

Rexroth IndraLogic XLC IndraMotion MLC 13VRS

PLCopen Libraries

Library
R911336360

Edition 04



Title Rexroth IndraLogic XLC
IndraMotion MLC 13VRS
PLCopen Libraries

Type of Documentation Library

Document Typecode DOK-XLCMLC-FUNLIB**V13-LI04-EN-P

Internal File Reference RS-ed362e7d1cdd5b270a6846a5012127e2-3-en-US-3

Change Record

Edition	Release Date	Note
Edition 01	2012-03	First edition for MLC/XLC 13VRS
Edition 02	2013-02	Revision and supplement "open-loop"
Edition 03	2013-08	Revision and ML_Robot removed
Edition 04	2015-02	Revision of ML_PLCopen (ML_ControlWriteActualData, etc.), ML_ControllerBase (ML_Convert_DoublePrecision)

Copyright © Bosch Rexroth AG 2015

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

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

Editorial Department System Development Automation Motion Logic Control KWa, HBU (vha/MePe)

Table of Contents

	Page
1 About this documentation.....	13
1.1 Validity of the documentation.....	13
1.2 Using safety instructions.....	13
1.2.1 Safety instructions – Structure.....	13
1.2.2 Explaining signal words and safety alert symbol.....	13
1.2.3 Symbols used.....	15
1.3 Required and supplementing documentations.....	15
2 Standard Interfaces at Function Blocks.....	23
2.1 Motivation.....	23
2.2 Function Block Types.....	23
2.3 Function Block Types and their Interfaces.....	23
2.4 Inputs and Outputs of State-controlled Function Blocks, Terminating Processing.....	25
2.5 Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing.....	28
2.6 Inputs and Outputs of State-controlled Function Blocks, Permanent Processing.....	31
2.7 Inputs and Outputs of Edge-controlled Function Blocks, Permanent Processing.....	34
3 RIL_CommonTypes.library.....	37
3.1 RIL_CommonTypes, Overview.....	37
3.2 Global_Variables.....	38
3.2.1 IL_NULL.....	38
3.3 Data Types for Referencing Objects of IndraMotion Blocks	38
3.3.1 Referencing of Objects, General.....	38
3.3.2 CONTROLS.....	39
3.3.3 AXIS_REF.....	39
3.3.4 OBJECTS.....	40
3.3.5 AXIS_GROUP_REF.....	41
3.3.6 KIN_OBJECTS.....	41
3.3.7 TOUCHPROBE_REF.....	42
3.3.8 TP_OBJECTS.....	42
3.4 Data Types for the Support of Cams, MotionProfiles and FlexProfiles	43
3.4.1 Data Type for Switching between Cams, General.....	43
3.4.2 MC_CAM_ID.....	43
3.4.3 MC_STEP_MODE, MotionProfile Extended to FlexProfile.....	44
3.5 Data Types for Synchronization in the Position-controlled Synchronous Operation Modes	67
3.5.1 General.....	67
3.5.2 MC_START_MODE.....	68
3.5.3 MC_SYNC_DIRECTION.....	70
3.6 Data Types for Fault Diagnosis with IndraMotion Blocks	71
3.6.1 ERROR_CODE.....	71
3.6.2 ERROR_STRUCT.....	72
3.6.3 ERROR_TABLE.....	72
3.7 Data Types for the Indication of the Date and Absolute Time Designations for IndraMotion Blocks	73

Table of Contents

	Page
3.7.1	Indication of the Date and Absolute Time Designations, General..... 73
3.7.2	SYS_TIME64..... 73
3.7.3	EXT_SYS_TIME_DATE..... 74
3.7.4	SYS_TIME_DATE..... 74
3.8	Data Types for the Support of Touch Probe Functions 75
3.8.1	Data Types for the Support of Touch Probe Functions, General Information..... 75
3.8.2	MB_DIFF_EVAL..... 75
3.8.3	MB_EXPECT_WINDOW_MODE..... 76
3.8.4	MB_INTERPOLATION_MODE..... 77
3.8.5	MB_POSITION_EVAL..... 77
3.8.6	MB_PROBE_MODULE..... 77
3.8.7	MB_PROBE_INPUT..... 78
3.8.8	MB_PROBE_DATA..... 78
3.8.9	MB_PROBE_CONFIG..... 79
3.8.10	MB_PROBE_CONFIG, Special Features of SERCOS-based Axes..... 79
4	ML_Base.library..... 81
4.1	ML_Base, Overview..... 81
4.2	Data types 83
4.2.1	Data types/common types - general types..... 83
	MB_IDN..... 83
4.2.2	Data Types/ Flex Structs..... 84
	Data Types for the Support of Motion and FlexProfile Functions, General..... 84
	MB_CHANGESTEP_MODE..... 84
	MB_EVENT_ACTION..... 84
	MB_EVENT_TRIGGER..... 86
	MB_EXECUTION_MODE..... 86
	MB_FLEX_EVENT..... 87
	MB_MASTER_TYPE..... 87
	MB_PROFILE_START_POINT..... 87
	MB_PROFILE_STEP..... 89
	MB_STEP_TYPE..... 89
	MB_SYNC_TYPE..... 90
4.2.3	Data types/read write parameter - parameter access..... 90
	MB_SERCOS_DATA_STATUS..... 90
4.2.4	Data types/shared memory - direct access..... 91
	Data Types to Support the Shared Memory between Motion and PLC..... 91
	Axis Data via Shared Memory and Function Variable Access..... 91
	Axis Data via Shared Memory and Function Variable Access, General..... 91
	AxisData[] - Axis Data in Shared Memory..... 93
	DV_Axis[] - Axis Data via Function Variable Access..... 95
	GV_LinkAxis[] - Axis Data via Synchronous Function Variable Access in C2C Link 98
	Control Data via Shared Memory and Function Variable Access..... 100
	Control Data via Shared Memory and Function Variable Access, General..... 100
	ControlData[] - Control Data in Shared Memory..... 101
	DV_Control[] - Control Data via Function Variable Access 103

