted by `Go` at the `Instance1.Execute` input. While the first function block is still accelerating to reach a velocity of 3,000, it is interrupted by the `Test` signal which starts the second function block. The second function block is now running and decelerates to a velocity of 2,000.

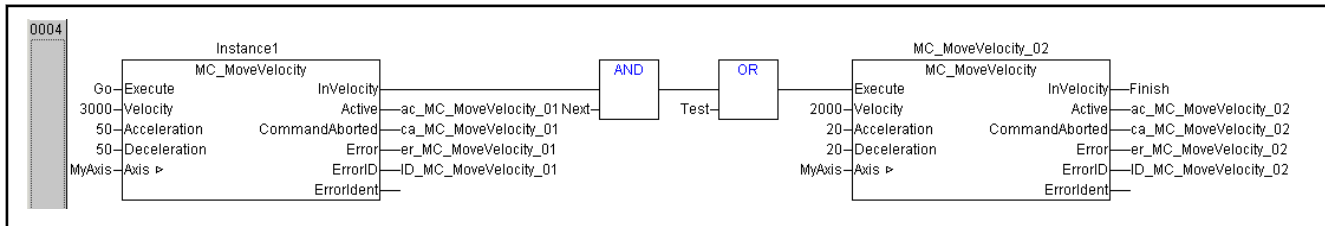


Fig. 8-35: Two instances of MC_MoveVelocity

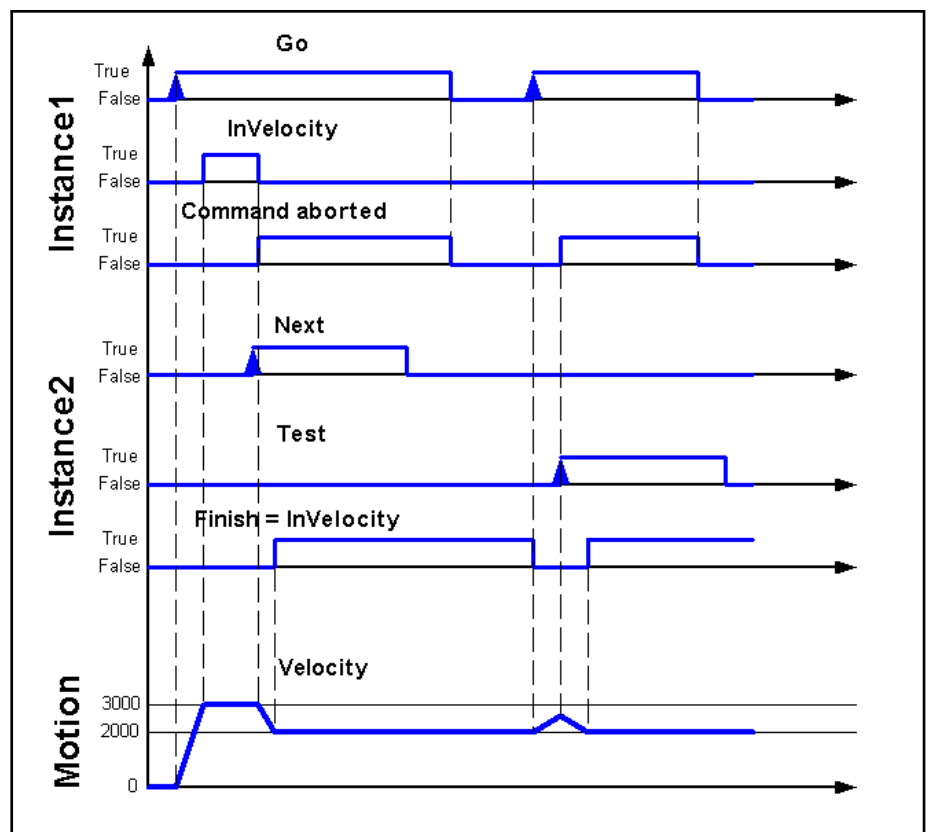


Fig. 8-36: Timing diagram: Two instances of MC_MoveVelocity

Troubleshooting

The function block generates error messages only at a positive edge or if TRUE is set at the Execute input.

Program:

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

8.5 Function blocks for parameter access (Parameter_FBs)

8.5.1 Parameter_FBs, general information

The RMB_EtherCat.library for IndraDrives does not support any acyclic parameter communication.

Function blocks are provided that require or get information of the cyclic telegram from IndraDrive.

- MC_ReadActual Position
- MC_ReadAxisError
- MC_ReadStatus

Reading or writing of parameters can optionally be implemented with "Acyclic Communication" function blocks of the "RIL_Ethercat" library.



The function blocks mentioned in the following are used to transfer parameters between control and drive.

	Function
MC_ReadActualPosition, page 460	Reads the current position of the drive.
MC_ReadAxisError, page 461,	Retrieves the error state of the addressed axis. One instance of the function block is to be used for each axis.
MC_ReadStatus, page 463	Reads the current status of the axis and outputs it in a decoded way.

Tab. 8-24: Function blocks of the "Parameter_FB" folder

8.5.2 MC_ReadActualPosition

Brief description The function block reads the current position of the drive.



A modification of the axis at the input of the function block requires another 0/1 transition at "Enable".

Library	Range
RMB_PLCopenFieldBus	Parameter_FBs

Tab. 8-25: Library assignment

Interface description

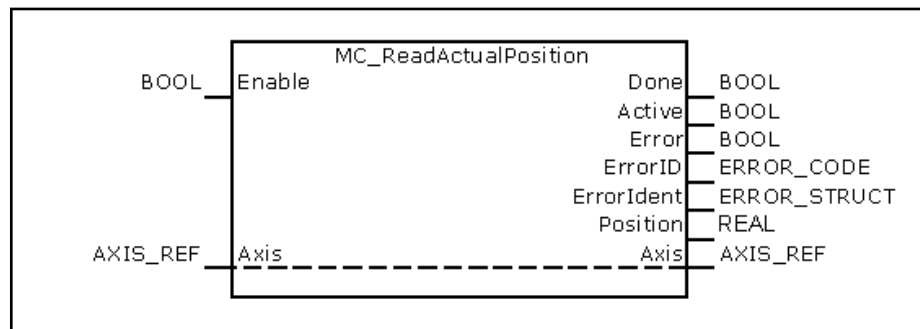


Fig. 8-37: MC_ReadActualPosition FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
VAR_INPUT	Enable	BOOL	Continuous reading of the position value as long as "Enable" is TRUE
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Position	REAL	Current absolute position of the drive ⁷⁾

Tab. 8-26: Interface of the MC_ReadActualPosition FB

⁷⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

Signal-time diagram

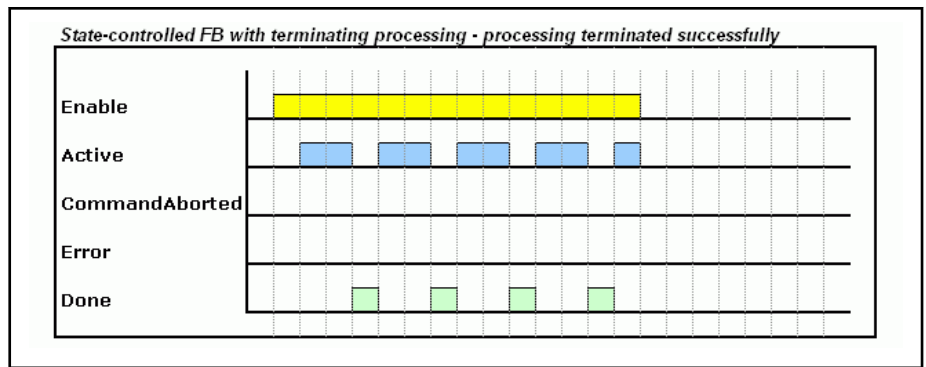


Fig. 8-38: Default signal-time diagram



The MC_ReadActualPosition function block only runs together with the IndraDrive drives.

Parameters used

The function block reads the current position of the drive "actual position value" S-0-0051 or the parameter specified in the drive on the input description of the field bus process data .

S-0-0051, the actual position value encoder 1 displays the current position of the motor encoder.

Troubleshooting

The MC_ReadActualPosition function block generates only error messages at a positive edge or if TRUE is set at the "Enable" input.

If there are configuration errors at the function block, the following is displayed

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

Otherwise, the task is given to the drive and it might react with an own error message.

8.5.3 MC_ReadAxisError

Brief description

The function block queries the error state of the addressed axis. One instance of the function block is to be used for each axis.



A modification of the axis at the input of the function block requires another 0/1 transition at "Enable".

Library	Range
RMB_PLCoOpenFieldBus	Parameter_FBs

Tab. 8-27: Library assignment

Interface description

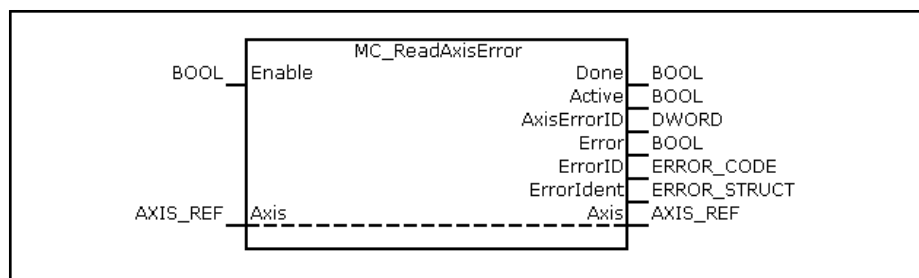


Fig. 8-39: MC_ReadAxisError FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
VAR_INPUT	Enable	BOOL	Continuous reading of the error information as long as "Enable" is TRUE
VAR_OUTPUT	Done	BOOL	Value available
	Active	BOOL	Processing of data after preprocessing running
	AxisErrorID	DWORD	It indicates the error of the drive of the axis
	Error	BOOL	Indicates that an error occurred when the function block instance was called
	ErrorID	ERROR_CODE	Short note to the error cause of the function block call, not of the axis error!
	ErrorIdent	ERROR_STRUCT	Detailed information on the error of the function block call, not the axis error!

Tab. 8-28: Interface of the MC_ReadActualPosition FB

Signal-time diagram

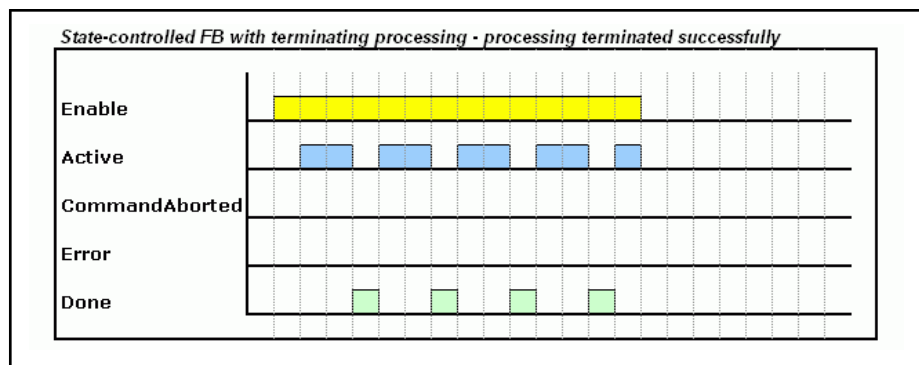


Fig. 8-40: Default signal-time diagram



The MC_ReadAxisError function blocks only runs together with the IndraDrive drives.

Parameters used

This function block queries the error status (S-0-0390, Diagnostic number) of the axis addressed.

This parameter stores the diagnostic number shown in the display. Thus, the control can generate individual diagnostics using this number (e.g. in languages in which the diagnostic texts are not saved in the drive).

Troubleshooting

The MC_ReadAxisError function block generates error messages only at a positive edge or if TRUE is set at the "Enable" input.

If there are configuration errors at the function block, the following is displayed

F_RELATED_TABLE

```

Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;

```

8.5.4 MC_ReadStatus

Brief description This function block reads the current status of the axis and outputs this in a decoded way.



A modification of the axis at the input of the function block requires another 0/1 transition at "Enable".



Due to the extension of the function block, it was modified in a way which is incompatible with the previous version!

Library	Range
RMB_PLCoenFieldBus	Parameter_FBs

Tab. 8-29: Library assignment

Interface description

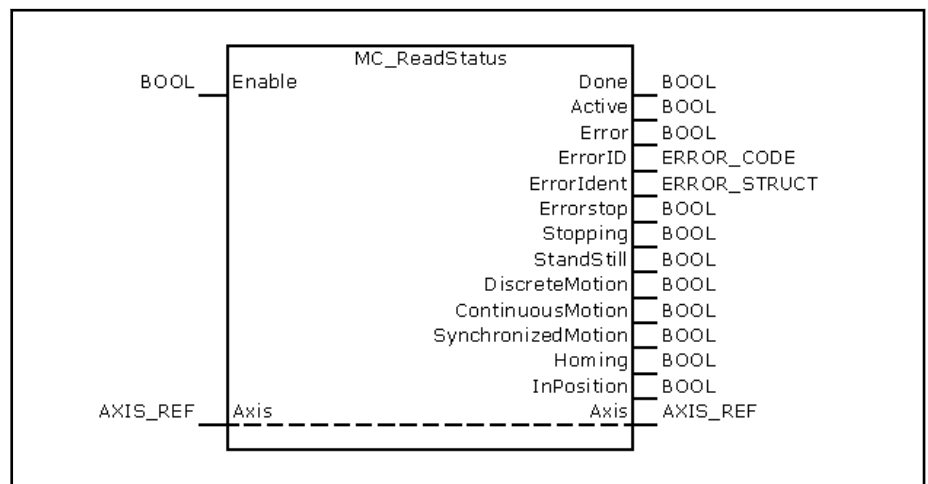


Fig. 8-41: MC_ReadStatus FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the specific axis
VAR_INPUT	Enable	BOOL	Continuous reading of the axis state as long as "Enable" is TRUE
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data after preprocessing running
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Errorstop	BOOL	Drive/axis in "Errorstop" state
	Stopping	BOOL	Drive/axis in "Stopping" state
	StandStill	BOOL	Drive/axis in "Standstill" state

	Name	Type	Comment
	DiscreteMotion	BOOL	Drive/axis in "DiscreteMotion" state
	ContinuousMotion	BOOL	Drive/axis in "ContinuousMotion" state
	Synchronized-Motion	BOOL	Drive/axis in "SynchronizedMotion" state
	Homing	BOOL	Drive in "Homing" state
	InPosition	BOOL	Drive in "InPosition" state

Tab. 8-30: Interface of the MC_ReadStatus FB

Signal-time diagram

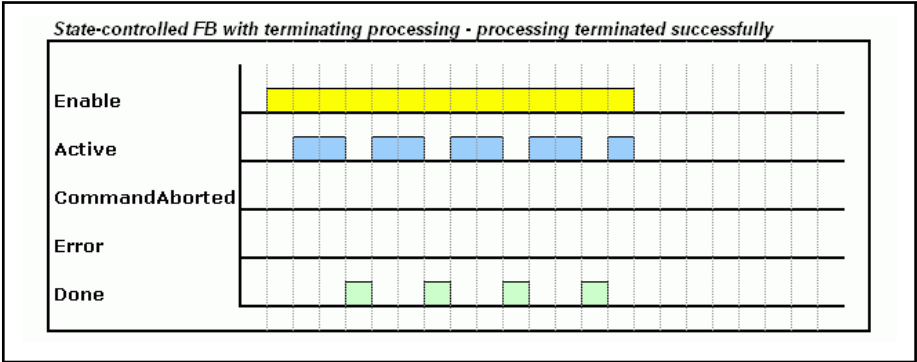


Fig. 8-42: Default signal-time diagram

 The MC_ReadStatus function block only runs together with the IndraDrive drives.

Parameters used This function block reads the current axis status (S-0-0144, signal status word or the parameter specified in the drive on the input description of the field bus process data and outputs it in a decoded way.

Troubleshooting The function block generates error messages only at a positive edge or if TRUE is set at the "Enable" input.

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

If the task was transferred to the drive and cannot be executed there, the drive outputs a respective error message.

8.6 System_Function blocks

The function blocks are contained in the "_Internal"

- MP_Drive

 These elements are not to be used, since side effects including malfunctions in the motion of the axes cannot be excluded!

8.7 PLC controls drives via field bus connection (field bus axes)

8.7.1 General information

IndraDrives can be controlled by an IndraControl via a field bus, depending on the drive interface.



The drives are connected to the control via a field bus.

The coupling via a Sercos II or Sercos III bus is not part of the description.

The activation is executed from the PLC program using MotionLogic function blocks according to the PLCopen concept.

Before operating such a field bus axis at the field bus, the following requirements have to be met:

- An IndraWorks project is created
- A control has been added to the project
- A field bus interface (onboard or function module) is set as master

Depending on the used library, declare the drives in the application:

- For RMB_PLCopenFieldBus.library, refer to [chapter 7.7.3 "Drive settings for the PLC communication" on page 422](#)
- For RMB_PLCopenEtherCAT_SoE.library, refer to ["Import Drive_XY drive" on page 433](#)

The PLC programs use MotionLogic function blocks, functions and data types summarized in several [Libraries, page 423](#).

These libraries are integrated together with the PLC program(s) into the IndraWorks project.