Table of Contents

	Page
Programmable Limit Switch Data via Shared Memory and Function Variable Access.....	105
Programmable Limit Switch Data via Shared Memory and Function Variable Access, General.....	105
PlsData[] - PLS Data in Shared Memory.....	106
DV_Pls[] - PLS Data via Function Variable Access.....	108
Touch Probe Data via Shared Memory and Function Variable Access.....	110
Touch Probe Data via Shared Memory and Function Variable Access, General.....	110
ProbeData[] - Touch Probe Data in Shared Memory	111
DV_Probe[] - Touch Probe Data via Function Variable Access.....	113
Oscilloscope Data via Function Variable Access.....	116
Oscilloscope Data via Function Variable Access, General.....	116
DV_Osc[] - Oscilloscope Data via Function Variable Access.....	116
ML_AXISDATA_SM.....	118
ML_CONTROLDATA_SM.....	120
ML_PLSDATA_SM.....	120
ML_PROBEDATA_SM.....	121
ML_LINK.....	121
ML_SERCOS.....	122
ML_SYNC.....	122
SM_TYPES.....	123
SM_EXT_TYPES.....	123
4.3 Error/diagnostic memory - supporting user diagnostics and logbook access.....	124
4.3.1 ML_ReadDiagLog.....	124
4.3.2 ML_SetAxisDiagEntry.....	125
4.3.3 ML_SetDiagEntry.....	127
4.3.4 E_LOG_CLASS.....	128
4.3.5 E_LOG_FACILITY.....	129
4.3.6 E_LOG_SEVERITY.....	129
4.3.7 ML_LOGBOOK.....	129
4.4 Global Variable List.....	130
4.4.1 GVL_Base.....	130
4.4.2 GVL_BaseConstant.....	131
4.4.3 Global Variable Lists/Global Variable Index.....	132
GVL_A_GlobalVarIdx.....	132
4.4.4 Parameter constants.....	133
Access to Data of an Axis, Control or Another Object.....	133
GVL_A_Param_ID.....	135
GVL_C_Param_ID.....	135
GVL_M_Param_ID.....	136
GVL_N_Param_ID.....	136
GVL_O_Param_ID.....	137
GVL_P_Param_ID.....	137
GVL_S_Param_ID.....	138
4.5 Interfaces, functions and function blocks.....	138
4.5.1 IL_SercosAttribute.....	138
4.5.2 IL_SercosElementsToParaNo.....	141

Table of Contents

	Page
4.5.3 IL_SercosStringToParaNo.....	142
4.6 IP settings - IP configuration.....	144
4.6.1 ML_GetGateway.....	144
4.6.2 ML_GetIpAddress.....	145
4.6.3 ML_GetMacAddress.....	146
4.6.4 ML_GetSubnetMask.....	147
4.6.5 ML_SetGateway.....	148
4.6.6 ML_SetIpAddress.....	150
4.6.7 ML_SetSubnetMask.....	151
4.7 SystemTime - system time.....	152
4.7.1 IL_SetDateTime.....	152
4.7.2 ML_GetSystemTimeUTC.....	153
4.7.3 ML_SetSystemTimeUTC.....	154
4.7.4 ML_DAYSOFWEEK.....	156
4.8 Target - Target system.....	156
4.8.1 ML_GetCpuLoad.....	156
4.8.2 ML_GetDiagnosisLog.....	157
4.8.3 ML_GetFirmwareString.....	157
4.8.4 ML_GetHardwareInfoCtrl.....	158
4.8.5 ML_GetInfoApplication.....	159
4.8.6 ML_GetMaxTemperature.....	160
4.8.7 ML_GetNewestDiagnosisIndex.....	160
4.8.8 ML_GetOhcFan.....	160
4.8.9 ML_GetScalingType.....	161
4.8.10 ML_GetTargetInfo.....	161
4.8.11 ML_GetTemperature.....	161
4.8.12 ML_GetWarningCpuLoad.....	162
4.8.13 ML_SetWarningCpuLoad.....	162
4.8.14 APP_INFO.....	163
4.8.15 E_SCALING_TYPE.....	163
4.8.16 INFO_APPLICATIONS.....	163
4.8.17 ML_DIAGNOSIS_LOG.....	163
4.8.18 TARGET_INFO.....	164
5 Function Blocks According to the PLCopen Specification.....	165
6 ML_PLCopen.library.....	169
6.1 ML_PLCopen, General.....	169
6.2 Global_Variables.....	172
6.2.1 ProbeSelect.....	172
6.3 Data types/data types of TouchProbe.....	172
6.3.1 PROBECONFIG.....	172
6.4 State diagram of a real axis.....	172
6.5 Function blocks for single-axis control.....	176
6.5.1 MC_Power.....	176

Table of Contents

	Page
6.5.2 MC_Power, special features of Sercos-based axes.....	177
6.5.3 MC_Reset.....	177
6.5.4 MC_Reset, Special Features of SERCOS-Based Axes.....	178
6.5.5 MC_Stop.....	178
6.5.6 MC_Stop, Special Features of SERCOS-Based Axes.....	180
6.5.7 MC_MoveAbsolute.....	180
6.5.8 MC_MoveAbsolute, Special Features of SERCOS-Based Axes.....	182
6.5.9 MC_MoveAdditive.....	183
6.5.10 MC_MoveAdditive, Special Features of SERCOS-Based Axes.....	185
6.5.11 MC_MoveRelative.....	185
6.5.12 MC_MoveRelative, Special Features of SERCOS-Based Axes.....	188
6.5.13 MC_MoveVelocity.....	189
6.5.14 MC_MoveVelocity, Special Features of SERCOS-Based Axes.....	192
6.5.15 MC_TorqueControl.....	192
6.5.16 MC_TorqueControl, special features of Sercos-based axes.....	194
6.5.17 MB_SetPositionControlMode.....	195
6.5.18 MB_Home.....	196
6.5.19 MB_Home, special features of Sercos-based axes.....	197
6.6 Function blocks for multi-axis control.....	198
6.6.1 The MotionControl Action Chain.....	198
6.6.2 MB_Phasing.....	201
6.6.3 MB_PhasingSlave.....	203
6.6.4 MC_CamIn.....	205
6.6.5 MC_CamIn, special features of Sercos-based axes.....	207
6.6.6 MC_CamOut.....	209
6.6.7 MC_CamOut, Special Features of SERCOS-based Axes.....	209
6.6.8 MB_MotionProfile.....	210
6.6.9 MC_MotionProfile, special features of Sercos-based axes.....	212
6.6.10 PLCopen function blocks to support MotionProfiles.....	214
General information.....	214
MB_ChangeProfileSet	216
MB_ChangeProfileSet, Special Features of SERCOS-based Axes.....	218
MB_ChangeProfileStep.....	219
MB_ChangeProfileStep, Special Features of SERCOS-based Axes.....	222
MB_ChangeCamData.....	223
MB_ChangeCamData, Special Features of SERCOS-based Axes.....	224
6.6.11 ML_FlexProfile.....	224
6.6.12 ML_FlexProfile, special features of Sercos-based axes.....	227
6.6.13 Switching and Synchronizing the FlexProfile	230
Selecting the Coordinate System.....	230
Switching and Synchronizing	230
Processing Types.....	233
6.6.14 PLCopen Function Blocks for Supporting FlexProfiles.....	234
General information.....	234
MB_ChangeFlexProfileSet	236
Characteristics of the Event Control.....	237

Table of Contents

	Page
MB_ChangeFlexEventSet.....	239
6.6.15 MC_GearIn.....	240
6.6.16 MC_GearIn, special features of Sercos-based axes.....	242
6.6.17 MC_GearOut.....	243
6.6.18 MC_GearOut, Special Features of SERCOS-based Axes.....	244
6.6.19 MB_GearInPos.....	245
6.6.20 MB_GearInPos, special features of Sercos-based axes.....	246
6.7 Function Blocks for the Axis Group Control.....	248
6.7.1 MC_AddAxisToGroup.....	248
6.7.2 MC_RemoveAxisFromGroup.....	249
6.8 Function Blocks for the Use of the Cyclic Data Channel.....	250
6.8.1 General Information on Using the Cyclic Data Channel.....	250
6.8.2 ML_OpenCyclicAnalogChannel.....	252
6.8.3 ML_WriteCyclicAnalog.....	253
6.8.4 ML_OpenCyclicPositionChannel.....	254
6.8.5 ML_WriteCyclicPosition.....	255
6.8.6 ML_OpenCyclicTorqueChannel.....	258
6.8.7 ML_WriteCyclicTorque.....	259
6.8.8 ML_OpenCyclicVelocityChannel.....	260
6.8.9 ML_WriteCyclicVelocity.....	261
6.9 Function blocks for controlled motion.....	262
6.9.1 General Information on Controlled Motion.....	262
6.9.2 ML_ControlOn.....	264
6.9.3 ML_ControlOff.....	264
6.9.4 ML_ControlWriteActualData.....	265
6.9.5 ML_OpenCyclicAnalogChannelCtrl.....	266
6.9.6 ML_WriteCyclicAnalogCtrl.....	267
6.9.7 ML_OpenCyclicPositionChannelCtrl.....	268
6.9.8 ML_WriteCyclicPositionCtrl.....	269
6.9.9 ML_OpenCyclicTorqueChannelCtrl.....	270
6.9.10 ML_WriteCyclicTorqueCtrl.....	271
6.9.11 ML_OpenCyclicVelocityChannelCtrl.....	272
6.9.12 ML_WriteCyclicVelocityCtrl.....	274
6.10 Parameter Access.....	275
6.10.1 Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object.....	275
6.10.2 Time Required for Data Transport with Read and Write Parameter Function Blocks.....	277
6.10.3 MB_Command.....	277
6.10.4 MB_Command, Special Features of SERCOS-based Axes.....	280
6.10.5 MB_ReadLintParameter.....	281
6.10.6 MB_ReadLintParameter, Special Features of SERCOS-based Axes.....	282
6.10.7 MB_ReadListParameter.....	283
6.10.8 MB_ReadListParameter, Special Features of SERCOS-based Axes.....	285
6.10.9 MB_ReadLrealListParameter.....	285
6.10.10 MB_ReadLrealListParameter, Special Features of SERCOS-based Axes.....	288
6.10.11 MB_ReadLrealParameter.....	288
6.10.12 MB_ReadLrealParameter, Special Features of SERCOS-based Axes.....	289

Table of Contents

	Page
6.10.13 MB_ReadMaxRealValue.....	290
6.10.14 MB_ReadMaxRealValue, Special Features of SERCOS-based Axes.....	291
6.10.15 MB_ReadMaxValue.....	291
6.10.16 MB_ReadMaxValue, Special Features of SERCOS-based Axes.....	292
6.10.17 MB_ReadMinRealValue.....	293
6.10.18 MB_ReadMinRealValue, Special Features of SERCOS-based Axes.....	294
6.10.19 MB_ReadMinValue.....	294
6.10.20 MB_ReadMinValue, Special Features of SERCOS-based Axes.....	295
6.10.21 MB_ReadName.....	296
6.10.22 MB_ReadName, Special Features of SERCOS-based Axes.....	297
6.10.23 MB_ReadParameter.....	297
6.10.24 MB_ReadParameter, Special Features of SERCOS-based Axes.....	298
6.10.25 MB_ReadRealParameter.....	299
6.10.26 MB_ReadRealParameter, Special Features of SERCOS-based Axes.....	300
6.10.27 MB_ReadSercosAttribute.....	300
6.10.28 MB_ReadSercosAttribute, Special Features of SERCOS-based Axes.....	301
6.10.29 MB_ReadSercosDataStatus.....	302
6.10.30 MB_ReadSercosDataStatus, Special Features of SERCOS-based Axes.....	303
6.10.31 MB_ReadStringParameter.....	304
6.10.32 MB_ReadStringParameter, Special Features of SERCOS-based Axes.....	305
6.10.33 MB_ReadUnit.....	305
6.10.34 MB_ReadUnit, Special Features of SERCOS-based Axes.....	306
6.10.35 MB_WriteLintParameter.....	306
6.10.36 MB_WriteLintParameter, Special Features of SERCOS-based Axes.....	307
6.10.37 MB_WriteListParameter.....	308
6.10.38 MB_WriteListParameter, Special Features of SERCOS-based Axes.....	310
6.10.39 MB_WriteLrealParameter.....	310
6.10.40 MB_WriteLrealParameter, Special Features of SERCOS-based Axes.....	311
6.10.41 MB_WriteParameter.....	312
6.10.42 MB_WriteParameter, Special Features of SERCOS-based Axes.....	313
6.10.43 MB_WriteRealParameter.....	313
6.10.44 MB_WriteRealParameter, Special Features of SERCOS-based Axes.....	314
6.10.45 MB_WriteStringParameter.....	315
6.10.46 MB_WriteStringParameter, Special Features of SERCOS-based Axes.....	316
6.10.47 MC_ReadActualPosition.....	316
6.10.48 MC_ReadActualPosition, Special Features of SERCOS-Based Axes.....	317
6.10.49 MC_ReadAxisError.....	317
6.10.50 MC_ReadAxisError, Special Features of SERCOS-Based Axes.....	318
6.10.51 MC_ReadStatus.....	318
6.10.52 MC_ReadStatus, Special Features of sercos-Based Axes.....	320
6.11 Function blocks to use the touch probe function.....	320
6.11.1 TouchProbe and Data Function Blocks for Using the Touch Probe Function.....	320
6.11.2 ML_AbortTrigger.....	321
6.11.3 ML_InitTouchProbe.....	322
6.11.4 ML_TouchProbe.....	329
6.11.5 ML_TouchProbeContinuous.....	331