Additionally to the controlling of the field bus drive via the PLC program, values have to be exchanged with the drive saved in its parameters.

8.7.2 Creating field bus axes in the project

System and drive settings are required in the IndraWorks project at the PLC to ensure a correct communication between the drives and the PLC. The field bus axes are regular field bus devices at the field bus.

Drag the required drives from the library to the field bus master object.

Optionally, add a drive via the context menu of the master object: **Add ► Slave ► Drives**.

In this case, the new drive is added as last drive below the field bus master object.

Required settings for a PROFIBUS device

The drive has several sub devices in a Profibus device.

The drives (device files) in their original state are not provided with the required settings to communicate together with the PLCopen function blocks from the RMB_PLCopenFieldBus.Library via Profibus.

For IndraDrive, the following settings are required:

- ParamCh_IDN,
- F-Module_not_used, **(if the module is not used)**
- Input_8_Words,
- Output_10_Words.

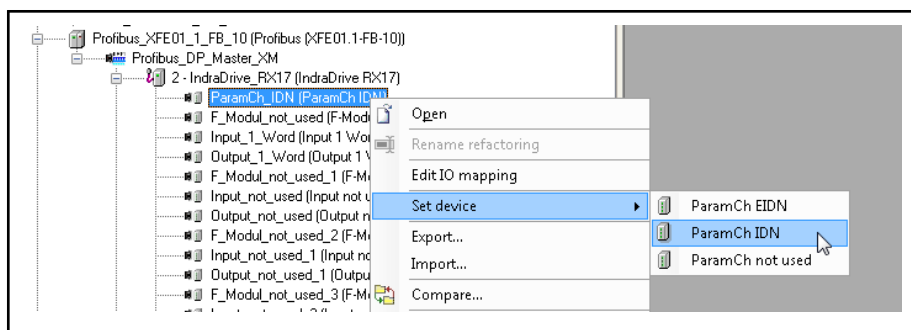


Fig. 8-43: IndraDrive, setting the parameter channel on the sub devices

Even the drives are not configured in their original state for such a communication with a PLC via Profibus.

Therefore, drive settings have to be made for the PLC communication.

Required settings for an EtherCAT device

For EtherCAT devices, the following settings are required:

- Configuring the first secondary operation mode, see [chapter 8.2.1 "Configuring the operation modes \(e.g. with FWA MPB20\)" on page 430](#)
- Configuration of signal status word and signal control word, see [chapter 8.2.2 "Configuration of signal status word and signal control word" on page 430](#)
- Configuring process data, see ["Configuring process data" on page 432](#)
- Configuring cyclic data, see [chapter 8.2.6 "Configuring cyclic data" on page 437](#)

8.7.3 Drive settings for the PLC communication

General information

To activate field bus drives with the PLCopen function blocks, some basic settings are required regarding the communication of the PLCopen functions blocks with the field bus drives.



The settings described do only refer to the communication between the drive and the PLC and do NOT describe the general drive commissioning.

The required settings can be selected via the "IndraWorks Engineering" or via "IndraWorks DS".



Currently, it **cannot** be parameterized via the Profibus connection of the PLC.

From the PC, on which the IndraWorks Engineering is running, the communication to parameterization has to be established via RS232 serial or via the existing Profibus DP master.

In the following, the parameterization is described via serial communication.

IndraDrive communication settings



The following settings do not apply to the operation of drives via EtherCAT.

For EtherCAT, select the following configurations, see [chapter 8.2 "IndraDrive – Required configurations" on page 429](#)

General information	In the following, the IndraDrive settings required for the communication are described.
Setting bus device address	Set the bus device address of the drive in the node "Master communication with regard to the set station address.
Length of parameter channel	By entering the parameter P-0-4083 directly using the parameter editor, the length of the parameter channel is set to 10 bytes. A newer IndraDrive firmware (from MPH-03VRS) automatically sets to 5 words (ParamCH_5_Words).
Setting process data and profile type	The profile type "Freely configurable mode" and the parameters of the cyclic data exchange with the PLC are set in the following: <i>Process data inputs (AT)</i> • P-0-4078, Field bus status word • S-0-0051, Actual position value encoder 1 • S-0-0040, Actual velocity value • S-0-0390, Diagnostic number • S-0-0144, Signal status word <i>Process data outputs (MDT)</i> • P-0-4077, Field bus control word • S-0-0282, Positioning command value • S-0-0259, Positioning velocity • S-0-0260, Positioning velocity • S-0-0359, Positioning deceleration • S-0-0145, Signal control word
Signal control word	Bit 0: S-0-0346: Positioning control word, bit number 4
Signal status word	Bit 0: S-0-0343: Message interpolator stopped, bit number 0
Operation mode settings	As primary operation mode, the operating mode "Drive-controlled positioning" has to be set. "Velocity control" has to be assigned as first secondary operation mode. The second and following secondary operation modes are not relevant. These modes are not selected by the PLCopen function blocks.
 The MB_Home function block only references if the drive is parameterized for referencing.	

8.7.4 Including required libraries

The PLCopen concept was implemented for IndraWorks Engineering for field bus types and is supported by the following libraries:

Sercos II and Sercos III

- ML_PLCopen.library, function blocks of PLCopen for Sercos drives.
- RIL_CommonTypes.library, system-comprehensive data types, addressing, error management.
- ML_Base.library, supplementing data types to the ML* system,
- Supplementing libraries, e.g. RIL_Uutilities.library.

PROFIBUS DP

- RMB_PLCopenFieldBus.library, function blocks and data types for Profibus, Profinet and Ethernet drives.

RMB_PLCoOpenEtherCAT_SoE.library

- RIL_CommonTypes.library, error management.
- Supplementing libraries, e.g. RIL_Uilities.library.

EtherCAT

- RMB_PLCoOpenEtherCAT_SoE.library, function blocks and data types for EtherCAT drives.

It was tried to maintain a uniform concept for field bus types. This also affects identical names for functionally identical function blocks and data types.

Adding required libraries

The RMB_PLCoOpenFieldBus.Library and RMB_PLCoOpenEtherCAT_SoE.Library libraries can be added via the library manager **Add library**....

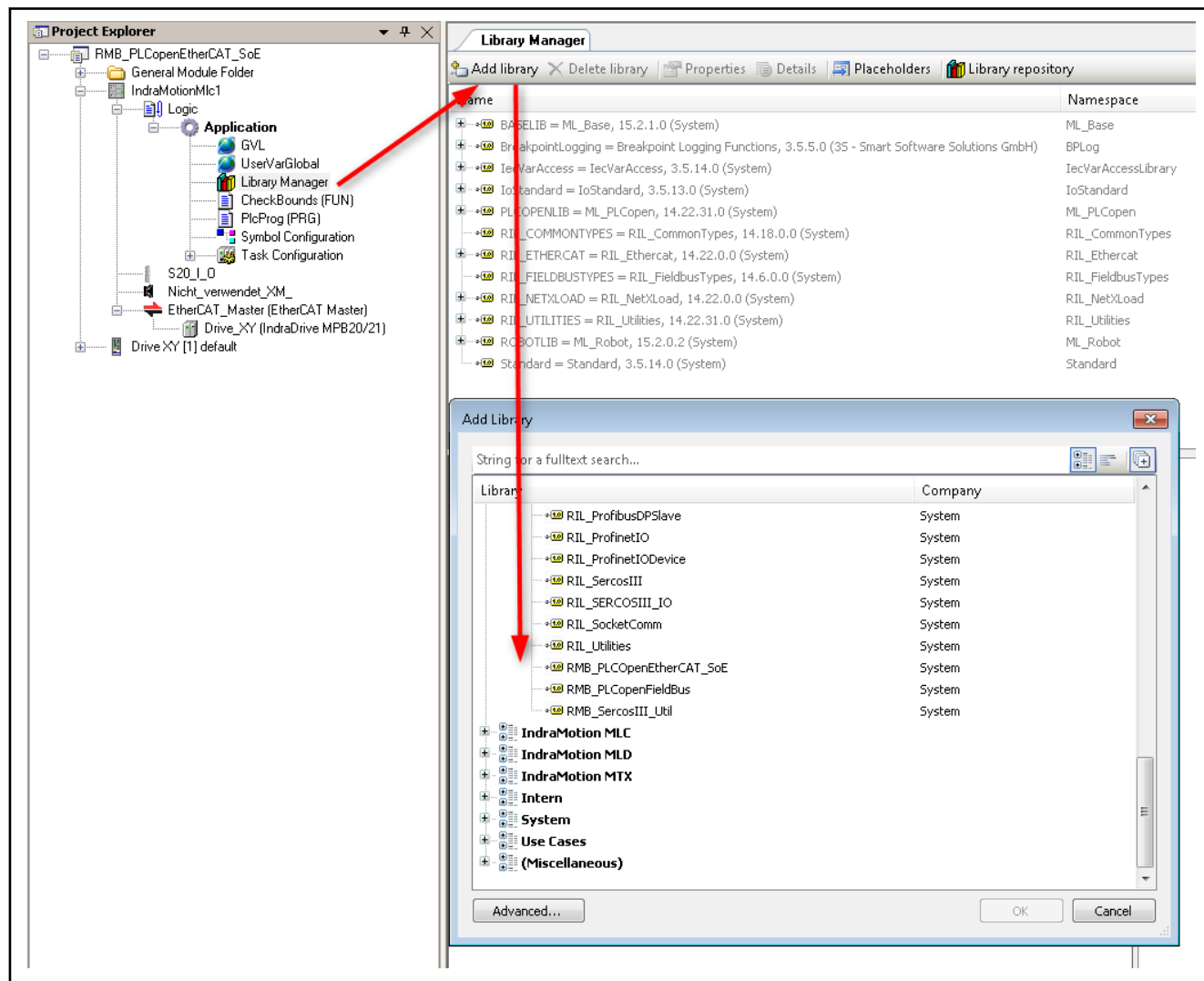


Fig. 8-44: Example: Library Manager

The elements of the RMB_PLCoOpenFieldBus.library and RMB_PLCoOpenEtherCAT_SoE.Library have to be used by specifying the namespace.

8.7.5 Addressing field bus drives in the PLC project

AXIS_REF, axis addresses of field bus drives

The structure AXIS_REF (RIL_CommonTypes) contains address information on the desired control and on the object controlled by this control (axis number).

In contrast to the AXIS_REF structure (RIL_CommonTypes) supporting the addressing of Sercos-based IndraDrives, the AXIS_REF structure (RMB_PLCopenFieldbus) is extended by two additional elements (MP_Drive and MP_DriveParameter). These two elements are only used by the system.

The axis number can differ from the station number set at the drive.

The Global Variable List "MLCVarGlobal" with all Sercos-based axes is automatically created and maintained for Sercos-based axes.

The global variable list "GVL_FieldBusDrives" with all field bus drives has to be manually created and maintained for the field bus drives.

The two first elements of each drive "CntrlNo" and "AxisNo" are preassigned in this list. The elements "MP_Drive" and "MP_DriveParameter" remain unchanged. Therefore, they do not have to be included in the list.

Global Variable List "GVL_FieldBusDrives", example

Namespace specification

Standard namespace for the RMB_PLCopenFieldBus.library: "FieldBus-Drives".

Program:

```
VAR GLOBAL
Drive_X: FieldBusDrives.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=FieldBusAxis_11); // IndraDrive X - example
Drive_Y: FieldBusDrives.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=FieldBusAxis_12); // IndraDrive Y - example
Drive_Z: FieldBusDrives.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=FieldBusAxis_13); // IndraDrive Z - example
Drive_n: FieldBusDrives.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=FieldBusAxis_14); // IndraDrive n - example
END_VAR
```

Standard namespace for the RMB_PLCopenEtherCAT_SoE.library: "PLCopenEtherCAT_SoE".

Program:

```
VAR GLOBAL
Drive_X: PLCopenEtherCAT_SoE.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=1001); // IndraDrive X - example
Drive_Y: PLCopenEtherCAT_SoE.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=1002); // IndraDrive Y - example
Drive_Z: PLCopenEtherCAT_SoE.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=1003); // IndraDrive Z - example
Drive_n: PLCopenEtherCAT_SoE.AXIS_REF :=
    (CntrlNo:=LOCAL_CNTRL,
     AxisNo:=1004); // IndraDrive n - example
END_VAR
```

Declaration AXIS_REF

Drive_A: FieldBusDrives.AXIS_REF :=

The data types are in the RMB_PLCopenFieldBus.library, folder "Datatypes", element "AXIS_REF".

Assignment CntrlNo

```
(CntrlNo: = LOCAL_CNTRL
```

The enumeration value LOCAL_CNTRL is contained in RIL_CommonTypes.lib, folder "Data types of logical addresses", CONTROLS.

The field bus function blocks always refer to the local PLC, that is LOCAL_CNTRL.

Assignment AxisNo

Taking the example of the RMB_PLCopenFieldBus.library:

```
AxisNo: = FieldBusAxis_1);
```

The enumeration value FieldBusAxis_1 is in the RIL_CommonTypes.lib, folder "Data types of logical addresses", OBJECTS.

To avoid ambiguities, there is an individual number range stored for field bus axes

```
FieldBusAxis_1 (6001) up to FieldBusAxis_99 (6099)
```

.

For field bus drives, specifications on the control and the axis number are used for the error output (ErrorIdent).

A functionality is not combined with the assignment of the axis number.



It is recommended to compare the station number and axis number.

AxisData[], axis data

The AxisData array, known from the Sercos axes, is not supported for field bus axes.

8.7.6 I/O addressing – RMB_PLCopenFieldbus

General information

On the Profibus, the field bus drives appear as ordinary I/O modules. An I/O address has to be assigned.

To facilitate the I/O address assignment, it is directly executed as variable mapping on the Profibus module of the IndraDrive.

I/O address assignment: IndraDrive

The parameter communication, the F-module (safety technology) and the cyclic communication at IndraDrives are always individual submodules of the device.

The communication "Drive to the PLC" consists of the following:

- Subdevice (ParamCh_5_Words), 5 words for acyclic parameter communication,
- Subdevice (F-module), optional use; not used here,
- Subdevices of the cyclic data exchange (8 words input or 10 word output)

Addresses are mapped at the respective subdevice channels of the drive.

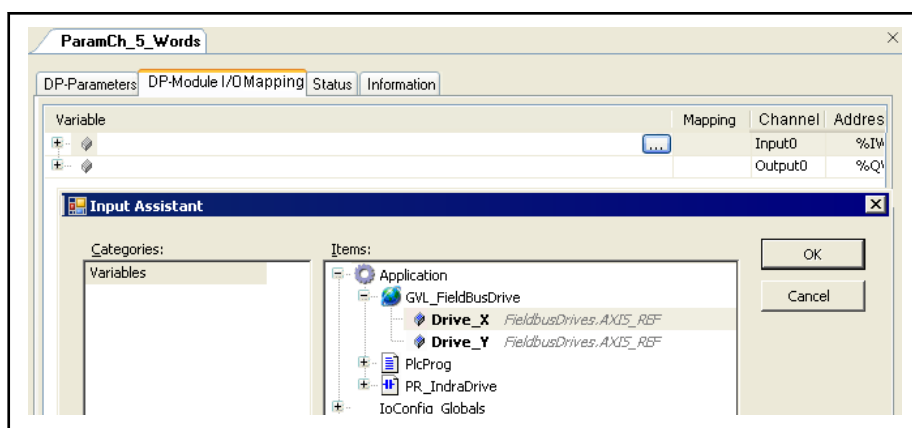


Fig. 8-45: Example: Input help for the mapping to select the drive

In AXIS_REF, the field bus drives are distinguished for the cyclic parameter exchange and "IndraDriveParameter" for the acyclic parameter exchange according to "IndraDrive"

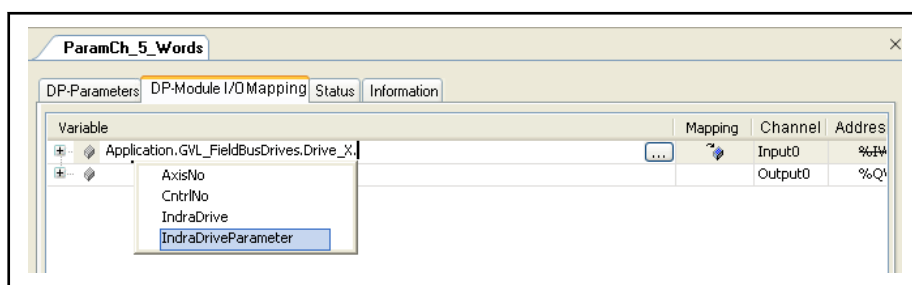


Fig. 8-46: Example: Mapping for the parameter exchange

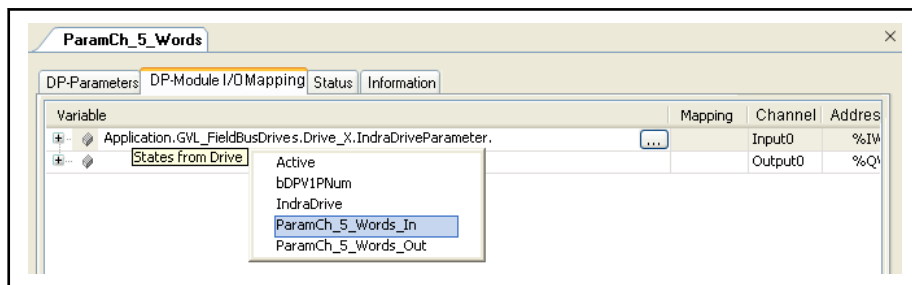


Fig. 8-47: Example: Example for the inputs of the parameter exchange

By direct mapping, the initial addresses for the parameter communication, the cyclic data exchange and - if available - an F-module with regard to the drive do not have to be considered.

8.7.7 I/O addressing – RMB_PLCoOpenEtherCAT_SoE

To control the drive via function blocks of the RMB_PLCoOpenEtherCAT_SoE.library, the following I/O assignments are required, see [chapter 8.2.6 "Configuring cyclic data" on page 437](#).

9 ML_ControllerBase.library

9.1 Introduction and overview

The library helps to implement a controller based on ready-made modules into the PLC.

By using the library, not only the implementation is facilitated, but also the controller behavior is unified.

The library is split into:

- [Basic class of the controller, page 473](#), controller functionality,
- [Controller management, page 476](#), controller response.



The library requires the **MOT (Motion) function package** of the system IndraLogic XLC/IndraMotion MLC 14VRS

9.2 Basic class of the controller

9.2.1 Basic class

The basic class of all controllers in the PLC is called `ML_ControllerBase`. Individual controllers can be derived from this class.

The controller management has to implement some of these specified methods to call an individual controller. The interfaces listed in the following are a collection of these methods grouped according to the functionality.

Depending on the respective system state of the control, the controller management calls the interface methods of the individual controller to set the state in the controller as well. Example are phase switching, operating states (bb, AB, AH...) or control states (position, velocity, force...).

`IActualValue`, `IModuleControl`, `IOperationModeHandler`, `IOutputData` with its encapsulated methods have to be implemented into the individual controller.

`IRetainHandler`, `IDigitalIO` and `IAdditionalSetValue` are optional.

Interface name	Function
<code>IActualValue</code>	Actual value detection
<code>IModuleControl</code>	Controller initialization Phase switching
<code>IOperationModeHandler</code>	For operation modes of a controller, refer to Controller management, page 476
<code>IOutputData</code>	Command value generation of a controller

Tab. 9-1: Implementable controller interface into the PLC

If a controller implements an interface, the respective "Get" method has to be loaded from the basic class.

During implementation, it has to be returned which function block implements this interface.

If the derived class of `ML_ControllerBase` implements the method, `THIS^` has to be returned.

Access methods to interfaces

Method name	Implemented interface
getIActualValue	IActualValue
getIModuleControl	IModuleControl
getIOperationModeHandler	IOperationModeHandler
getIOutputData	IOutputData
getIRetainHandler	IRetainHandler
getIDigitalIO	IDigitalIO
getIAdditionalSetValue	IAdditionalSetValue

Tab. 9-2: Access methods to interfaces

If the derived class of `ML_ControllerBase` implements the method, `THIS^` has to be returned.

Program:

```

METHOD getIActualValue : IActualValue
VAR_INPUT
END_VAR
VAR_
END_VAR

getIActualValue := THIS^;

```

9.2.2 Controller Data

All controllers contain data that is identical in each controller. Actual values and command values are part of this data.

In order not to implement this data in each controller, the data was summarized in a structure which every controller uses.

A pointer to this structure is assigned to each method of the individual interfaces. Thus, the methods can access this data. An overview on the most important elements in this structure is given in the following.

Elements of the `ML_OORV_CONTROLLERDATA` structure

Element name	Type	Description
stControlAxisRef	AXIS_REF	Axis reference to the controller axis of the controller
IrActualPos	LREAL	Actual position of the controller
IrActualVel	LREAL	Actual velocity of the controller
IrActualAcceleration	LREAL	Actual acceleration of the controller
IrActualForce	LREAL	Actual force of the controller
IrTargetPos	LREAL	Cyclic position command value of the controller
IrTargetVel	LREAL	Cyclic velocity command value of the controller
IrTargetAcceleration	LREAL	Cyclic acceleration command value of the controller
IrTargetForce	LREAL	Cyclic force command value of the controller

fbConverterCAxis	ML_Convert	Conversion functions to convert between internal and external units (4 bytes) Data conversion of a controller on page 475
fbConverterCAxis_DoublePrecision	ML_Convert_DoublePrecision	Conversion functions to convert between internal and external units (8 bytes) Data conversion of a controller on page 475
IrCycleTime	LREAL	Cycle time of the Motion kernel in μ s. New command values are generated during this period.

Tab. 9-3: Cyclic actual and command values

The internal units of the controller were specified. Thus, all data in the controller is assigned the corresponding unit.

Type	Rotary scaling	Translatory scaling
Position	Degrees	mm
Velocity	Degree/s	mm/s
Acceleration	Degree/s ²	mm/s ²
Force	Nm	KN
Analog value	%	%

Tab. 9-4: Data units

9.2.3 Data conversion of a controller

At the interfaces to the user, it is converted from user to internal units. The two function blocks "ML_Convert" and "ML_Convert_DoublePrecision" are provided.

The provided methods for conversion are:

Conversion methods between internal and external scaling

Method	Description
convFromAxis_Position	Converts an external to an internal position value
convFromAxis_Velocity	Converts an external to an internal velocity value
convFromAxis_Acceleration	Converts an external to an internal acceleration value
convFromAxis_Force	Converts an external to an internal force value
convFromAxis_Analog	Converts an external to an internal analog value
convToAxis_Position	Converts an internal to an external position value
convToAxis_Velocity	Converts an internal to an external velocity value
convToAxis_Acceleration	Converts an internal to an external acceleration value
convToAxis_Force	Converts an internal to an external force value
convToAxis_Analog	Converts an internal to an external analog value

Tab. 9-5: Conversion functions between internal and external scaling



Both function blocks differ with regard to their return value in representable precision.

The "ML_Convert" function block provides a 4-byte value as result. In contrast, the "ML_Convert_DoublePrecision" function block provides a result with an 8-byte data width.

If values - deviating from their respective precision - are loaded, rounding error are caused in the converted value.

9.2.4 Interface IActualValue

This interface contains the method for the actual value detection.

runActualValue The `runActualValue` method is called for the actual value detection of the controller. This method determines the actual axis values of the sensors. This values have to be entered into the controller data either in the `lrActualPos` element (actual position value) or `lrActualForce` element (actual force value).

9.2.5 Interface of the IModuleControl

This obligatory interface contains administrative methods.

InitModule This `InitModule` method is called when starting the whole controller. The controller can be initialized in this method. This method is called until it returns `TRUE`

SwitchToParamMode This `SwitchToParamMode` method is called when switching to the parameterization mode. This method is called until it returns `TRUE`

SwitchToOperationMode This `SwitchToOperationMode` method is called when switching to the operating mode. This method is called until it returns `TRUE`

9.2.6 Interface of the IOperationModeHandler

This interface is the interface to the operation modes of the controller. The operation modes are specified by the controller management. The following operation modes are intended.

bInitOperationCard This `bInitOperationCard` method is called to switch to a new operation mode. The number of the new operation mode is written in the controller data in the `byActOpCardIndex` element.

bCyclic This `bCyclic` method is called in each cycle. The number of the operation mode is written to the controller data in the `byActOpCardIndex` element. The control algorithm for this operation mode is implemented in this method.

9.2.7 Interface of the IOutputData

This interface contains the method for the control variable output to the actuator.

runOutputData The method "runOutputData" is called in each cycle. The output of the control variables to the actuator is implemented in this method.

9.3 Controller management

9.3.1 Connecting the controller to the controller management

The controller management creates the runtime environment of a controller.

To connect the controller to the controller management, the instance of a controller function block has to be connected to the `arItfModuleInterface` input of a `ML_OORV` function block instance.

The function block instance call has to be executed in a program connected to a task defining the cycle time of the controller.

As transfer parameter, apply the axis reference of the controller axis to the **AXIS** input the controller has to operate for.

MY_CONTROLLER_FB is an implemented controller derived from ML_ControllerBase.

Program:

```

Program MotionProg
VAR
  myController: MY_CONTROLLER_FB;
  myAdministration: ML_OORV := (aritifModuleInterface := myController);
END_VAR

myAdministration(AXIS := ControlAxis);

```

9.3.2 Cyclic procedure of the controller management

The following procedure is performed in each cycle of the controller management:

Detecting control information

Control information of the controller axis is loaded to the data pool of the controller. This information determines the subsequent response in the cyclic procedure.

State machine: operating state

It is run through the [State machine: Operating state, page 477](#).

Writing status information

The controller writes status information to the axis. The following values are converted and synchronized:

Element in the structure of the controller data	Parameters in the controller axis
IrActualPos	A-0-0100, Actual position value
IrActualVel	A-0-0102, Actual velocity value
IrActualTorque	A-0-0110, Actual torque/force value
wCStatusWord	A-0-0019, Status word of controller

Tab. 9-6: Synchronized status information

9.3.3 State machine: Operating state

In this state machine, initialization of the controller and phase switching are performed. In this state machine, the methods of the `IModuleControl` interface are used. The following table shows the states and the calls of the interface methods.

State	Description	Transition	Calling the interface method
IModuleControl	Initial state	Successful reading of all interfaces and initialization of the controller	getModuleControl getActualValue getOutputData getOperationModeHandler InitModule
PreParamState	Waiting for parameterization mode	ParamState if the user switches to parameterization mode	SwitchToParamMode

ParamState	Waiting for operating mode	OperationState if the user switches to the operating mode	SwitchToOperationMode
OperationState	Operating state In this state, the cyclic procedure is executed in the controller.	PreParamState if the operating mode is exited	see: Cyclic procedure of the controller, page 478

Tab. 9-7: State machine: operating state

9.3.4 Cyclic procedure of the controller

If the operating state is reached, the following procedure is executed in each cycle.

- Actual value detection** It is executed via the `runActualValue` method of the `IActualValue` interface of the controller.
- Operation mode handling** The currently selected operation mode specified by the control is evaluated and the respective operation mode of the controller is executed. See [Operation mode handling, page 478](#).
- Control value output** It is executed via the `runOutputData` method of the `IOutputData` interface of the controller.

9.3.5 Operation mode handling

If a controller is in operating mode, the state machine of the operation modes is effective. The operation mode states and the transitions are defined in the state machine.

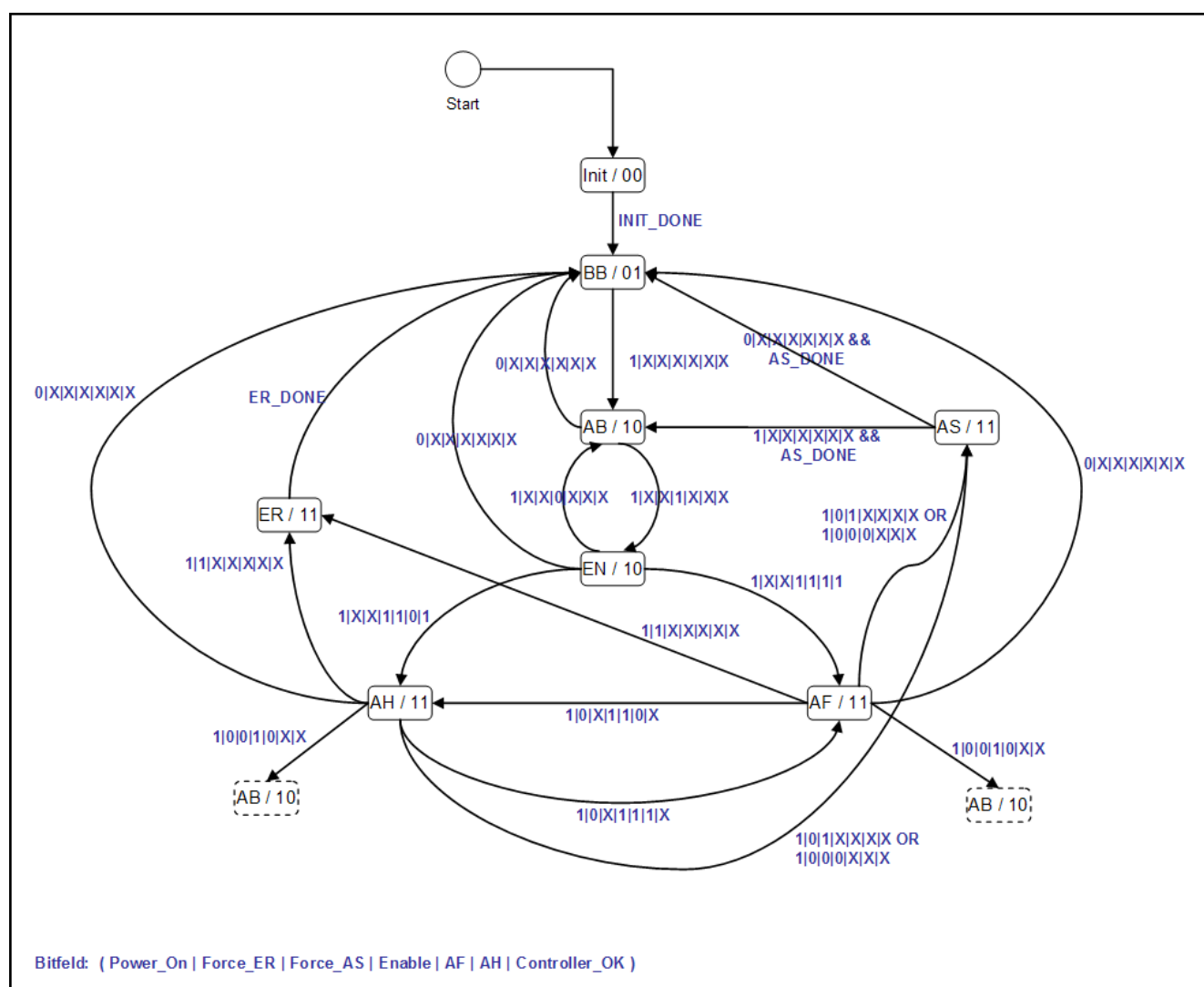


Fig. 9-1: State machines of the operation modes

Transitions from one state to the other are executed by the user via Motion functions blocks and the controller.

For the transitions, the conditions are specified for the respective control bits. If a bit is ignored for the state transition, an $\times$ is entered into the respective column.

The following table shows the significance of the control bits and who operates them.

Control bit	Description	Bit handling
Power_On	Status "Power on"	The controller specifies this bit. It indicates whether all conditions are fulfilled to add power to the device.
Force_ER	Force error	The controller sets this bit to indicate that an error is pending and to force the transition to the error state.
Force_AS	Force drive stop	The controller sets this bit to force a drive stop.
Enable	Power on	The user sets this bit. The controller has to add power (e.g. voltage supply of motor or pump).

AF	Drive enabled	The user sets this bit. Drive enabled granted.
AH	Drive Halt (AH)	The user sets this bit. The controller has to stop and control the drive.
Controller_OK	Controller ready for control.	The controller sets this bit if the controller is ready to control.
AS_DONE	Drive stop executed. Drive in standstill.	The controller sets this bit if the operation mode K_BAK_IND_INT_AS was selected and the drive is in "standstill".
ER_DONE	Error reaction executed.	The controller sets this bit if the operation mode K_BAK_IND_INT_ER was selected and the drive error reaction executed.
INIT_DONE	Initialization completed	The controller sets this bit when the initialization is completed.

Tab. 9-8: Significance of the control bits of the operation mode switching

The operation mode of the controller is called in each state.

The following table shows in which state which operation mode is called.

State	Operation mode
INIT	K_BAK_IND_INT_NO_RF
BB	K_BAK_IND_INT_NO_RF
AB	K_BAK_IND_INT_RF_INIT
EN	K_BAK_IND_INT_RF_CHECK
AH	K_BAK_IND_INT_AH
ER	K_BAK_IND_INT_ER
SFC	K_BAK_IND_INT_AS
AF	Operation modes selected by the user: K_OPCARD_PRIM K_OPCARD_SCND_1 K_OPCARD_SCND_2 K_OPCARD_SCND_3

Tab. 9-9: Operation mode call in the state...

In the "AF" state, the user calls the "Commanded operation mode". In all other states, one specified operation mode is called with one specified task.

This task is explained in the following:

Operation mode	Description
K_BAK_IND_INT_AH	Drive Halt: Best possible standstill. When selecting this operation mode, the controller has to decelerate the axis as much as possible and continue controlling.
K_BAK_IND_INT_ER	Error: This operation mode is selected if the axis is in transition to the error state. The controller has to execute the error transition.
K_BAK_IND_INT_AS	Drive stop: The controller has to stop the axis. The AS_DONE control bit has to be operated to signal that the drive reached standstill.
K_BAK_IND_INT_RF_CHECK	Drive enabled "Check": The controller has to check whether the controller may be enabled, i.e. whether all conditions are met to close the control loop.

K_BAK_IND_INT_RF_INIT	The controller is not enabled. No control may take place. In this operation mode, the controller has to check whether the conditions to add power are fulfilled. The control bit "Power_On" has to be operated accordingly.
K_BAK_IND_INT_NO_RF	The controller is not enabled. No control may take place. In this operation mode, the controller has to check whether the conditions to add power are fulfilled. The control bit "Power_On" has to be operated accordingly.
K_OPCARD_PRIM	Primary operation mode: The control loop is closed. The controller is in a cyclic position control.
K_OPCARD_SCND_1	1. secondary mode of operation: The control loop is closed. The controller is in a cyclic velocity control.
K_OPCARD_SCND_2	2. secondary mode of operation: The control loop is closed. The controller is in cyclic force control.
K_OPCARD_SCND_3	3. secondary mode of operation: The control loop is closed. The controller is in a cyclic analog value control.