Table of Contents

	Page
6.11.6 ML_WriteExpectWindow.....	334
6.12 Time Required to Process Motion Function Blocks in the IndraMotion MLC.....	337
6.13 Axis types and function blocks supported by these axis types.....	338
6.14 Error Tables (Error_Tables).....	340
6.14.1 SERCOS_TABLE, ERROR_TABLE 16#0010	340
6.14.2 MLC_TABLE, ERROR_TABLE 16#0030	343
7 RMB_PLCOpenFieldBus.library.....	353
7.1 RMB_PLCOpenFieldBus.library, General.....	353
7.2 Parameter address constants.....	353
7.2.1 Access to data of a (field bus) drive.....	353
7.2.2 GVL_S_Param_ID.....	353
7.2.3 GVL_P_Param_ID.....	354
7.3 Data types.....	354
7.3.1 Data types, general.....	354
7.3.2 MB_IDN.....	355
7.3.3 AXIS_REF.....	355
7.4 Function Blocks for Single-Axis Control (Motion_FBs).....	357
7.4.1 Motion_FBs, General.....	357
7.4.2 MC_Power.....	359
7.4.3 MC_Power, Special Features of Field Bus Drives.....	360
7.4.4 MC_Reset.....	360
7.4.5 MC_Reset, Special Features of Field Bus Drives.....	361
7.4.6 MC_Stop.....	362
7.4.7 MC_Stop, Special Features of Field Bus Drives.....	363
7.4.8 MB_Stop.....	363
7.4.9 MB_Stop, Special Features of Field Bus Drives.....	365
7.4.10 MB_Home.....	365
7.4.11 MB_Home, Special Features of Field Bus Drives.....	366
7.4.12 MC_MoveAbsolute.....	366
7.4.13 MC_MoveAbsolute, Special Features of Field Bus Drives.....	368
7.4.14 MC_MoveAdditive.....	368
7.4.15 MC_MoveAdditive, Special Features of Field Bus Drives.....	370
7.4.16 MC_MoveRelative.....	371
7.4.17 MC_MoveRelative, Special Features of Field Bus Drives.....	373
7.4.18 MC_MoveVelocity.....	374
7.4.19 MC_MoveVelocity, Special Features of Field Bus Drives.....	377
7.5 Function Blocks for Parameter Access (Parameter_FBs).....	377
7.5.1 Parameter_FBs, General Information.....	377
7.5.2 Access to Data of a Drive (IndraDrive).....	379
7.5.3 Special Features of Twin or Multi-Axis Modules.....	380
7.5.4 MB_ReadListParameter.....	381
7.5.5 MB_ReadListParameter, Special Features of the Field Bus Drives.....	383
7.5.6 MB_ReadListSegParameterPtr.....	384
7.5.7 MB_ReadListSegParameterPtr, Special Features of the Field Bus Drives.....	386
7.5.8 MB_ReadMaxRealValue.....	387

Table of Contents

	Page
7.5.9 MB_ReadMaxRealValue, Special Features of the Field Bus Drives.....	388
7.5.10 MB_ReadMaxValue.....	388
7.5.11 MB_ReadMaxValue, Special Features of the Field Bus Drives.....	389
7.5.12 MB_ReadMinRealValue.....	390
7.5.13 MB_ReadMinRealValue, Special Features of the Field Bus Drives.....	391
7.5.14 MB_ReadMinValue.....	391
7.5.15 MB_ReadMinValue, Special Features of the Field Bus Drives.....	392
7.5.16 MB_ReadParameter.....	393
7.5.17 MB_ReadParameter, Special Features of the Field Bus Drives.....	394
7.5.18 MB_ReadRealParameter.....	394
7.5.19 MB_ReadRealParameter, Special Features of the Field Bus Drives.....	395
7.5.20 MB_ReadSercosAttribute.....	396
7.5.21 MB_ReadSercosAttribute, Special Features of the Field Bus Drives.....	397
7.5.22 MB_ReadStringParameter.....	397
7.5.23 MB_ReadStringParameter, Special Features of the Field Bus Drives.....	398
7.5.24 MB_WriteListParameter.....	399
7.5.25 MB_WriteListParameter, Special Features of the Field Bus Drives.....	401
7.5.26 MB_WriteListSegParameterPtr.....	401
7.5.27 MB_WriteListSegParameterPtr, Special Features of the Field Bus Drives.....	404
7.5.28 MB_WriteParameter.....	404
7.5.29 MB_WriteParameter, Special Features of the Field Bus Drives.....	405
7.5.30 MB_WriteRealParameter.....	405
7.5.31 MB_WriteRealParameter, Special Features of the Field Bus Drives.....	406
7.5.32 MB_WriteStringParameter.....	407
7.5.33 MB_WriteStringParameter, Special Features of the Field Bus Drives.....	408
7.5.34 MC_ReadActualPosition.....	408
7.5.35 MC_ReadActualPosition, Special Features of the Field Bus Drives.....	409
7.5.36 MC_ReadAxisError.....	410
7.5.37 MC_ReadAxisError, Special Features of the Field Bus Drives.....	411
7.5.38 MC_ReadStatus.....	411
7.5.39 MC_ReadStatus, Special Features of the Field Bus Drives.....	413
7.6 System_FBs	413
7.7 PLC Controls Drives via a Field Bus Connection (Field Bus Axes).....	413
7.7.1 General Information.....	413
7.7.2 Related Documentation for IndraDrive.....	414
7.7.3 Creating Field Bus Axes in the Project.....	415
7.7.4 Drive Settings for the PLC Communication.....	416
General Information.....	416
Establishing the Communication for Drive Parameterization.....	417
Communication Settings for IndraDrive.....	418
7.7.5 Including Required Libraries.....	419
7.7.6 Addressing Field Bus Drives in the PLC Project.....	421
AXIS_REF, Axis Addresses of the Field Bus Drives.....	421
AxisData[], Axis Data.....	423
7.7.7 I/O Addressing of the Drives in the PLC Project.....	423

Table of Contents

	Page
8 ML_ControllerBase.library	425
8.1 Introduction.....	425
8.2 Basic class of the controller.....	425
8.2.1 Basic class.....	425
8.2.2 Controller Data.....	426
8.2.3 Data conversion of a controller	427
8.2.4 Interface IAcutalValue.....	428
8.2.5 Interface of the IModuleControl.....	428
8.2.6 Interface of the IOperationModeHandler.....	428
8.2.7 Interface of the IOutputData.....	428
8.3 Controller management.....	428
8.3.1 Connecting the controller to the controller management.....	428
8.3.2 Cyclic procedure of the controller management.....	429
8.3.3 State machine: Operating state.....	429
8.3.4 Cyclic procedure of the controller.....	430
8.3.5 Operation mode handling.....	430
8.3.6 Predefined generators of the operation modes.....	432
8.3.7 Diagnostic handling.....	433
9 ML_IPOImpactControl.library	435
9.1 Introduction and Overview.....	435
9.2 ML_IPOImpactControl.....	436
9.3 ML_IPOImpactControl Methods.....	437
9.3.1 mSetNewState.....	437
9.3.2 mGetNewState.....	437
9.3.3 mSetDirection.....	437
9.3.4 mGetDirection.....	438
9.3.5 mSetAcceleration.....	438
9.3.6 mGetAcceleration.....	438
9.3.7 mSetBreakBehaviour.....	439
9.3.8 mGetBreakBehaviour.....	439
9.3.9 mSetPosition.....	439
9.3.10 mGetPosition.....	440
9.3.11 mSetValidValues.....	440
9.3.12 mGetValidValues.....	440
9.3.13 mSetVelocity.....	441
9.3.14 mGetVelocity.....	441
9.3.15 mGetAxisControlledBy.....	441
9.4 Applying the Methods.....	441
9.4.1 Introduction.....	441
9.4.2 Example 1: Activate "Break" State	441
9.4.3 Example 2: Activate "Jump" State.....	443
10 ML_ParamBase.library	445
10.1 Usage.....	445