Tab. 9-10: Operation mode of a controller

9.3.6 Predefined generators of the operation modes

Predefined generators are available in the operation modes. They have to determine the command values of the controller and to copy the data structure of the controller `ML_OORV_CONTROLLERDATA` (see [Controller Data](#), page 474).

Therefore, the "ML_Generator" function block provides the following methods.

GenerateCopyActualPos	This method copies the current controller position to the position command value of the data structure <code>ML_OORV_CONTROLLERDATA</code> .
GenerateCopyActualVel	This method copies the current controller velocity to the velocity command value of the data structure <code>ML_OORV_CONTROLLERDATA</code> .
GenerateCopyActualAcc	This method copies the current controller acceleration to the acceleration command value of the data structure <code>ML_OORV_CONTROLLERDATA</code> .
GenerateCopyActualTrq	This method copies the current controller force to the force command value of the data structure <code>ML_OORV_CONTROLLERDATA</code> .
GenerateCopyActualAna	This method copies the current controller analog value to the analog command value of the data structure <code>ML_OORV_CONTROLLERDATA</code> .
GenerateCopyCyclicPosition	This method copies the cyclic command position of the interpolator to the control in the position command value of the data structure <code>ML_OORV_CONTROLLERDATA</code> .
GenerateCopyCyclicVelocity	This method copies the cyclic velocity command value of the interpolator of the control to the velocity command value of the data structure <code>ML_OORV_CONTROLLERDATA</code> .
GenerateCopyCyclicAcceleration	This method copies the cyclic acceleration command value of the control interpolator to the velocity command value of the data structure <code>ML_OORV_CONTROLLERDATA</code> .
GenerateCopyCyclicTorque	This method copies the cyclic force command value of the control interpolator to the force command value of the data structure <code>ML_OORV_CONTROLLERDATA</code> .
GenerateCopyCyclicAnalog	This method copies the cyclic analog command value of the control interpolator to the analog command value of the data structure <code>ML_OORV_CONTROLLERDATA</code> .

9.3.7 Diagnostic handling

The controller management can report messages, warnings and errors. They are automatically logged in the diagnostic logbook. The `ML_MessageHandler` function block is provided to handle the data structure of the controller `ML_OORV_CONTROLLERDATA` (see [Cyclic actual and command values](#), page 474).

The following methods are available.

`run_MSGH_SetMessagecode`

Diagnostics can be initiated via this method. Diagnostics consist of four bytes and are coded as hexadecimal. The upper nibble determines the category (severity):

Diagnostic code	Description
Axxxxxx	Diagnostics is a message
Exxxxxx	Diagnostics is a warning
Fxxxxxx	Diagnostics is an error

Tab. 9-11: Coding the diagnostics

The system prioritize diagnostics:

- If no warning or error is reported, the latest message is output.
- When a warning is issued and no error is pending, a warning is output.
- When an error is issued and the priority of the error is higher than of all errors issued before, this is output.

The priority (class) is coded in the second nibble of an error.

`run_MSGH_ClearWarning`

This message can delete a previously issued warning. If no error is pending, the last message is output.

`run_MSGH_ClearErrors`

This method can delete all pending errors and warnings. This method is automatically executed by the controller management if the user triggers a "Delete error" command.

10 ML_IPOImpactControl.library

10.1 Introduction and overview

Target To implement the alternating control, a functionality allowing a direct interference in the interpolation of the MLC control is provided.

The objective is to manipulate the command value specification of the interpolator to adjust the behavior desired by the PLC controller.



The library requires the **MOT (Motion) function package** of the control.

Technical concept Normally, a new command value is provided when the interpolator executes a motion command. This command value is specified in each Motion cycle. To manipulate this specification, a call in an additional state machine precedes the call in the interpolation. This state machine monitors parameters used by the PLC controller to set a desired behavior and to influence the actual interpolation.

Refer to the following simplified example:

To "freeze" the command value generation on an existing value, the state machine in front can request the interpolator not to perform further calculations. Subsequently, the interpolator permanently outputs the latest value known. This causes a motion stop even though the actual motion command is still active. The PLC program does not detect this state, since neither "Done" nor the flag "CommandAborted" are set.

The concept is designed for the following manipulating options:

Manipulating options

Action	Comment
Direction-dependent interpolation stop	This action is also called "Break".
Direction-dependent jump to the end value of the interpolation.	This action is also called "Jump".
Setting individual position and velocity values to the "Break" state or at the end of a motion command.	This functionality is required to write values back from the PLC to the active motion command.

The transition to the "Break" state can be executed with different acceleration values.



The manipulation affects only the following commands:

- [MC_MoveAbsolute, page 183](#)
- [MC_MoveAdditive, page 186](#)
- [MC_MoveRelative, page 188](#)
- [MC_MoveVelocity, page 192](#)

The figure below shows the interpolated values for position, velocity and acceleration during an "MC_MoveAbsolute" command and the interference in the interpolation at 100 degrees.

Between (1) and (2), the interpolation is stopped ("Break" state). No new command values are generated.

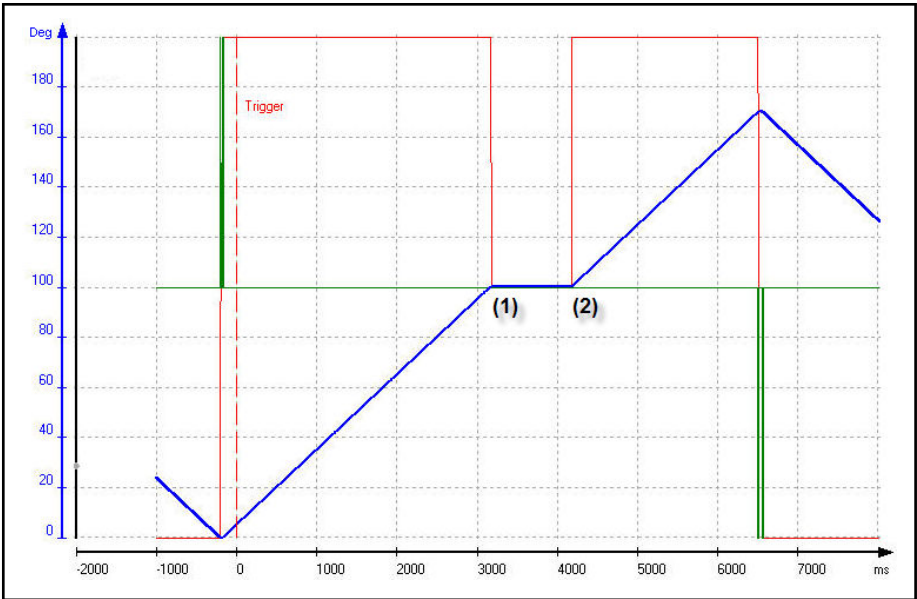


Fig. 10-1: Blue = interpolated position, red = interpolated velocity, green = interpolated acceleration

10.2 ML_IPOImpactControl

Brief description The ML_IPOImpactControl function block and its methods manipulate the interpolation.

The function block has to be called cyclically, since it additionally monitors the error state of the axis.

If the axis reports an error, all manipulations set are automatically reset.

Target system	Library
IndraLogic XLC/IndraMotion MLC 14VRS MOT function package	ML_IPOImpactControl

Tab. 10-1: Library assignment

Interface

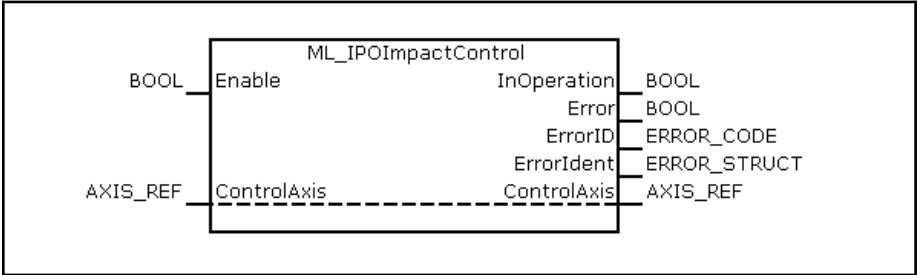


Fig. 10-2: ML_IPOImpactControl function block

	Name	Type	Comment
VAR_INPUT	Enable	BOOL	If "Enable" is TRUE, the interpolation can be affected
VAR_OUTPUT	InOperation	BOOL	Function block is enabled, interpolation can be affected
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on error cause

	Name	Type	Comment
	ErrorIdent	ERROR_STRUCT	Detailed error information
VAR_IN_OUT CONSTANT	ControlAxis	AXIS_REF	Contains information on the concrete axis.

Tab. 10-2: Interface of the ML_IPOImpactControl function block

10.3 ML_IPOImpactControl Methods

10.3.1 mSetNewState

Brief description This method sets a new state to affect the interpolation.

Signature

```
METHOD mSetNewState : DINT
VAR_INPUT
    eNewState: eIPOCONTROL_STATE;
END_VAR
```

Possible values for eNewState:

Value	Comment
IPOCONTROL_STATE_NEUTRAL	Interpolation should go into "neutral" state or remain.
IPOCONTROL_STATE_BREAK	Interpolation should go into "break" state.
IPOCONTROL_STATE_JUMP	Interpolation should go into "jump" state.

10.3.2 mGetNewState

Brief description This method provides the administrative state set with "mSetNewState".

Signature

```
METHOD mGetNewState : DINT
VAR_INPUT
    peNewState: POINTER TO eIPOCONTROL_STATE;
END_VAR
```

Possible values for peNewState:

Value	Comment
IPOCONTROL_STATE_NEUTRAL	Interpolation should go into "neutral" state or remain.
IPOCONTROL_STATE_BREAK	Interpolation should go into "break" state.
IPOCONTROL_STATE_JUMP	Interpolation should go into "jump" state.



This function does not provide the current interpolation state.
This information provides the "mGetAxisControlledBy" method.

10.3.3 mSetDirection

Brief description This method sets the direction in which the interpolation should be affected.

Signature

```
METHOD mSetDirection : DINT
VAR_INPUT
```

ML_IPOImpactControl.library

```

    eDirection: eIPOCONTROL_DIRECTION;
END_VAR

```

Possible values for eDirection:

Value	Comment
IPOCONTROL_DIRECTION_NONE	Interpolation is not affected in any direction.
IPOCONTROL_DIRECTION_FORWARD	Interpolation should be affected in forward direction.
IPOCONTROL_DIRECTION_BACKWARD	Interpolation should be affected in backward direction.
IPOCONTROL_DIRECTION_BACKWARD OR IPOCONTROL_DIRECTION_FORWARD	Interpolation should be affected in both directions.

10.3.4 mGetDirection

Brief description This method provides the direction, in which the interpolation should be affected.

Signature

```

METHOD mGetDirection : DINT
VAR_INPUT
    peDirection: POINTER TO eIPOCONTROL_DIRECTION;
END_VAR

```

Possible values for peDirection:

Value	Comment
IPOCONTROL_DIRECTION_NONE	Interpolation is not affected in any direction.
IPOCONTROL_DIRECTION_FORWARD	Interpolation should be affected in forward direction.
IPOCONTROL_DIRECTION_BACKWARD	Interpolation should be affected in backward direction.
IPOCONTROL_DIRECTION_BACKWARD OR IPOCONTROL_DIRECTION_FORWARD	Interpolation should be affected in both directions.

10.3.5 mSetAcceleration

Brief description This method sets the individual acceleration to be considered by the interpolation at the state transition

"neutral" ↔ "break"

.

Signature

```

METHOD mSetAcceleration : DINT
VAR_INPUT
    rAcceleration: REAL ;
END_VAR

```


10.3.6 mGetAcceleration

Brief description This method provides the individual acceleration to be considered by the interpolation at the state transition.

Signature

```
METHOD mGetAcceleration : DINT
VAR_INPUT
    prAcceleration: POINTER TO REAL;
END_VAR
```

10.3.7 mSetBreakBehaviour

Brief description This method sets the "Break" behavior to be set at the transition
"neutral" ↔ "break"

Signature

```
METHOD mSetBreakBehaviour : DINT
VAR_INPUT
    eBreakBehaviour : eIPOCONTROL_BREAKBEHAVIOUR;
END_VAR
```

Possible values for eBreakBehaviour:

Value	Comment
IPOCONTROL_BREAKBEHAVIOUR_DIRECT	State transition is direct.
IPOCONTROL_BREAKBEHAVIOUR_MOVECMD	The state transition is performed with the acceleration value of the current motion command.
IPOCONTROL_BREAKBEHAVIOUR_INDIVIDUAL	The state transition is performed with an individual acceleration value.

10.3.8 mGetBreakBehaviour

Brief description This method provides the "Break" behavior set at the transition
"neutral" ↔ "break"

Signature

```
METHOD mGetBreakBehaviour : DINT
VAR_INPUT
    peBreakBehaviour: POINTER TO eIPOCONTROL_BREAKBEHAVIOUR;
END_VAR
```

Possible values for peBreakBehaviour:

Value	Comment
IPOCONTROL_BREAKBEHAVIOUR_DIRECT	State transition is direct.
IPOCONTROL_BREAKBEHAVIOUR_MOVECMD	The state transition is performed with the acceleration value of the current motion command.
IPOCONTROL_BREAKBEHAVIOUR_INDIVIDUAL	The state transition is performed with an individual acceleration value.

10.3.9 mSetPosition

Brief description This method sets the new position to be considered by the interpolation.

Signature

```
METHOD mSetPosition : DINT
VAR_INPUT
    lrPosition: LREAL;
END_VAR
```

10.3.10 mGetPosition

Brief description This method provides the new position to be considered by the interpolation.

Signature

```
METHOD mGetPosition : DINT
VAR_OUTPUT
    lrPosition: LREAL;
END_VAR
```

10.3.11 mSetValidValues

Brief description This method makes the settings regarding the velocity and position values to be considered by the interpolation.

Signature

```
METHOD mSetValidValues : DINT
VAR_INPUT
    eValidValues: eIPOCONTROL_RESPECT;
END_VAR
```

Possible values for eValidValues:

Value	Comment
IPOCONTROL_RESPECT_NONE	The interpolation does not consider any value.
IPOCONTROL_RESPECT_POSITION	The interpolation considers the position value.
IPOCONTROL_RESPECT_VELOCITY	The interpolation considers the velocity value.
IPOCONTROL_RESPECT_POSITION OR IPOCONTROL_RESPECT_VELOCITY	The interpolation considers both values.

10.3.12 mGetValidValues

Brief description This method provides the settings regarding the velocity and position values to be currently considered by the interpolation.

Signature

```
METHOD mGetValidValues : DINT
VAR_OUTPUT
    peValidValues: POINTER TO eIPOCONTROL_RESPECT;
END_VAR
```

Possible values for peValidValues:

Value	Comment
IPOCONTROL_RESPECT_NONE	The interpolation does not consider any value.
IPOCONTROL_RESPECT_POSITION	The interpolation considers the position value.
IPOCONTROL_RESPECT_VELOCITY	The interpolation considers the velocity value.
IPOCONTROL_RESPECT_POSITION OR IPOCONTROL_RESPECT_VELOCITY	The interpolation considers both values.

10.3.13 mSetVelocity

Brief description This method sets the new velocity to be considered by the interpolation.

Signature

```
METHOD mSetVelocity : DINT
VAR_INPUT
    rVelocity: REAL;
END_VAR
```

10.3.14 mGetVelocity

Brief description This method provides the new velocity to be considered by the interpolation.

Signature

```
METHOD mGetVelocity : DINT
VAR_INPUT
    prVelocity: POINTER TO REAL;
END_VAR
```

10.3.15 mGetAxisControlledBy

Brief description This method provides the information on whether the axis interpolation is affected by the PLC program.

Signature

```
METHOD mGetAxisControlledBy : DINT
VAR_INPUT
    peAxisControlledBy: POINTER TO eIPOCONTROL_CONTROLEDBY;
END_VAR
```

Possible values for peAxisControlledBy:

Value	Comment
IPOCONTROL_CONTROLEDBY_DEFAULT	Interpolation not affected.
IPOCONTROL_CONTROLEDBY_PLC	The PLC program affects the interpolation.

10.4 Using methods

The two examples - states "Break" and "Jump" - to show the call of the methods and its effect.

Enable "Break" state

In "Break" state, the interpolation is paused. Thus, no new command values are calculated in the Motion cycle.

The Motion function block continuous operating normally and does neither set the state "Done" nor the state "CommandAborted".