Table of Contents

	Page
10.2	Function blocks..... 446
10.2.1	PARAM_Parametersystem..... 446
	Initialization..... 446
	Registering Parameters..... 446
	Functionalities..... 446
	General information..... 446
	Language Switching..... 446
	Reading Parameters..... 447
	Reading Parameters..... 447
	Saving Parameters..... 447
	Restoring Parameters..... 448
	Loading Default Values..... 448
	Creating Parameter Lists..... 449
	Synchronize Parameter Values..... 450
	Cyclic Write Protection..... 450
10.2.2	PARAM_SercosParameter 450
	General information..... 450
	Interfaces..... 450
	Usage..... 450
	Read Interface..... 450
	Write Interface..... 451
	Command Interface..... 451
	Attributes..... 452
	Access Methods..... 452
	Read Data Status..... 452
	Read IDN..... 453
	Read Name..... 453
	Read Attribute..... 453
	Read Minimum Value 453
	Read Maximum Value..... 453
	Read Date..... 454
	Write Date..... 454
	Administration..... 454
	Loading Default Values..... 454
	Parameter Type..... 454
	Cyclic Write Protection..... 455
10.2.3	PARAM_ExtSercosParameter..... 455
	General information..... 455
	Attributes..... 455
	Access Methods..... 456
	Reading Date..... 456
	Writing Date..... 456
	Administration..... 457
	Loading Default Values..... 457
	Parameter Type..... 457
10.2.4	PARAM_CmdExecutor..... 457

Table of Contents

	Page
Brief description.....	457
Methods.....	457
10.3 Command Handling.....	458
10.4 Data Types.....	459
10.4.1 Language.....	459
10.4.2 Data Access.....	460
10.5 Auxiliary Functions.....	461
10.6 Constants.....	461
 11 Service and support.....	 465
 Index.....	 467

1 About this documentation

1.1 Validity of the documentation

Target group This documentation is intended for users who wish to know more about the libraries of IndraLogic XLC and IndraMotion MLC.

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

This documentation describes the function blocks, functions, interfaces and data types of the libraries

- [RIL_CommonTypes](#), page 37, data types for general use in all IndraWorks systems,
- [ML_Base](#), page 81, data types of the IndraLogic XLC and IndraMotion MLC systems for Sercos-based axes.
- *PLCopen function blocks, functions and data types for*
 - Sercos-based axes, [ML_PLCopen](#), page 169
 - Profibus-based drives, [RMB_PLCopenFieldbus](#), page 353,
- The library [ML_ControllerBase](#), page 425, provides the option to implement a controller into the PLC based on premanufactured modules.
- The library [ML_IPOLImpactControl](#), page 435, contains a function block for direct interpolation influencing in an XLC/MLC.
- The library [ML_ParamBase.library](#), page 445, allows to equip the controller axis with an extended parameterability.

The documentation also contains

- [Information on the standard interfaces of the function blocks](#), page 23, and
- [the error reaction of the function blocks](#), page 340.

1.2 Using safety instructions

1.2.1 Safety instructions – Structure

The safety instructions are structured as follows:

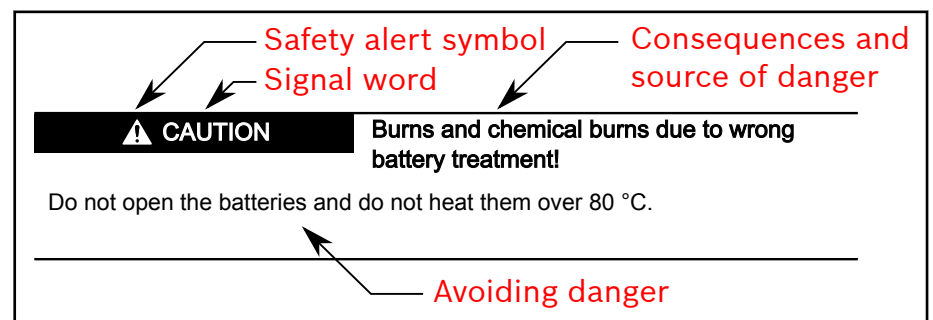


Fig. 1-1: Safety instructions – structure

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).

About this documentation

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

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

DANGER

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

WARNING

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

CAUTION

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

NOTICE

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

1.2.3 Symbols used

Pointers are displayed as follows:



This is a note.

Tips are displayed as follows:



This is a tip.

1.3 Required and supplementing documentations

Documentation titles with type codes and parts numbers

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

About this documentation

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

Tab. 1-1: XCL/MLC documentation overview

Motion		MLC	XLC
/23/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS PLCopen Libraries DOK-XLCMLC-FUNLIB**V13-LIRS-EN-P, R911336360 This documentation describes the function blocks, functions and data types of the RIL_Common-Types, ML_Base and ML_PLCopen libraries for the IndraLogic XLC/IndraMotion MLC. It also includes the error reactions of function blocks.	X	X
/27/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Generic Application Template DOK-XLCMLC-TF*GAT**V13-APRS-EN-P, R911336370 This documentation provides a structured template to the IndraLogic PLC programmer. This template can be used to add and edit the PLC programming code. It includes the template, the template wizard and example applications.	X	X
/31/	Rexroth IndraMotion MLC 13VRS RCL Programming Instruction DOK-MLC***-RCL*PRO*V13-APRS-EN-P, R911336297 This documentation provides information on the RobotControl. The programming language RCL (RobotControl Language) is focused. The program structure, variables, functions, motion statements and the required system parameters are described.	X	
/32/	Rexroth IndraMotion MLC 13VRS Robot Control V2 DOK-MLC***-ROCO***V13-RERS-EN-P, R911341035 The documentation provides information about the Robot Control V2. The focus of this documentation is on PLCopen programming. The program structure, variables, functions and motion commands as well as the required system parameters are described. Furthermore, the documentation contains the description of the ML_Robot library.	X	
/21/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Parameters DOK-XLCMLC-PARAM***V13-RERS-EN-P, R911336364 This documentation describes the parameters of the XLC/MLC systems as well as the interaction between parameterization and programming. It includes the axis parameters, control parameters, kinematic parameters, touch probe parameters and programmable limit switch parameters.	X	X
/10/	Rexroth IndraDrive Firmware for Drive Controllers MPH-, MPB-, MPD-, MPC-07 DOK-INDRV*-MP*-07VRS**-FKRS-EN-P, R911328670 This documentation describes all functional properties of the IndraDrive firmware in the variants MPH-07, MPB-07, MPD-07 and MPC-07.		
/11/	Rexroth IndraDrive MPx-17 Functions DOK-INDRV*-MP*-17VRS**-APRS-EN-P, R911331236 This documentation describes all functional properties of the IndraDrive firmware in the variants MPB-17, MPM-17, MPC-17 and MPE-17.		