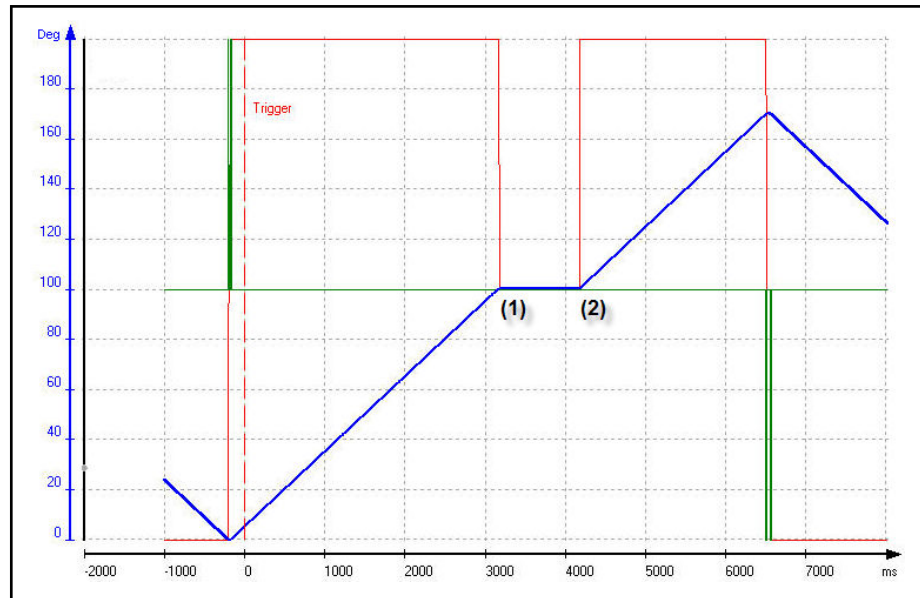


Fig. 10-3: Blue = interpolated position, red = interpolated velocity, green = interpolated acceleration

Method calls

```
//Create an instance of ML IPOImpactControl, here MyIpoImpactControl,
//within the variable section of the PLC
MyIPOImpactControl: ML_IPOImpactControl;

//Enable MyIPOImpactControl and wait for InOperation becomes "TRUE"
MyIPOImpactControl(...);

//Set all values to default. Just to start from a neutral situation
//Set state to neutral
MyIPOImpactControl.mSetNewState(eNewState:=IPOCONTROL_STATE_NEUTRAL);
//Set direction to neutral.
MyIPOImpactControl.mSetDirection(eDirection:=IPOCONTROL_DIRECTION_NONE);
//Set valid values to neutral.
MyIPOImpactControl.mSetValidValues(eValidValues:=IPOCONTROL_RESPECT_NONE);

//Now start a MC_MoveAbsolute.
//For this example the move starts at 0° and ends at 170° degree
MC_MoveAbsolute(...);

//Wait for position 100° degree reached (See picture above, mark:(1))
//Set the impact direction to forward
MyIPOImpactControl.mSetDirection(eDirection:=IPOCONTROL_DIRECTION_FORWARD);
//Set state to break
MyIPOImpactControl.mSetNewState(eNewState:=IPOCONTROL_STATE_BREAK);

//Optional it is possible to set a individual position and velocity between
//the marks:(1) and (2) (not shown in the picture above)
//These values will be used as a new start condition for the interpolation
//Set a new velocity
MyIPOImpactControl.mSetVelocity(rVelocity:= 1.0);
//Set a new position
MyIPOImpactControl.mSetPosition(lrPosition:= 105.0);
//Activates the values
MyIPOImpactControl.mSetValidValues(eValidValues:=IPOCONTROL_RESPECT_POSITION
OR IPOCONTROL_RESPECT_VELOCITY);

//Check if the axis is under PLC control
// ^peAxisControlledBy has to be IPOCONTROL_CONTROLLED_BY_PLC
MyIPOImpactControl.mGetAxisControlledBy(peAxisControlledBy);

//Wait 1000ms (See picture above, mark:(2))
```

```
//Set state to neutral, movement continues
MyIPOImpactControl.mSetNewState(eNewState:=IPOCONTROL_STATE_NEUTRAL);
```

Enable "Jump" state

In "Jump" state, the interpolation jumps to the end state of the current Motion command.

The Motion function block goes automatically into the "Done" state.



This state transition is not possible for MC_MoveVelocity.

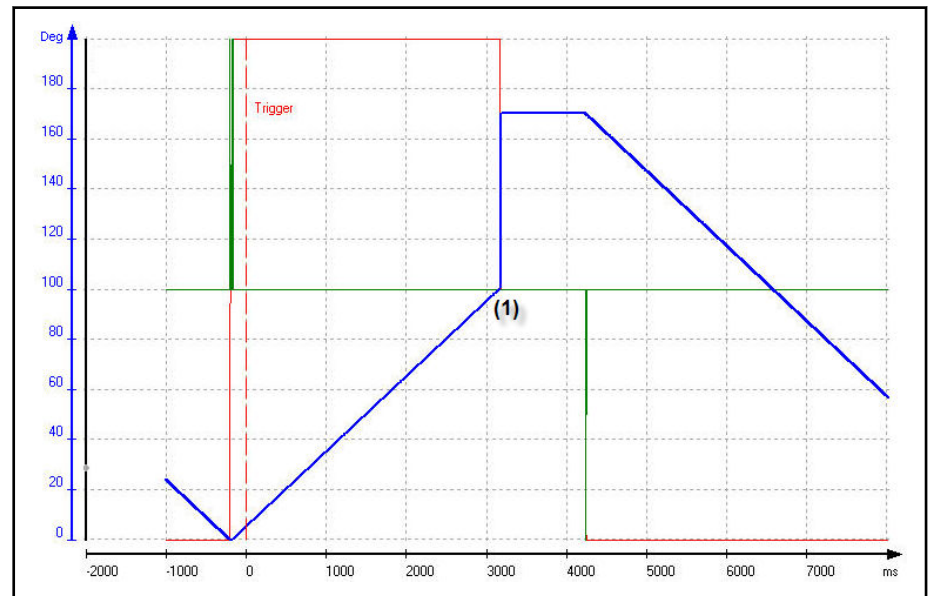


Fig. 10-4: Blue = interpolated position, red = interpolated velocity, green = interpolated acceleration

Method calls

```
//Create an instance of ML_IPOImpactControl, here MyIpoImpactControl,
//within the variable section of the PLC
MyIPOImpactControl: ML_IPOImpactControl;

//Enable MyIPOImpactControl and wait for InOperation becomes "TRUE"
MyIPOImpactControl(...);

//Set all values to default. Just to start from a neutral situation
//Set state to neutral
MyIPOImpactControl.mSetNewState(eNewState:=IPOCONTROL_STATE_NEUTRAL);
//Set direction to neutral.
MyIPOImpactControl.mSetDirection(eDirection:=IPOCONTROL_DIRECTION_NONE);
//Set valid values to neutral.
MyIPOImpactControl.mSetValidValues(eValidValues:=IPOCONTROL_RESPECT_NONE);

//Now start a MC_MoveAbsolute.
//For this example the move starts at 0° and ends at 170° degree
MC_MoveAbsolute(...);

//Wait for position 100° degree reached (See picture above, mark:(1))
//Set the impact direction to forward
MyIPOImpactControl.mSetDirection(eDirection:=IPOCONTROL_DIRECTION_FORWARD);
//Set state to jump
MyIPOImpactControl.mSetNewState(eNewState:=IPOCONTROL_STATE_JUMP);

//Now the MC_MoveAbsolute is done immediately, position reaches 170° degree.

//Optional it is possible to set a individual position and velocity at mark:(1)
// (not shown in the picture above)
//These values will be used as a new start condition for the next MC_MoveCmd.
//Set a new velocity
```

ML_IPOImpactControl.library

```
MyIPOImpactControl.mSetVelocity(rVelocity:= 1.0);  
    //Set a new position  
MyIPOImpactControl.mSetPosition(lrPosition:= 180.0);  
    //Activates the values  
MyIPOImpactControl.mSetValidValues(eValidValues:=IPOCONTROL_RESPECT_POSITION  
                                     OR IPOCONTROL_RESPECT_VELOCITY);
```

11 ML_ParamBase.library

11.1 Introduction and overview



The library requires the **MOT (Motion) function package** of the system IndraLogic XLC/IndraMotion MLC 14VRS.

Using this library, the controller axis can be equipped with extended parameterability, i.e. the PLC functionality is controlled via S- and P-parameters and the status is determined.

This functionality parameterizes a controller in the PLC, similar to a drive. This is shown in the following figure.

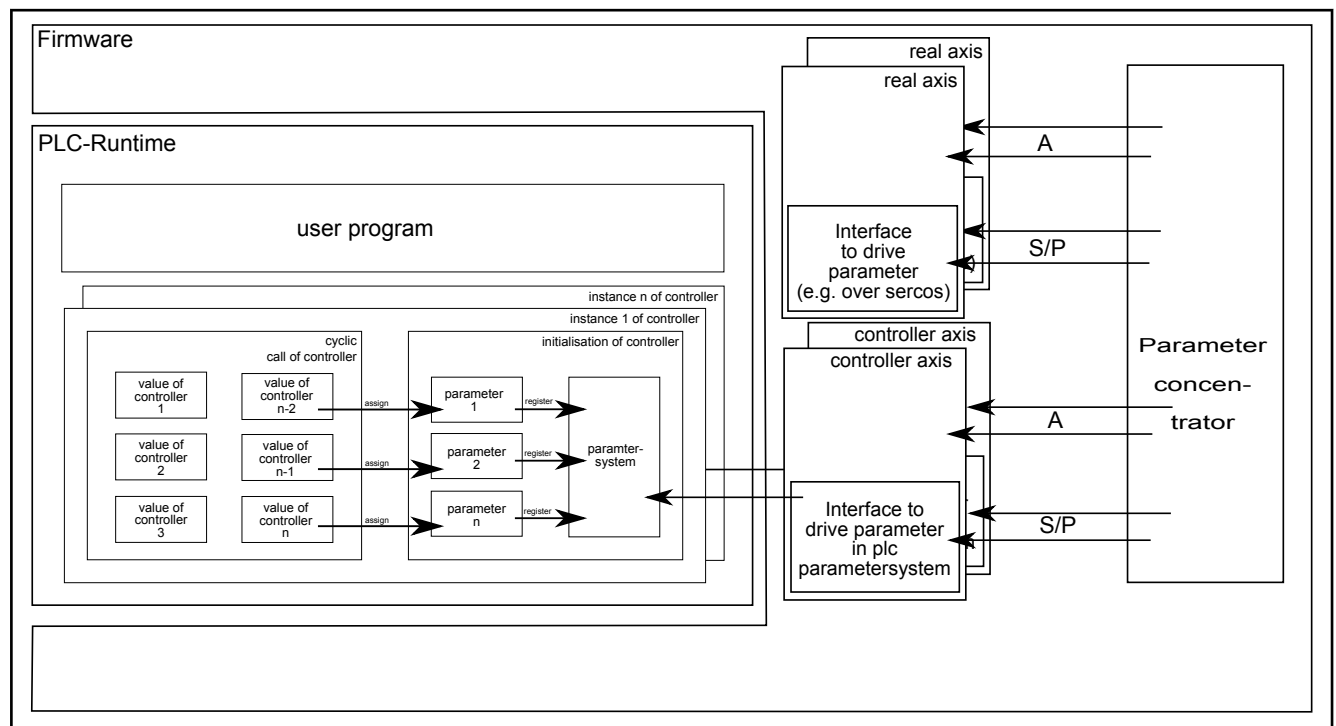


Fig. 11-1: Using a controller axis and a PLC

A controller axis can be provided with S- and P-parameters like a real axis.

Since the controller axis is calculated in the PLC, the S- and P-parameters have to be in the PLC. Thus, a parameter system is created in the PLC which connects to the firmware.

Furthermore, parameters are created in the PLC. The data is assigned to these parameters. To use these parameters, they have to be registered in the parameter system.

This library contains the [PARAM_Parametersystem](#), page 494, and two different parameter definitions:

- [PARAM_SercosParameter](#), page 498
- [PARAM_ExtSercosParameter](#), page 503

A function block to process command sequences is added to the library ([PARAM_CmdExecuter](#), page 505).

11.2 Function blocks

11.2.1 PARAM_Parametersystem

Initialization

Brief description The parameter system has to be initialized for each download or PLC reset. To assign the parameter system to a controller axis, specify an axis reference for the initialization.

Signature

```
METHOD Init
VAR_INPUT
    Axis : AXIS_REF;
END_VAR
```

Registering parameters

Brief description The parameter system can only identify which parameters are assigned to the controller axis if they are registered in the parameter system.

Therefore, the **addParameter** method has to be called using its address. Each parameter to be used has to be registered.

If a parameter is registered with an already registered EIDN, the registration fails and the old parameter is retained.

Signature

```
METHOD addParameter : DWORD
VAR_INPUT
    Parameter : POINTER TO PARAM_SercosParameter;
END_VAR
```

Functionalities

General information

The following functions are provided by the parameter system.

Additionally, the parameter system implements some interfaces ([PARAM_SercosParameter](#), page 498).

The respective parameter number is specified in the description of the method. The parameters have to be externally created and then registered.

Therefore, the mode of operation of the above mentioned interfaces has to be considered.

The named methods are to be seen as low level methods.

Language selection

The parameter system supports the usage of different languages of one parameter. Thus, the display language of the parameter name and the unit are switched. The languages defined in PARAM_Language ([Language](#), page 507) can be used.

For more information, refer to the section "Data types" ([Data types](#), page 507).

The language can be switched in the PLC via "Property Language". Additionally, the parameter system reacts to the EIDN S-0-0265.

Reading parameters

Brief description

Parameters can be read in the PLC via the **ParamRead** method. In this case, the parameters are directly accessed. Apart from the parameter value, its name, unit, minimum value, maximum value and attribute can be read.

This method is a system-close function. Thus, it is provided with some of the characteristics explained in the following. The advantage is that the parameter data is available in the same cycle.

The pointer **length** points to a variable that specifies the number of valid bytes written in **data**.

For lists, units and names, this specification is increased by the size of the sercos header (4 bytes).

Signature

```
METHOD ParamRead : SERCOS_ERROR
VAR_INPUT
  element: SERCOS_ELEMENT; // Element to read (Name, Unit, Value,...)
  eidn:    DWORD;           // EIDN to Read, e.g. 265 (S-0-0265)
  length:  POINTER TO UINT; // Pointer to Value for written size in data
  data:    PARAM_upData;    // Union to get different typesm, of pointer
END_VAR
```

NOTICE

Overwriting memory

Reserve sufficient memory. Otherwise, accidental overwriting can result in the memory.

Reading parameters

Brief description

As for the **ParamRead** method, **ParamRead** can be used to write on parameter data.

Depending on the attribute, the values cannot be written at any point in time. Thus, a write protection can be defined for certain sercos phases in the parameters.

Signature

```
METHOD ParamWrite : SERCOS_ERROR
VAR_INPUT
  element: SERCOS_ELEMENT; // Element to write (Name, ...)
  eidn:    DWORD;           // EIDN to write
  length:  POINTER TO UINT; // Pointer to Value to write size
  data:    PARAM_upData;    // Pointer to Value to write
END_VAR
```

Backing up parameters

Brief description

All parameters registered in the parameter system can be saved on the CF card.

This command is executed via the **S-0-0264** parameter. A file name can be specified for this method.

If an empty string is transferred, a file name as "ControllerCF<AxisNumber>.bin" is created.

The following elements are backed up:

- EIDN
- High word of the attribute
- Data

- CRC16 checksum

Signature

```
METHOD dwSaveToCF : DWORD
VAR_INPUT
    StrFilename : STRING := '';
END_VAR
```



This functionality can take quite some time. Thus, a watchdog can occur.

This function should be used asynchronously with parameter access.

Restoring parameters

Brief description

The parameters backed up via the command of S-0-0264 are restored with the command **S-0-0263**.

When calling, a file can be specified. In case of an empty string, a generic string is used.



Only parameters already registered in the parameter system can be restored.

Signature

```
METHOD dwLoadFromCF : DWORD
VAR_INPUT
    StrFilename : STRING := '';
END_VAR
```



This functionality can take quite some time. Thus, a watchdog can occur.

This function should be used asynchronously with parameter access.

Loading default values

Use the command **S-0-0262** to restore the default values. It is configured via the parameter P-0-4090.



The default values are not defined for list parameters.

The configuration can be executed in the PLC by the property **LoadDefaultSettings**. Currently, only the value LOAD_DEFAULT (:= 1) is allowed.

Using the **dwLoadDefaultParam()** method, it can also be executed in the PLC. First, there is an attempt to write the scaling parameters.

Therefore, the **LoadDefaultParam** method of the parameter is first called with the option K_LOADDEFAULT_WEIGHTING_ONLY and subsequently with K_LOADDEFAULT_NO_OPTIONS ([Administration, page 502](#)).



This functionality can take quite some time. Thus, a watchdog can occur.

This function should be used asynchronously with parameter access.

Creating parameter lists

Lists of all parameters

Brief description The parameter system provides a list with all registered parameters via the parameter **S-0-0017**.

The list can also be read out via the PLC using the "dwCreateListAllParam" method.

Signature

```
METHOD dwCreateListAllParam : DWORD
VAR_INPUT
    upData:    PARAM upData;
    puiLength: POINTER TO UINT;
END_VAR
```



This functionality can take quite some time. Thus, a watchdog can occur.

This function should be used asynchronously with parameter access.

Lists of all commands

Brief description A list of the registered commands can be obtained via the parameter **S-0-0025**.

The list can also be read out via the **dwCreateListAllCommands** method using the PLC.

Signature

```
METHOD dwCreateListAllCommands : DWORD
VAR_INPUT
    upData:    PARAM upData;
    puiLength: POINTER TO UINT;
END_VAR
```



This functionality can take quite some time. Thus, a watchdog can occur.

This function should be used asynchronously with parameter access.

Lists of all parameters backed up on the CF Card

Brief description The list with parameters to be stored on the CF card can be obtained via parameter **S-0-0192**

The list can also be read out via the **dwCreateListSaveOnCF** method using the PLC.

Signature

```
METHOD dwCreateListSaveOnCF : DWORD
VAR_INPUT
    upData:    PARAM upData;
    puiLength: POINTER TO UINT;
END_VAR
```



This functionality can take quite some time. Thus, a watchdog can occur.

This function should be used asynchronously with parameter access.

Checksum of the parameters to be backed up

Brief description

When reading this parameter, the checksum is formed via the "IDN list of the operating data to the backed up" (**S-0-0192**).

The checksum can also be formed via the **dwCreateChecksumListSaveOnCF** method using the PLC.

Program:

```
METHOD dwCreateChecksumListSaveOnCF : DWORD
VAR_INPUT
    pdwChecksum : PARAM_upData;
END_VAR
```



This functionality can take quite some time. Thus, a watchdog can occur.

This function should be used asynchronously with parameter access.



To determine parameter change, two checksum values read at different point in times can be compared.

If they differ, the value of at least one parameter listed in the "IDN list of operating data to be backed up" is changed at runtime.

Synchronizing parameter values

Brief description

Available from ML_ParamBase 12.4.0.0

Certain parameters can be read and written with the read value using this method. This method might be called again, e.g. after switching the scaling.

The transfer parameter **iType** determines the parameters to be synchronized. If the value **K_WEIGHTING_ALL** is used, all parameters are synchronized. Otherwise, only parameters in which **iType** is identical to **iGetParamType()** are synchronized.

Signature

```
METHOD dwReadWriteParamType : DWORD
VAR_INPUT
    iType : INT := K_WEIGHTING_ALL;
END_VAR
```

Cyclic write protection

The method sets the parameter property cyclic command values to get. The parameter is identified via the parameter number.

For more information, refer to [Administration, page 502](#).

The **eiEidn** input identifies the parameter.

Signature

```
METHOD dwSetCyclicConfigured : DWORD
VAR_INPUT
    eiEidn : DWORD;
    boValue : BOOL;
END_VAR
```

11.2.2 PARAM_SercosParameter

General information

This class is the basic implementation of a parameter. Its use is explained in the following.

Interfaces

Usage

The parameter is provided with three interfaces to be able to execute special checks and processes.

One function block can be assigned to the parameter for each interface. The function block implements this interface and the desired method can be executed.

For a command parameter, only the command interface is accessed.

The two remaining interfaces are used for normal parameter usage for read-only and write request.

The respective pointer of the parameter is also transferred. This provides the pointer with information on the parameter.

Read interface

Brief description The **I_PARAM_ReadAccess** interface is provided with the **dwReadData** method.

Signature

```
METHOD dwReadData : DWORD
VAR INPUT
  // Pointer to parameter to modify, read, ...
  poParam: POINTER TO PARAM_SercosParameter;
  // Pointer to data to write
  upData: PARAM_upData;
END_VAR
```

Write interface

Brief description The **I_PARAM_WriteAccess** interface is provided with the **dwReadData** method.

Signature

```
METHOD dwWriteData : DWORD
VAR INPUT
  // Pointer to parameter to write
  poParam: POINTER TO PARAM_SercosParameter;
  // Pointer to data
  upData: PARAM_upData;
  // Accessmode
  eAccessMode: PARAM_Access;
END_VAR
```

Command interface

Brief description The **I_PARAM_CmdAccess** interface is provided with the "dwCmdInit", "dwCmdCyclic" and "dwCmdAbort" methods.

The command is initialized in the **dwCmdInit** method. (Initialization of the process block, opening of files, ...)

dwCmdInit signature

```
METHOD dwCmdInit : SERCOS_CMD_RETURN
VAR INPUT
  // Pointer to parameter
  poParam: POINTER TO PARAM_SercosParameter;
  // Pointer to process data block
  puPDB: POINTER TO PARAM_uPDB;
END_VAR
```

The command is cyclically processed in the **dwCmdCyclic** method.

dwCmdCyclic signature

```

METHOD dwCmdCyclic : SERCOS_CMD_RETURN
VAR INPUT
  // Pointer to parameter
  poParam:    POINTER TO PARAM_SercosParameter;
  // Pointer to process data block
  puPDB:      POINTER TO PARAM_uPDB;
END_VAR

```

When resetting the command, the **dwCmdAbort** function is called. This is used for cleaning purposes. (Closing files...)

dwCmdAbort signature

```

METHOD dwCmdAbort : SERCOS_CMD_RETURN
VAR INPUT
  // Pointer to parameter
  poParam:    POINTER TO PARAM_SercosParameter;
  // Pointer to process data block
  puPDB:      POINTER TO PARAM_uPDB;
END_VAR

```

Attributes

Overview The PARAM_SercosParameter function block is provided with the following public attributes.

Class variables: PARAM_SercosParameter

	Data type	Description
c_dwDataStatus	DWORD	Indicates the error code of the parameter. No error = 0
c_dwEidn	DWORD	EIDN of the parameter
c_paName	POINTER TO PARAM_Language_Array	Parameter name
c_ppdwAttribute	POINTER TO POINTER TO DWORD	Parameter attribute. Analog to sercos attribute
c_ppaUnit	POINTER TO POINTER TO PARAM_Language_Array	Parameter unit
c_puMin	POINTER TO PARAM_uData	Minimum value of the parameter
c_puMax	POINTER TO PARAM_uData	Maximum value of the parameter
c_upActData	PARAM_uData	Parameter value
c_uDefault	PARAM_uData	Default value
itfWrite	I_PARAM_WriteAccess	Interface for writing method
itfRead	I_PARAM_ReadAccess	Interface for reading method
itfCmd	I_PARAM_CmdAccess	Interface for command execution

c_paName The variable references to a global array using the pointers on the translated names ([Language, page 507](#)). This saves memory space in case of several controller axes.

c_ppdwAttribute The variable for the attribute points to a pointer that points to an attribute. This allows the creation of scaling-dependent attributes. The pointer has to be in the function block instantiating the parameter. The attribute can be stored globally.

c_ppaUnit The unit is displayed language-dependent as the name. The same data structure is used. But in contrast to the names, a pointer references to it. Thus, a scaling dependency can also be created.

c_puMin, c_puMax	Since only individual list elements are checked in lists, the extreme value is only an individual variable. To identify that a parameter is not provided with extreme values, the value IL_NULL can be assigned.
c_upActData	The parameter date is saved in the function block in which the parameter was instantiated. The pointer on it is saved in this variable.
c_uDefault	The default value is saved as single value.



The default values are not defined for list parameters.

Access methods

Read data status

Brief description

The data status indicates the parameter validity. This is valid with a value of 0.

Signature

```
METHOD dwGetDataStatus : SERCOS_ERROR
VAR INPUT
  d_data : POINTER TO PARAM_uData; // Pointer to data to write (dwData)
END_VAR
```

Reading IDN

Brief description

The IDN of a parameter can be read via the following method.

Signature

```
METHOD dwGetIDN : SERCOS_ERROR
VAR INPUT
  d_data : PARAM_upData; //Object contains idn (PARAM_uData.eiData)
END_VAR
```

Reading name

Brief description

The parameter name is read via this method.

Signature

```
METHOD dwGetName : SERCOS_ERROR
VAR INPUT
  // Pointer to Data to write
  pstrData : POINTER TO PARAM_rStringData;
  // Size of date incl. Header
  puiLength : POINTER TO UINT;
  // Read Name in this language
  eLanguage : PARAM_Language := PARAM_Language.PARAM_LANG_DE;
END_VAR
```

Reading attribute

Brief description

The sercos attribute can be obtained using this method.

Signature

```
METHOD dwGetAttribute : SERCOS_ERROR
VAR INPUT
  // pointer to data to write (PARAM_uData.dwData)
  puData : POINTER TO PARAM_uData;
END_VAR
```

Reading minimum value

Brief description

The minimum value can be obtained using this method.

Signature

```
METHOD dwGetMinValue : SERCOS_ERROR
VAR INPUT
  // pointer to data to write (PARAM_uData.<Datatype>)
```

ML_ParamBase.library

```

puData      : POINTER TO PARAM_uData;
// pointer to data to set written size
puiLength   : POINTER TO UINT;
END_VAR

```

Reading maximum value

Brief description The minimum value can be obtained using this method.

Signature

```

METHOD dwGetMaxValue : SERCOS_ERROR
VAR_INPUT
// pointer to data to write (PARAM_uData.<Datatype>)
puData      : POINTER TO PARAM_uData;
// pointer to data to set written size
puiLength   : POINTER TO UINT;
END_VAR

```

Reading date

Brief description To read the date, the interface to the outside is defined as follows.

While reading, it is checked whether a read-only interface is registered. If this is the case, the value is written in the **upDate** union. Otherwise, the content of **c_upActData** is copied.

The access type is ignored.

Signature

```

METHOD dwGetOperationData : SERCOS_ERROR
VAR_INPUT
// Pointer to data to write (<Datatype>)
upData:    PARAM_upData;
// written size
puiLength: POINTER TO UINT := IL_NULL;
// access type
eAccess:   PARAM_Access    := PARAMETER_ACCESS;
END_VAR

```

Writing date

Brief description The following method is provided to write a date.

This method checks whether it is a parameter or a command.

If it is a command, it is checked whether the required interface is registered. If this is the case, the method is called. Otherwise, the error code K_STS_DATA_NO_ACCESS is returned.

In case of a parameter, the write interface is opened if available. Alternatively, **c_upActData** is copied. Even though the parameter can be provided with extreme values, it cannot be specified to mandatorily execute an extreme value check.

The access type is ignored.

Signature

```

METHOD dwSetOperationData : SERCOS_ERROR
VAR_INPUT
upData:    PARAM_upData;    // Pointer with data to write
puiLength: POINTER TO UINT;  // size to write
eAccess:   PARAM_Access     := PARAMETER_ACCESS; // access type
END_VAR

```

Administration**Loading default values**

Brief description To load the default values, the parameter is provided with the method mentioned below. This method only operates if called with the option K_LOADDEFAULT_NO_OPTIONS.

Signature

```
METHOD dwLoadDefaultData : DWORD // Options to load default data
VAR_INPUT
    dwOptions: DWORD := K_LOADDEFAULT_NO_OPTIONS;
END_VAR
```

Parameter type

Brief description Available from ML_ParamBase 12.4.0.0
This method returns the value K_WEIGHTING_NONE.

Signature

```
METHOD iGetParamType : INT
VAR_INPUT
END_VAR
```

Cyclic write protection

For parameters returning a cyclic command value specification, write protection can be added via parameter access.

This information can be set using the following method. If **boValue** is TRUE, the parameter is marked as cyclically configured. The error code K_STS_DATA_CONFCYCLIC ([SERCOS_ERROR](#), [page 509](#)) is returned when writing the parameter in phase 4.

Signature

```
METHOD dwSetCyclicConfigured : DWORD
VAR_INPUT
    boValue : BOOL := FALSE;
END_VAR
```

11.2.3 PARAM_ExtSercosParameter

General information

The function block PARAM_ExtSercosParameter inherits the properties of [PARAM_SercosParameter](#) , [page 498](#). Thus, only the differences and extensions are shown.

Attributes

Class variables
PARAM_ExtSercosParameter

The function block PARAM_ExtSercosParameter is provided with the following inputs in addition to the function block PARAM_SercosParameter.

Variable name	Data type	Description
c_combination	POINTER TO PARAM_uData	Valid bit combinations
c_dwControl	DWORD	Extended control word of the parameter
c_upParamData	PARAM_upData	Parameter value (written by parameter system)
c_upPropData	PARAM_upData	Parameter value (written by Property)
c_iParamType	INT	Parameter classification
c_plrScalingFactor	POINTER TO LREAL	Scaling factor

c_combination A pointer can be stored on an array of valid combinations in this variable. Array element 0 contains the query mask. In the last array element, all bits are set to 1.

Validity is only checked if the **PARAM_ESP_CHECKCOMB** flag is set in the extended control word.

c_dwControl

Different options can be set using this control word.

Control flags

Flag	Significance
PARAM_ESP_CHECKEXTREM	Check extreme values
PARAM_ESP_CHECKCOMB	Check combination
PARAM_ESP_VAR_MIN	Parameter is provided with a minimum value whose value corresponds to another parameter date.
PARAM_ESP_VAR_MAX	Parameter is provided with a maximum value whose value corresponds to another parameter date.
PARAM_ESP_MEMORY_CF	Save parameter on CF card.
PARAM_ESP_WRITEPROTECTION_P2	Parameter can be written in P2
PARAM_ESP_WRITEPROTECTION_P4	Parameter can be written in P4

c_upParamData, c_upPropData

As for the attribute **c_upActData**, only the pointer is saved.

The value IL_NULL can be assigned to both the pointers (e.g. in case of an actual value).

c_iParamType

A parameter can be classified using these variables, i.e. if it is scaling-independent, specifies the scaling or how it is scaled.

Overview on parameter types

Constant	Description
K_WEIGHTING_NONE	Scaling-independent parameter
K_WEIGHTING_PARAM	Scaling-determined parameter

User-defined scalings can be created. Their values must be greater than 0.

c_plrWeightingFactor

If a scaling factor is set, it is considered when reading/writing the active date (**c_upActdata**).

Access methods

Reading date

Brief description

While reading, it is checked whether a read-only interface is registered. If this is the case, the value is written in the **upDate** union. Otherwise, the respective date is read with regard to the access path. If neither the property date nor the parameter date exists, **c_upActData** is used.

Signature

```
METHOD dwGetOperationData : SERCOS_ERROR
VAR_INPUT
    upData:    PARAM_upData;           // Pointer to data to write
    puiLength: POINTER TO UINT := IL_NULL; // written size
    eAccess:    PARAM_Access := PARAMETER_ACCESS; // access type
END_VAR
```

Writing date

Brief description

This method checks whether it is a parameter or a command.

If it is a command, it is checked whether the required interface is registered. If this is the case, the method is called. Otherwise, the error code K_STS_DATA_NO_ACCESS is returned.

In case of a parameter, the write interface is opened if available. Otherwise, the respective date is written with regard to the access path. If neither the property date nor the parameter date exists, **c_upActData** is used.

Signature

```
METHOD dwSetOperationData : SERCOS_ERROR
VAR_INPUT
  upData:    PARAM_upData;      // Pointer with data to write
  puiLength: POINTER TO UINT;   // size to write
  eAccess:   PARAM_Access := PARAMETER_ACCESS; // access type
END_VAR
```

Administration

Loading default values

Brief description

To load the default values, the parameter is provided with the method mentioned below. The method can be called with the options K_LOADDEFAULT_NO_OPTIONS and K_LOADDEFAULT_WEIGHTING_ONLY

Signature

```
METHOD dwLoadDefaultData : DWORD // Options to load default data
VAR_INPUT
  dwOptions: DWORD := K_LOADDEFAULT_NO_OPTIONS;
END_VAR
```

Parameter type

Brief description

Available from ML_ParamBase 12.4.0.0

This method returns the variable value "c_iParamType" ([Class variables PARAM_ExtSercosParameter, page 503](#)).

Signature

```
METHOD iGetParamType : INT
VAR_INPUT
END_VAR
```

11.2.4 PARAM_CmdExecuter

Brief description

This function block processes command sequences. A state machine is used.

The commands are processed subsequently and depending on their state. Thus, the respective method of **I_PARAM_CmdAccess** is called.

The return value of the last method called and the user inputs control the state machine during the command processing ([fig. 11-2 "State machine of the command handler" on page 507](#)).

If a command is in the "K_STATE_NOT_ACTIVE" state, it is deleted from the memory.

Methods

Adding command

The **dwAddCmd** method is used to add a command.

This method checks whether another command can be included in the administrative structure and has not yet been entered. If both is the case, the "Init" method of the command is called.

Signature

```
METHOD dwAddCmd : SERCOS_ERROR
VAR INPUT
  // Pointer to Parameter
  pParam : POINTER TO PARAM_SercosParameter;
  // Interface to work parameter
  itfCmd : I_PARAM_CmdAccess := IL_NULL;
  // Control Cmd: decide next state
  usiCC : USINT := 2#11;
END_VAR
```

Return value	Description
K_STS_DATA_INVALID	Function parameter invalid
K_STS_DATA_CMDACTIVE	Command already active
K_STS_DATA_CMDNOTEXEETIME	Array full. Command can currently not be executed.
K_STS_DATA_CMDNOTEXEETIME (parameter status is K_STATE_ERROR)	"Init" method not successfully executed
K_STS_OK	Method successfully executed

Tab. 11-1: Return values

Clearing command memory

This method clears the internal memory. The function is automatically executed by the function block during initialization.
The method returns K_STS_OK.

Signature

```
METHOD dwClearStorage : SERCOS_ERROR
VAR INPUT
END_VAR
```

Controlling command The CC (Control Command) can be set to the command using this method.

Signature

```
METHOD dwSetControlCmd : SERCOS_ERROR
VAR INPUT
  // Pointer to Parameter
  pParam : POINTER TO PARAM_SercosParameter;
  // CC Controlvalue
  usiCC : USINT;
END_VAR
```

 When calling this method, the respective parameter value is not set.