Tab. 1-2: XCL/MLC documentation overview

About this documentation

Field Buses		MLC	XLC
/4/	Rexroth IndraWorks 13VRS Field Buses DOK-IWORKS-FB*****V13-APRS-EN-P, R911336872 This documentation describes the supported field buses and their diagnostic function blocks.	X	X
/4a/	Rexroth IndraWorks 13VRS Field Buses Libraries DOK-IWORKS-FB*LIB**V13-LIRS-EN-P, R911337857 This manual describes the field bus libraries: RIL_ProfibusbDP_02, RIL_ProfibusbDPSlave, RIL_Profine-tIO, RIL_ProfinetIODevice, RIL_EtherNetIPAdapter, RIL_MappingList, RIL_SERCOSIII, RIL_Inline including their diagnostics and error reactions of the function blocks.	X	X
/63/	SERCOS System Manual for I/O Devices DOK-CONTRL-ILS3*****-APRS-EN-P, R911333512 This documentation describes the configuration, parameterization, commissioning and diagnostics of I/O devices with a SERCOS interface.	X	X

Tab. 1-3: XCL/MLC documentation overview

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

Tab. 1-4: XCL/MLC documentation overview

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

Tab. 1-5: XCL/MLC documentation overview

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

Tab. 1-6: XCL/MLC documentation overview

Hydraulics		MLC	XLC
/40/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Sequential Programming DOK-XLCMLC-SEQPROG*V13-LIRS-EN-P, R911336368 This documentation includes the library for the sequential programming.	X	X
/41/	Rexroth IndraMotion MLC 13VRS MH_TechHydrBase DOK-MLC***-TF*HBASEV13-LIRS-EN-P, R911336312 No purpose defined.	X	
/42/	Rexroth IndraMotion MLC 13VRS MH_TechHydrMotion DOK-MLC***-TF*HMOT*V13-LIRS-EN-P, R911336316 No purpose defined.	X	
/43/	Rexroth IndraMotion MLC 13VRS MH_HydrControl DOK-MLC***-TFHCLIB*V13-LIRS-EN-P, R911336328 No purpose defined.	X	
/44/	Rexroth IndraMotion MLC 13VRS MH_SynchControl DOK-MLC***-TFHSLIB*V13-LIRS-EN-P, R911336330 No purpose defined.	X	

Tab. 1-7: XCL/MLC documentation overview

About this documentation

Hardware		MLC	XLC
/1/	Rexroth IndraControl L45/L65/L85 Control DOK-CONTRL-ICL45L65L85-PRRS-EN-P, R911332116 This documentation describes the IndraControl L45/L65/L85 controls.	X	X
/2/	Rexroth IndraControl L25 DOK-CONTRL-IC*L25*****-PRRS-EN-P, R911328474 This documentation describes the IndraControl L25 controls.	X	X
/24/	Rexroth IndraControl Lxx 13VRS Function Modules DOK-CONTRL-FM*LXX**V13-APRS-EN-P, R911336283 This documentation describes all function modules of the Lxx controls including engineering and diagnostics.	X	X
/12/	Rexroth IndraDrive Drive Controllers MPx-02 to MPx-07 DOK-INDRV*-GEN-**VRS**-PARS-EN-P, R911297317 This documentation describes all parameters implemented in the firmware for drive controllers of the IndraDrive family. It supports the parameterization of the drive controllers.		
/13/	Rexroth IndraDrive MPx-02 to MPx-07 and HMV DOK-INDRV*-GEN-**VRS**-WARS-EN-P, R911297319 This documentation describes all diagnostics implemented in the following firmwares: - Drive controller firmwares from MPx-02 to MPx-08 and - firmwares of the supply devices of the type "HMF". It supports the operating crew as well as the programmer at troubleshooting.		
/35/	Rexroth IndraDrive Drive Controller Control Sections CSB01, CSH01, CDB01 DOK-INDRV*-CSH*****-PRRS-EN-P, R911295012 This documentation is a project planning manual to select and use the control sections CSB01, CDB01 and CSH01 for drive controllers of the product families Rexroth IndraDrive M and Rexroth IndraDrive C.		

Tab. 1-8: XCL/MLC documentation overview

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

Tab. 1-9: XCL/MLC documentation overview

System Overview		MLC	XLC
/48/	Rexroth IndraMotion for Printing 13VRS System Overview DOK-IM*PR*-SYSTEM**V13-PRRS-EN-P, R911336289 This documentation describes the product IndraMotion for Printing. It introduces the control systems, drive systems and I/O systems as well as the commissioning and programming.	X	
/48/	Rexroth IndraMotion for Packaging 13VRS System Overview DOK-IM*PA*-SYSTEM**V13-PRRS-EN-P, R911336287 This documentation describes the product IndraMotion for Packaging. It introduces the control systems, drive systems and I/O systems as well as the commissioning and programming.	X	
/9/	Rexroth IndraMotion MLC 13VRS System Overview DOK-MLC***-SYSTEM**V13-PRRS-EN-P, R911336310 This documentation provides an overview on the hardware/software components of the automation system IndraMotion MLC in the mentioned version. It helps assembling a system.	X	
/9/	Rexroth IndraLogic XLC 13VRS System Overview DOK-XLC***-SYSTEM**V13-PRRS-EN-P, R911336354 This documentation provides an overview on hardware/software components of the automation system IndraLogic XLC in the respective version. It helps assembling a system.		X

Tab. 1-10: XCL/MLC documentation overview

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

Tab. 1-11: XCL/MLC documentation overview

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

With function blocks, it is possible to encapsulate complex tasks so they can be reused 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 completely and tasks which require continuous intervention once they are started.

State-controlled	If a function block always repeats its task after switch-on whenever it has reached a defined state, this FB processes in a "state-controlled" way.
Edge-controlled	If a function block fulfills its task exactly once after being switched on, this FB processes in an "edge-controlled" way.
Terminating processing	If a block can definitively process its order and afterwards stop working, this is called "definitive processing".
Permanent processing	If a block can never conclude its order definitively but is continuously in engagement, this is defined as "continuous processing".

In combination, there are four FB types possible:

Control	Processing	Example
State-controlled (Enable)	Terminating (Done)	Permanent reading of a parameter. (if the reading is completed, the function block has completed its task. It 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 deactivates the control.)
Edge-controlled (Execute)	Terminating (Done)	One-time writing of a parameter. (If the writing is completed, the FB has completed its task. A new edge is required at the control input (Execute) in order to start new writing.) 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 deactivated via the interfaces of this FB anymore.) e.g. MC_MoveVelocity

Tab. 2-1: Overview on FB types

2.3 Function 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

Standard Interfaces at Function Blocks

- Execute = edge-controlled
- Outputs** In order to mark whether a function block processes in a terminating way or permanent, different variable names are used for the outputs which indicate the processing state.