11.3 Command handling

A command handling was created in the parameter system according to the sercos specification. This allows a parallel processing of several commands in a common task.

Therefore, it is required to configure the program "HydrControlCmdProg" in a free-running task. The program is provided in this library. The task cyclically calls the "PARAM_CmdExecuter" function block ([PARAM_CmdExecuter](#), [page 505](#)).

The concept intends that each function block used for command processing has to implement the "I_PARAM_CmdAccess" interface. The individual meth-

ods of this interface control the sequence of the sercos command state machine using different return values. To control the state machine, the respective parameters can be written. The respective entries are represented by **CC**. The command in the parameter status is represented in the following figure.

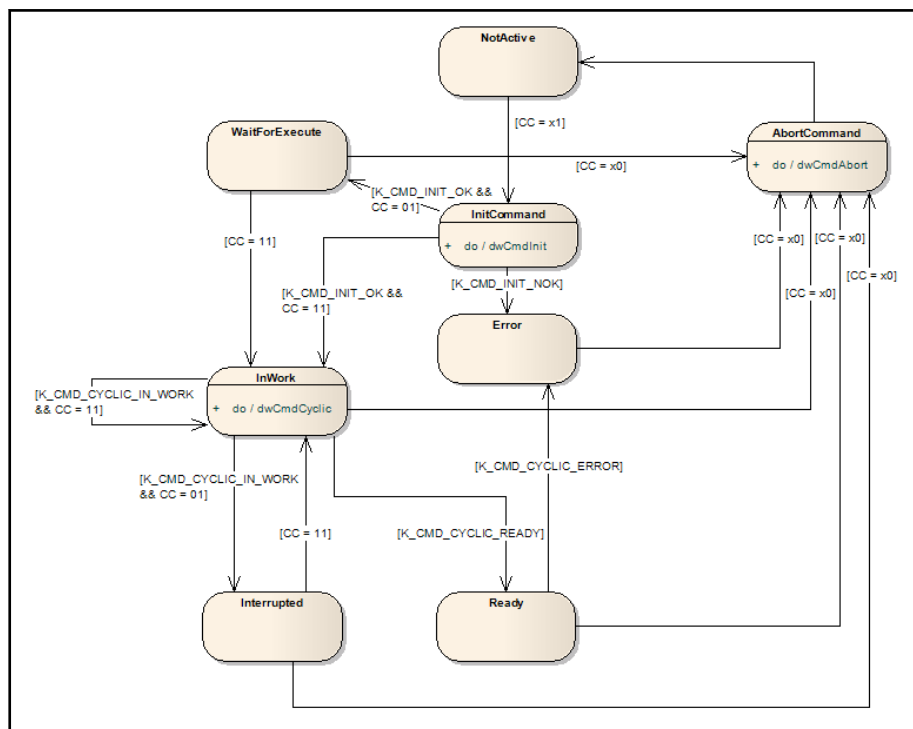


Fig. 11-2: State machine of the command handler

Since the command handler is supposed to simultaneously process several commands, the command has to create several time slots. This behavior is achieved via the return values of the "dwCmdCyclic" method.

A command is automatically added to the processing if the respective parameter is labeled as command and a command interface is specified.

11.4 Data types

11.4.1 Language

PARAM_Language *Type declaration*

```

TYPE PARAM_Language :
(
    PARAM_LANG_DE := 0, (* German, 0 *)
    PARAM_LANG_EN := 1, (* English, 1 *)
    PARAM_LANG_FR := 2, (* French, 2 *)
    PARAM_LANG_ES := 3, (* Spanish, 3 *)
    PARAM_LANG_IT := 4  (* Italian, 4 *)
) WORD;
END_TYPE

```

The available languages are listed in this enumeration.

The numbers correspond to the definition of the S-0-0265 parameter.

PARAM_Language_Array *Type declaration*

```

TYPE PARAM_Language_Array :
    ARRAY[PARAM_LANG_DE..PARAM_LANG_IT] OF POINTER TO STRING;
END_TYPE

```

The array definition saves name and unit of the parameter with regard to the language. In these cases, the parameter accesses the element which is assigned to the language set.

11.4.2 Data access

PARAM_uData This union contains all required data types and their pointers. To avoid confusion, the definition is not explained in detail.

PARAM_rStringData To access a string element (name, unit) or string parameter, use this variable definition.

According to the sercos standard, the data contains a header with the current and maximum length. Both values are identical for this usage.

Type declaration

```
TYPE PARAM_rStringData :
  STRUCT
    uiActLen: UINT;    (* Actual Length *)
    uiMaxLen: UINT;    (* Maximal Length *)
    strData:  STRING;  (* Data as String *)
  END_STRUCT
END_TYPE
```

PARAM_rListData To access list parameters, the required memory can be reserved with this variable. Analog to type **PARAM_rStringData**, the structure contains the sercos-specific header.

Type declaration

```
TYPE PARAM_rListData :
  STRUCT
    uiActLen: UINT;    (* Actual Length *)
    uiMaxLen: UINT;    (* Maximal Length *)
    rData:    PARAM_uArray; (* Data *)
  END_STRUCT
END_TYPE
```



When declaring a variable of this type, approximately 65,532 bytes of memory are reserved.

PARAM_upData For data access with unknown or variable data size, a variable of this type can be used as transfer value.

Type declaration

```
TYPE PARAM_upData :
  UNION
    puData:    POINTER TO PARAM_uData;    // Singleparameter
    pStringData: POINTER TO PARAM_rStringData; // Stringparameter
    pListData:  POINTER TO PARAM_rListData;  // Listparameter
  END_UNION
END_TYPE
```

PARAM_uArray The required data type can now be assigned to the respective pointer. This union defines arrays with different data widths so that the complete size of the array always corresponds to 65 532 (64 kB minus header size).

The individual data types from **PARAM_u1Byte** to **PARAM_u8Byte** are unions with data types of respective size.

Type declaration

```
TYPE PARAM_uArray :
  UNION
    ar8Byte: ARRAY[0.. 8191] OF PARAM_u8Byte; // Num. of Elem., size 8 Byte
    ar4Byte: ARRAY[0..16382] OF PARAM_u4Byte; // Num. of Elem., size 4 Byte
    ar2Byte: ARRAY[0..32765] OF PARAM_u2Byte; // Num. of Elem., size 2 Byte
    ar1Byte: ARRAY[0..65531] OF PARAM_u1Byte; // Num. of Elem., size 1 Byte
    str : STRING(65532);
  END_UNION
```

```
END_UNION
END_TYPE
```



When declaring a variable of this type, approximately 65,532 bytes of memory are reserved.

11.5 Auxiliary functions

dwGetEidnS *signature*

```
FUNCTION dwGetEidnS : DWORD
VAR_INPUT
    uIDN : UINT;
END_VAR
```

This function can create the EIDN of an S-parameter without SE/SI.

dwGetEidnS_SESI *Signature*

```
FUNCTION dwGetEidnS_SESI : DWORD
VAR_INPUT
    uIDN : UINT;
    usiSI : USINT;
    usiSE : USINT;
END_VAR
```

This function can create the EIDN of an S-parameter with SE/SI.

dwGetEidnP *signature*

```
FUNCTION dwGetEidnP : DWORD
VAR_INPUT
    uIDN : UINT;
END_VAR
```

This function can create the EIDN of a P-parameter without SE/SI.

dwGetEidnP_SESI *signature*

```
FUNCTION dwGetEidnP_SESI : DWORD
VAR_INPUT
    uIDN : UINT;
    usiSI : USINT;
    usiSE : USINT;
END_VAR
```

This function can create the EIDN of a P-parameter with SE/SI.

11.6 Constants

General constants

Name	Significance
usiSIZE_OF_SERCOSHEADER	Constant for the size of the sercos-specific header (Data access, page 508).
STRING_NO_UNIT	Constant to show that a parameter is not provided with any unit.

SERCOS_ERROR

Error code	Value	Description
K_STS_OK	16#0000	Operation successful
K_STS_IDN_NONE	16#1001	IDN invalid
K_STS_IDN_INVALIDACCESS	16#1009	Invalid access to element 1

Error code	Value	Description
K_STS_NAME_NONE	16#2001	Name does not exist
K_STS_NAME_TOOSHORT	16#2002	Name transfer too short
K_STS_NAME_TOOLONG	16#2003	Name transfer too long
K_STS_NAME_READONLY	16#2004	Name is read-only
K_STS_NAME_WRITEPROTECT	16#2005	Name is currently write-protected
K_STS_ATTR_NONE	16#3001	Attribute does not exist
K_STS_ATTR_TOOSHORT	16#3002	Attribute transfer too short
K_STS_ATTR_TOOLONG	16#3003	Attribute transfer too long
K_STS_ATTR_READONLY	16#3004	Attribute is read-only
K_STS_ATTR_WRITEPROTECT	16#3005	Attribute is currently write-protected
K_STS_UNIT_NONE	16#4001	Unit does not exist
K_STS_UNIT_TOOSHORT	16#4002	Unit transfer too short
K_STS_UNIT_TOOLONG	16#4003	Unit transfer too long
K_STS_UNIT_READONLY	16#4004	Unit is read-only
K_STS_UNIT_WRITEPROTECT	16#4005	Unit is currently write-protected
K_STS_MIN_NONE	16#5001	Minimum value does not exist
K_STS_MIN_TOOSHORT	16#5002	Minimum value transmitted too short
K_STS_MIN_TOOLONG	16#5003	Minimum value transmitted too long
K_STS_MIN_READONLY	16#5004	Minimum value is read-only
K_STS_MIN_WRITEPROTECT	16#5005	Minimum value is currently write-protected
K_STS_MAX_NONE	16#6001	Maximum value does not exist
K_STS_MAX_TOOSHORT	16#6002	Maximum value transmitted too short
K_STS_MAX_TOOLONG	16#6003	Maximum value transmitted too long
K_STS_MAX_READONLY	16#6004	Maximum value is read-only
K_STS_MAX_WRITEPROTECT	16#6005	Maximum value is currently write-protected
K_STS_DATA_NONE	16#7001	Data does not exist
K_STS_DATA_TOOSHORT	16#7002	Data transfer too short
K_STS_DATA_TOOLONG	16#7003	Data transfer too long
K_STS_DATA_READONLY	16#7004	Data is read-only
K_STS_DATA_WRITEPROTECT	16#7005	Data is currently write-protected
K_STS_DATA_SMALLER	16#7006	Data is smaller than minimum value
K_STS_DATA_GREATER	16#7007	Data is greater than maximum value
K_STS_DATA_INVALID	16#7008	Data is invalid (e.g. bit combination)
K_STS_DATA_PASSWORD	16#7009	Data is password-protected
K_STS_DATA_CONFCYCLIC	16#700A	Parameter is cyclically configured
K_STS_DATA_INVALIDINDEX	16#700B	Index is invalid

Error code	Value	Description
K_STS_DATA_WRITEPROTECTOS	16#700C	Parameter is write-protected
K_STS_DATA_ISNAN	16#700D	Date is Not a Number (NaN)
K_STS_DATA_CMDACTIVE	16#7010	Command already active
K_STS_DATA_CMDNOTINTERR	16#7011	Command cannot be interrupted
K_STS_DATA_CMDNOTEXEETIME	16#7012	Command can currently not be executed
K_STS_DATA_CMDNOTEXECPARAM	16#7013	Command cannot be executed due to incorrect parameters
K_STS_DATA_MAXRANGEEXCEED	16#710C	Data exceeds maximum range
K_STS_DATA_NO_ACCESS	16#8001	Currently access not possible

SERCOS_CMD_STATES

Status of active command	Value	Description
K_STATE_NOT_ACTICE	2#0000	Command not active
K_STATE_WAIT_FOR_EXEC	2#0001	Command is set. Waits for execution
K_STATE_READY	2#0011	Command is executed. Waits for reset
K_STATE_INTERRUPTED	2#0101	Command is interrupted. Waits for execution or abort
K_STATE_IN_WORK	2#0111	Command active
K_STATE_ERROR	2#1111	Command aborted with errors

SERCOS_CMD_RETURN

Return value	Significance
K_CMD_INIT_OK	Command Init function completed without error
K_CMD_INIT_NOK	Command Init function completed with error
K_CMD_CYCLIC_IN_WORK	Cyclic command call interrupted (time slot)
K_CMD_CYCLIC_READY	Cyclic command call completed without error
K_CMD_CYCLIC_ERROR	Cyclic command call completed with error
K_CMD_ABORT_OK	Command Abort function completed without error
K_CMD_ABORT_NOK	Command Abort function completed with error

12 Service and support

Our worldwide service network provides an optimized and efficient support. Our experts offer you advice and assistance should you have any queries. You can contact us **24/7**.

Service Germany Our technology-oriented Competence Center in Lohr, Germany, is responsible for all your service-related queries for electric drive and controls.

Contact the **Service Hotline** and **Service Helpdesk** under:

Phone:	+49 9352 40 5060
Fax:	+49 9352 18 4941
E-mail:	service.svc@boschrexroth.de
Internet:	http://www.boschrexroth.com

Additional information on service, repair (e.g. delivery addresses) and training can be found on our internet sites.

Service worldwide Outside Germany, please contact your local service office first. For hotline numbers, refer to the sales office addresses on the internet.

Preparing information To be able to help you more quickly and efficiently, please have the following information ready:

- Detailed description of malfunction and circumstances
- Type plate specifications of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your e-mail address)

Index

A

About this documentation.....	1
Validity of the documentation.....	1
Access to data of a field bus drive.....	361
Access to data of an axis	
Profibus.....	387
Access to data of an axis, control or another object.....	128
Access to data of an axis, control, kinematics (axis group) or another object.....	278
Access to oscilloscope data via function variable access.....	107
Active.....	12
ANSI Z535.6-2006.....	2
APP_INFO.....	159
Axis data via function variable access.....	87
Axis data via shared memory and function variable access, general.....	83
Axis data via synchronous function variable access in C2C link.....	90
Axis types and function blocks supported by these types.....	355
AXIS_GROUP_REF.....	29
AXIS_REF.....	27, 363
Application of field bus drive.....	424, 468
AxisData[] - Axis data in shared memory.....	85
AxisData[],	
Application of the field bus drive.....	426, 470
AxisInterface	
Application, MB_GearInPos.....	249
Application, MB_MotionProfile.....	215
Application, MC_CamIn.....	209
Application, MC_GearIn.....	245
Application, ML_FlexProfile.....	230

B

BASELIBCONST.....	126
Basic class	
ML_ControllerBase.....	473

C

Command handling.....	506
CommandAborted.....	12
Control data via function variable access.....	94
Control data via shared memory and function variable access.....	92
ControlData[] - control data in shared memory.....	93
Controller data.....	474
Controller management.....	476
Connect controller.....	476
Cyclic procedure of the controller.....	478
Cyclic procedure of the controller management.....	477
Operation mode handling.....	478

Predefined generators of the operation modes.....	481
State machine: operating state.....	477
CONTROLS.....	27

D

Data types for synchronization in the position-controlled synchronous operation modes....	56
Data types to diagnose errors in the Motion function blocks.....	60
Data types to reference objects of Motion function blocks.....	26
Data types to report date and absolute time specifications for Motion function blocks.....	63
Data types to support cams, MotionProfiles and FlexProfiles.....	32
Data types to support MotionProfile and FlexProfile functions.....	76
Data types to support touch probe functions.....	64
Documentation	
Change record.....	1
Done.....	12
DV_Axis[].....	87
DV_Control[].....	94
DV_Osci[].....	108
DV_Pls[].....	100
DV_Probe[].....	105
dwGetEidnP.....	509
dwGetEidnP_SESI.....	509
dwGetEidnS.....	509
dwGetEidnS_SESI.....	509

E

E_LOG_CLASS.....	120
E_LOG_FACILITY.....	121
E_LOG_SEVERITY.....	121
E_SCALING_TYPE.....	159
E_SYSTEM_SPECIAL_PATH.....	159
Enable.....	12
Error tables	
MLC_TABLE.....	360
SERCOS_TABLE.....	357
ERROR_CODE.....	60
ERROR_STRUCT.....	61
ERROR_TABLE.....	62
Error_Tables	
MLC_TABLE.....	360
SERCOS_TABLE.....	357
Execute.....	12
ExecuteLock.....	12
Executing files.....	122
EXT_SYS_TIME_DATE.....	63

F

Field bus drive	

Index

- Acyclic parameter communication..... 426, 470
- AXIS_REF..... 424, 468
- AxisData[]..... 426, 470
- Bus device address..... 422, 467
- Communication-required settings..... 422, 466
- Create in project..... 421, 465
- Cyclic data exchange..... 426, 470
- Include libraries..... 423, 467
- IndraDrive communication settings..... 422, 466
- Length of parameter channel..... 422, 467
- Operation mode settings..... 423, 467
- PLC program, required declaration..... 424, 468
- Process data and profile type..... 422, 467
- Reference drive..... 423, 467
- Signal control word..... 423, 467
- Signal status word..... 423, 467
- FlexProfile
 - Event control..... 240
 - Processing types..... 236
 - Select coordinate system..... 233
 - Switch..... 233
 - Synchronize..... 233
- Function block prefixes..... 361, 429
- Function block types..... 11
- Function block types and their interfaces..... 12
- Function blocks for controlled motion..... 266
- Function blocks for multi-axis control..... 200
- Function blocks for parameter access
 - Profibus..... 385
- Function blocks for single-axis control..... 179
- Function blocks for the axis group control..... 251
- Function blocks for the parameter access
 - Sercos..... 278
- Function blocks for the use of the cyclic data channel..... 253
- Function blocks for the use of the touch probe function..... 333
- Function variable access
 - DV_Axis[]..... 87
 - DV_Control[]..... 94
 - DV_Osci[]..... 108
 - DV_Pls[]..... 100
 - DV_Probe[]..... 105
- Function variable access in C2C link
 - GV_LinkAxis[]..... 90
- G**
 - General information on controlled motion..... 266
 - General information on using the cyclic data channel..... 253
 - GV_LinkAxis[]..... 90
 - GVL_A_GlobalVarIdx..... 127
 - GVL_A_Param_ID..... 130
 - GVL_Base..... 125
 - GVL_BaseConstant..... 126
 - GVL_C_Param_ID..... 130
 - GVL_FieldBusDrives..... 364, 425, 439, 469
 - GVL_M_Param_ID..... 130
 - GVL_N_Param_ID..... 131
 - GVL_O_Param_ID..... 131
 - GVL_P_Param_ID..... 132, 362
 - GVL_S_Param_ID..... 132, 361
- H**
 - Hazard warnings..... 2
 - Helpdesk..... 513
 - Hotline..... 513
- I**
 - I_PARAM_CmdAccess
 - dwCmdAbort..... 500
 - dwCmdCyclic..... 500
 - dwCmdInit..... 499
 - I_PARAM_ReadAccess
 - dwReadData..... 499
 - dwWriteData..... 499
 - I/O addressing – RMB_PLCOpenEther-
 - CAT_SoE..... 427, 471
 - I/O addressing – RMB_PlcOpenFieldbus:..... 426, 470
 - IActualValue..... 473
 - getIActualValue..... 474
 - IAdditionalSetValue
 - getIAdditionalSetValue..... 474
 - IDigitalIO
 - getIDigitalIO..... 474
 - IL_NULL..... 26
 - IL_SercosAttribute..... 133
 - IL_SercosElementsToParaNo..... 135
 - IL_SercosStringToParaNo..... 137
 - IL_SetDateTime..... 146
 - IL_TARGET_NAME..... 126
 - IModuleControl..... 473
 - getIModuleControl..... 474
 - IndraDrive
 - Bus device address..... 422, 467
 - Length of parameter channel..... 422, 467
 - Setting process data and profile type..... 422, 467
 - Signal control word..... 423, 467
 - Signal status word..... 423, 467
 - IndraWorks DS
 - Field bus drives settings..... 422, 466
 - Indraworks Engineering
 - Field bus drives settings..... 422, 466
 - INFO_APPLICATIONS..... 160
 - InGear..... 12
 - InOperation..... 12
 - Input description of the field bus process data..... 416, 420, 461, 464
 - Inputs and outputs of edge-controlled function blocks, final processing..... 16
 - Inputs and outputs of edge-triggered function blocks, permanent processing..... 22

Inputs and outputs of state-controlled function blocks, final processing.....	13	MB_ChangeProfileStep.....	222
Inputs and outputs of state-controlled function clocks, permanent processing.....	19	MB_ChangeProfileStep, special features of Sercos-based axes.....	224
InSync.....	12	MB_CHANGESTEP_MODE.....	76
IntelliSense		MB_Command.....	281
Axis data.....	85, 87	MB_Command, special features of Sercos-based axes.....	286
Control data.....	93, 94	MB_DIFF_EVAL.....	65
Oscilloscope data.....	108	MB_EVENT_ACTION.....	77
PLS data.....	98, 100	MB_EVENT_TRIGGER.....	78
Touch probe data.....	103, 105	MB_EXECUTION_MODE.....	78
IntelliSense of a function variable synchronously in the C2C link.....	90	MB_EXPECT_WINDOW_MODE.....	66
Interface IActualValue.....	476	MB_FLEX_EVENT.....	79
runActualValue.....	476	MB_GearInPos.....	248
Interface of the IModuleControl.....	476	MB_GearInPos, special features of Sercos-based axes.....	249
InitModule.....	476	MB_Home.....	199, 373
SwitchToOperationMode.....	476	MB_IDN.....	75, 362
SwitchToParamMode.....	476	MB_INTERPOLATION_MODE.....	66
Interface of the IOperationModeHandler.....	476	MB_MASTER_TYPE.....	79
bCyclic.....	476	MB_MotionProfile.....	212
bInitOperationCard.....	476	MB_MotionProfile, special features of Sercos-based axes.....	215
Interface of the IOutputData.....	476	MB_Phasing.....	203
runOutputData.....	476	Application, MB_GearInPos.....	249
InTorque.....	12	Application, MB_MotionProfile.....	215
InVelocity.....	12	Application, MC_CamIn.....	209
IOperationModeHandler.....	473	Application, MC_GearIn.....	245
getIOperationModeHandler.....	474	Application, ML_FlexProfile.....	230
IOutputData.....	473	MB_PhasingSlave.....	205
getIOutputData.....	474	Application, MB_GearInPos.....	249
IRetainHandler		Application, MB_MotionProfile.....	215
getIRetainHandler.....	474	Application, MC_CamIn.....	209
		Application, ML_FlexProfile.....	230
K		MB_POSITION_EVAL.....	67
KIN_OBJECTS.....	30	MB_PROBE_CONFIG.....	69
		MB_PROBE_CONFIG, special features of Sercos-based axes.....	69
L		MB_PROBE_DATA.....	68
Length of parameter channel, IndraDrive.....	422, 467	MB_PROBE_INPUT.....	67
Library		MB_PROBE_MODULE.....	67
ML_Base.....	71	MB_PROFILE_START_POINT.....	79
ML_ControllerBase.....	473	MB_PROFILE_STEP.....	81
ML_IPOImpactControl.....	483	MB_ReadLintParameter.....	286
ML_ParamBase.....	493	MB_ReadListParameter.....	288, 389
ML_PLCOpen.....	171	MB_ReadListSegParameterPtr.....	392
RIL_CommonTypes.....	25	MB_ReadLrealListParameter.....	291
RMB_PLCOpenEtherCAT_SoE.....	429	MB_ReadLrealParameter.....	294
RMB_PLCOpenFieldBus.....	361	MB_ReadMaxRealValue.....	295, 394
Logbook access.....	116	MB_ReadMaxValue.....	297, 396
		MB_ReadMinRealValue.....	299, 397
M		MB_ReadMinValue.....	301, 399
MB_ChangeCamData.....	225	MB_ReadName.....	302
MB_ChangeFlexEventSet.....	242	MB_ReadParameter.....	304, 400
MB_ChangeFlexProfileSet.....	239	MB_ReadRealParameter.....	306, 402
MB_ChangeProfileSet.....	218	MB_ReadSercosAttribute.....	308, 403
MB_ChangeProfileSet, special features of Sercos-based axes.....	220	MB_ReadSercosDataStatus.....	309
		MB_ReadStringParameter.....	311, 405

Index

MB_ReadStringParameterExt.....	313	convFromAxis_Velocity.....	475
MB_ReadUnit.....	315	convToAxis_Acceleration.....	475
MB_SERCOS_DATA_STATUS.....	82	convToAxis_Analog.....	475
MB_SetPositionControlMode.....	198	convToAxis_Force.....	475
MB_STEP_TYPE.....	81	convToAxis_Position.....	475
MB_Stop.....	371, 445	convToAxis_Velocity.....	475
MB_SYNC_TYPE.....	82	ML_Convert_DoublePrecision.....	475
MB_WriteLintParameter.....	316	convFromAxis_Acceleration.....	475
MB_WriteListParameter.....	318, 406	convFromAxis_Analog.....	475
MB_WriteListSegParameterPtr.....	409	convFromAxis_Force.....	475
MB_WriteLrealParameter.....	321	convFromAxis_Position.....	475
MB_WriteParameter.....	322, 411	convFromAxis_Velocity.....	475
MB_WriteRealParameter.....	324, 413	convToAxis_Acceleration.....	475
MB_WriteStringParameter.....	326, 414	convToAxis_Analog.....	475
MB_WriteStringParameterExt.....	327	convToAxis_Force.....	475
MC_AddAxisToGroup.....	251	convToAxis_Position.....	475
MC_CAM_ID.....	32	convToAxis_Velocity.....	475
MC_CamIn.....	207	ML_DAYSOFWEEK.....	150
MC_CamIn, special features of Sercos- based axes.....	209	ML_DIAGNOSIS_LOG.....	160
MC_CamOut.....	211	ML_FlexProfile.....	227
MC_GearIn.....	243	ML_FlexProfile, special features of Sercos- based axes.....	230
MC_GearIn, special features of Sercos- based axes.....	245	ML_Generator.....	481
MC_GearOut.....	247	GenerateCopyActualAcc.....	481
MC_MoveAbsolute.....	183, 374, 448	GenerateCopyActualAna.....	481
MC_MoveAdditive.....	186, 376	GenerateCopyActualPos.....	481
MC_MoveRelative.....	188, 379	GenerateCopyActualTrq.....	481
MC_MoveVelocity.....	192, 382	GenerateCopyActualVel.....	481
MC_Power.....	179, 367	GenerateCopyCyclicAcceleration.....	481
MC_ReadActualPosition.....	329, 415, 460	GenerateCopyCyclicAnalog.....	481
MC_ReadAxisError.....	330, 417, 461	GenerateCopyCyclicPosition.....	481
MC_ReadStatus.....	332, 418, 463	GenerateCopyCyclicTorque.....	481
MC_RemoveAxisFromGroup.....	252	GenerateCopyCyclicVelocity.....	481
MC_Reset.....	180, 368	ML_GetCpuLoad.....	151
MC_START_MODE.....	57	ML_GetDiagnosisLog.....	151
MC_STEP_MODE.....	32	ML_GetFirmwareString.....	152
MC_Stop.....	181, 369	ML_GetGateway.....	138
MC_SYNC_DIRECTION.....	59	ML_GetHardwareInfoCtrl.....	153
MC_TorqueControl.....	195	ML_GetInfoApplication.....	153
ML_AbortTrigger.....	334	ML_GetIpAddress.....	139
ML_AXISDATA_SM.....	110	ML_GetMacAddress.....	141
ML_Base_Constant_xxx.library.....	126	ML_GetMaxTemperature.....	154
ML_Base, data types at the interface of the IndraMotion function blocks.....	71	ML_GetNewestDiagnosisIndex.....	155
ML_Base.library.....	71	ML_GetOhcFan.....	155
ML_CONTROLDATA_SM.....	111	ML_GetScalingType.....	156
ML_ControllerBase.....		ML_GetSubnetMask.....	142
Basic class.....	473	ML_GetSystemSpecialPath.....	156
ML_ControllerBase.library.....	473	ML_GetSystemTimeUTC.....	147
ML_ControlOff.....	268	ML_GetTargetInfo.....	157
ML_ControlOn.....	267	ML_GetTemperature.....	157
ML_ControlWriteActualData.....	269	ML_GetWarningCpuLoad.....	158
ML_Convert.....	475	ML_InitTouchProbe.....	336
confFromAxis_Acceleration.....	475	ML_IPOImpactControl.....	484
convFromAxis_Analog.....	475	mGetAcceleration.....	487
convFromAxis_Force.....	475	mGetAxisControlledBy.....	489
convFromAxis_Position.....	475	mGetBreakBehaviour.....	487
		mGetDirection.....	486
		mGetNewState.....	485

mGetPosition.....	488	ML_SystemTftpGetRetransmitCount.....	163
mGetValidValues.....	488	ML_SystemTftpGetTimeout.....	163
mGetVelocity.....	489	ML_SystemTftpSetRetransmitCount.....	164
mSetAcceleration.....	486	ML_SystemTftpSetTimeout.....	164
mSetBreakBehaviour.....	487	ML_TaskExecuteFile.....	122
mSetDirection.....	485	ML_TaskExecuteGetActive.....	122
mSetNewState.....	485	ML_TaskExecuteGetName.....	123
mSetPosition.....	488	ML_TaskExecuteGetStatus.....	123
mSetValidValues.....	488	ML_TaskExecuteKill.....	124
mSetVelocity.....	489	ML_TASKHANDLE.....	124
Use methods.....	489	ML_TouchProbe.....	344
ML_IPOImpactControl methods.....	485	ML_TouchProbeContinuous.....	347
ML_IPOImpactControl.library.....	483	ML_TransmissionInfo.....	164
ML_LINK.....	113	ML_TransmissionStatus.....	165
ML_LOGBOOK.....	121	ML_WriteCyclicAnalog.....	256
ML_MessageHandler.....	482	ML_WriteCyclicAnalogCtrl.....	271
run_MSGH_ClearErrors.....	482	ML_WriteCyclicPosition.....	259
run_MSGH_ClearWarning.....	482	ML_WriteCyclicPositionCtrl.....	273
run_MSGH_SetMessagecode.....	482	ML_WriteCyclicTorque.....	263
ML_OORV.....	476	ML_WriteCyclicTorqueCtrl.....	275
ML_OORV_CONTROLLERDATA.....	474	ML_WriteCyclicVelocity.....	265
ML_OpenCyclicAnalogChannel.....	255	ML_WriteCyclicVelocityCtrl.....	277
ML_OpenCyclicAnalogChannelCtrl.....	270	ML_WriteExpectWindow.....	349
ML_OpenCyclicPositionChannel.....	257	MLC_TABLE.....	360
ML_OpenCyclicPositionChannelCtrl.....	272	MotionControl action chain.....	200
ML_OpenCyclicTorqueChannel.....	261	MP_Drive.....	420, 464
ML_OpenCyclicTorqueChannelCtrl.....	274	MP_DriveParameter.....	420, 464
ML_OpenCyclicVelocityChannel.....	264	MP_FB_IndraDrv_AcyclicCommunication.....	420, 464
ML_OpenCyclicVelocityChannelCtrl.....	276		
ML_ParamBase		Multi-axis modules	
Auxiliary functions.....	509	Profibus.....	387
Data access.....	508		
Data types.....	507	O	
General constants.....	509	OBJECTS.....	28
Language.....	507	Oscilloscope data via function variable access.....	108
Language selection.....	494		
PARAM_Parametersystem.....	494	P	
SERCOS_CMD_RETURN.....	511	PARAM_CmdExecutor.....	505
SERCOS_CMD_STATES.....	511	dwAddCmd.....	506
SERCOS_ERROR.....	509	dwClearStorage.....	506
ML_ParamBase.library.....	493	dwSetControlCmd.....	506
ML_PLCOpen.library.....	171	PARAM_ExtSercosParameter.....	503
ML_PLSDATA_SM.....	112	Access methods.....	504
ML_PROBEDATA_SM.....	113	Administration.....	505
ML_ProcessState.....	125	Attributes.....	503
ML_ReadDiagLog.....	116	dwGetOperationData.....	504
ML_SERCOS.....	114	dwLoadDefaultData.....	505
ML_SetAxisDiagEntry.....	117	dwSetOperationData.....	505
ML_SetDiagEntry.....	119	iGetParamType.....	505
ML_SetGateway.....	143	Load default values.....	505
ML_SetIpAddress.....	144	Parameter type.....	505
ML_SetSubnetMask.....	145	Read date.....	504
ML_SetSystemTimeUTC.....	149	Write date.....	505
ML_SetWarningCpuLoad.....	158	PARAM_Language.....	507
ML_SYNC.....	114	PARAM_Language_Array.....	507
ML_SystemGetTransmissionInfo.....	161	PARAM_Parametersystem.....	494
ML_SystemGetTransmissionStatus.....	162		
ML_SystemStartTransmission.....	162		

S

Safety instructions.....	2
Sercos III, C2C link	
MotionControl action chain.....	202
SERCOS_TABLE.....	357
Service hotline.....	513
Shared memory	
ControlData[].....	93
PlsData[].....	98
ProbeData[].....	103
Shared Memory	
AxisData[].....	85
Shutdown.....	12
Signal alert symbol.....	2
Signal words.....	2
SM_EXT_TYPES.....	115
SM_TYPES.....	114
Standard interfaces at function blocks.....	11
State diagram.....	365, 440
State diagram of a real axis.....	176
Status during function variable access.....	89, 96, 101, 107, 109
Status during synchronous function variable access in the C2C link.....	91
Support.....	513
Symbols.....	4
SYS_TIME_DATE.....	64
SYS_TIME64.....	63

T

TARGET_INFO.....	161
Time required for data transport with read and write parameter function blocks.....	280
Time required to process Motion function blocks in the IndraMotion MLC.....	353
Touch probe data via function variable ac- cess.....	105
Touch probe data via shared memory and function variable access.....	102
TOUCHPROBE_REF.....	31
TP_OBJECTS.....	31
Twin axis modules	
Profibus.....	387

U

User diagnostics.....	116

W

Warnings.....	2

Notes

Notes

Bosch Rexroth AG

Electric Drives and Controls

P.O. Box 13 57

97803 Lohr, Germany

Bgm.-Dr.-Nebel-Str. 2

97816 Lohr, Germany

Phone +49 9352 18 0

Fax +49 9352 18 8400

www.boschrexroth.com/electrics



R911341491