- Done = Terminating processing
- In..... = continuous processing (e.g., InOperation, InSync, InVelocity, InGear, InTorque...)

The optional "Active" output can be used with terminating processing and permanent processing FBs behaving differently!

All outputs for the error identification (Error, ErrorID, ErrorIdent) can be found with terminating processing and permanent processing FBs behaving identically.

Standardized identifiers

Identifier	Description
Inputs	
Enable	Enable input of state-controlled FBs
Execute	Enable input of edge-controlled FBs. Other FBs can interrupt this.
ExecuteLock	Enable input of edge-controlled FBs. As long as this input is TRUE, the FB cannot be interrupted by others.
Outputs	
Active	Output marking the processing time
Done	Processing completed successfully and the data outputs are valid.
In.....	Output signalizes that the FB is processing its task and that the data outputs are valid.
Shutdown (optional)	The output signalizes that the FB is currently reaching a defined final state (e.g., it stops axes, releases resources, ...). The FB must be called as long as necessary for "Shutdown" to become FALSE.
CommandAborted	Output indicates that the FB was interrupted (e.g. by another FB).
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, Terminating Processing

Control/ Processing	I/O	Variable name	Description
State-controlled/terminating	E	Enable	The input variables are registered with a positive edge at "Enable". New input values do not become effective until there is the next positive edge at "Enable". "Enable" has to remain TRUE while the function block is processed! If "Enable" is disabled, processing is interrupted and the "Done", "Active", "CommandAborted", and "Error" outputs are set to FALSE. If required, "Shutdown" signalizes that further signals are necessary for the FB to reach its final state defined.
	A	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 "Enable" is still TRUE, the FB starts processing again.
	A	Active	If "Active" is TRUE, the function block is processing its actual task and is then in an intermediary state. Possible preprocessing are not marked with this output! "Done", "Error" and "CommandAborted" are FALSE!
	A	Shutdown (optional)	If a "Shutdown" output is available, the FB needs several cycles of calculation time when "Enable" is deactivated in order to reach a defined final state and to release the resources used. The FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	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 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 FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	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 FB must be called as long as necessary for "Shutdown" to become FALSE.

Tab. 2-3: I/O of state-controlled function blocks, terminating processing



If specific inputs must be applied not only with an edge at "Enable" but cyclically as long as the function block is processed, this must be explicitly documented; it is not allowed to specify this by means of variable names.



If a function block is provided with "Shutdown", the behavior of this FB when "Enable" is set to TRUE while "Shutdown" remains set (retrigger), must be precisely defined.

Standard Interfaces at Function Blocks

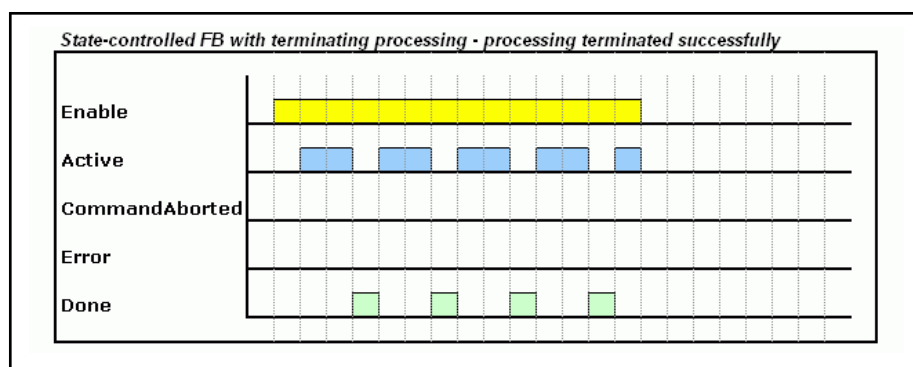


Fig. 2-1: I/O of state-controlled function blocks, definitive processing without "Shutdown". Processing successfully completed

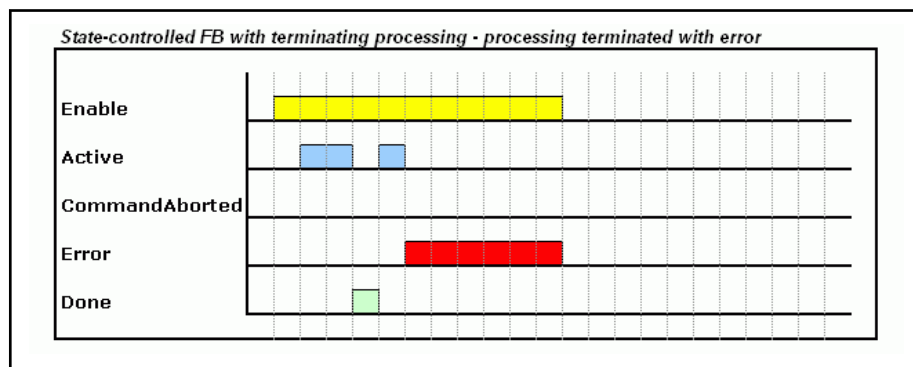


Fig. 2-2: I/O of state-controlled function blocks, definitive processing without "Shutdown". Processing completed with error

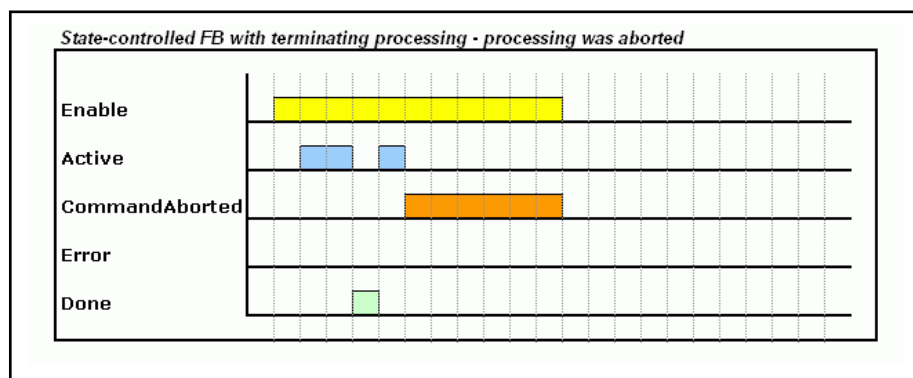


Fig. 2-3: I/O of state-controlled function blocks, definitive processing without "Shutdown". Processing interrupted

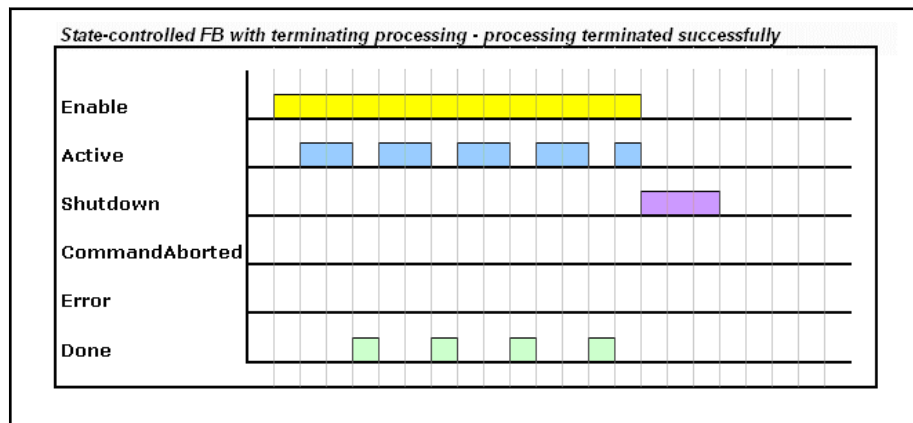


Fig. 2-4: I/O of state-controlled function blocks, definitive processing with "Shutdown". Processing successfully completed.

Standard Interfaces at Function Blocks

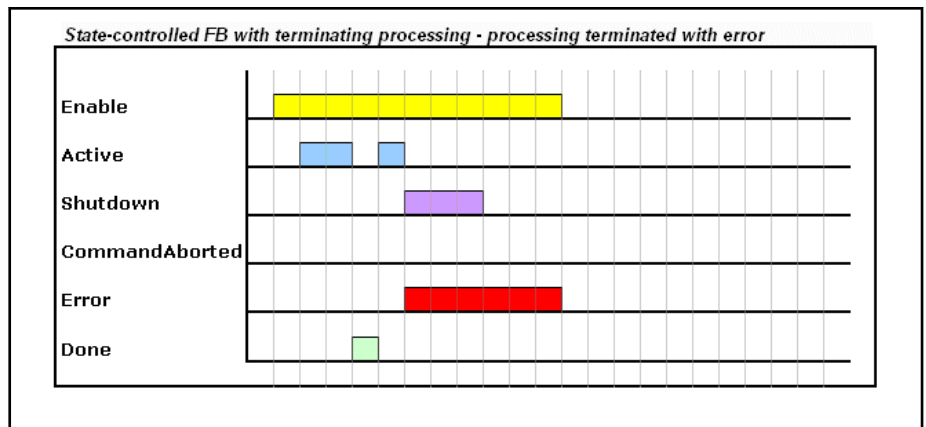


Fig. 2-5: I/O of state-controlled function blocks, definitive processing with "Shutdown". Processing completed with errors.

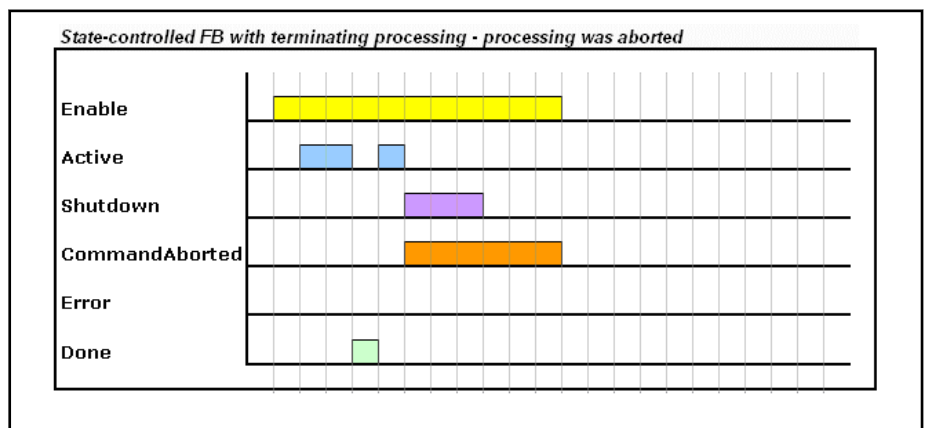


Fig. 2-6: I/O of state-controlled function blocks, definitive processing with "Shutdown". Processing interrupted.

Standard Interfaces at Function Blocks

2.5 Inputs and Outputs of Edge-controlled Function Blocks, Definitive Processing

Control/ Processing	I/O	Name	Description
Edge-controlled/terminating	E	Execute	The input variables are registered with a positive edge at "Execute". New input values do not become effective until there is the next positive edge at "Execute". The change of edges at "Execute" is sufficient to start the FB. The state of "Execute" is irrelevant for further processing. With a new edge change during the processing, the previous task is discarded, the inputs are taken over 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 FB documentation brings clarity.
	A	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 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 FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	Active	If "Active" is TRUE, the function block is processing its actual task and is then in an intermediary state. Possible preprocessing are not marked with this output! "Done", "Error", "Shutdown" and "CommandAborted" are FALSE!
	A	Shutdown (optional)	If a "Shutdown" output is available, the FB needs several cycles of calculation time after having completed its task (signalized via "Done") in order to reach a defined final state and to release the resources used. The FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	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 control input "Execute" is FALSE when the order is completed, "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 FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	Error	If "Error" is TRUE, the function block has been interrupted due to an error and is now in a final state. If 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 FB must be called as long as necessary for "Shutdown" to become FALSE.

Tab. 2-4: I/O of state-controlled function blocks, terminating processing

Standard Interfaces at Function Blocks



If certain inputs are applied not only with the 0/1 edge at "Execute", but cyclically as long as the FB is processed, this must be explicitly documented; it is not allowed to specify this by means of variable names!



If a function block is provided with "Shutdown", the behavior of this FB when an edge is redetected at "Execute" while "Shutdown" remains set (retrigger), must be precisely defined.

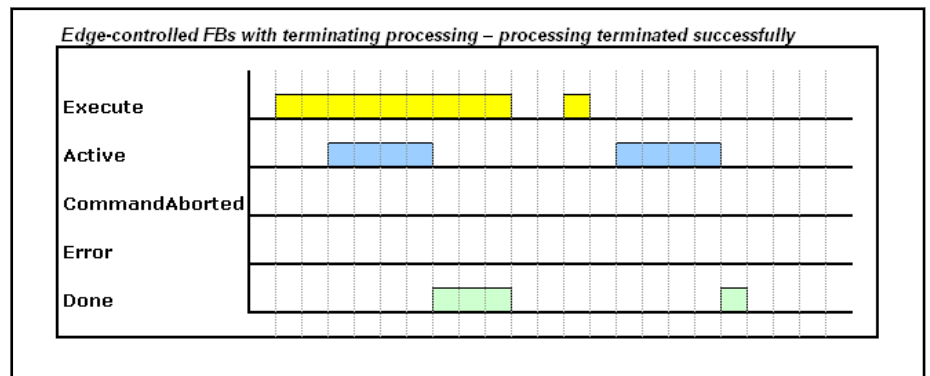


Fig. 2-7: I/O of edge-controlled function blocks, definitive processing without "Shutdown". Processing successfully completed

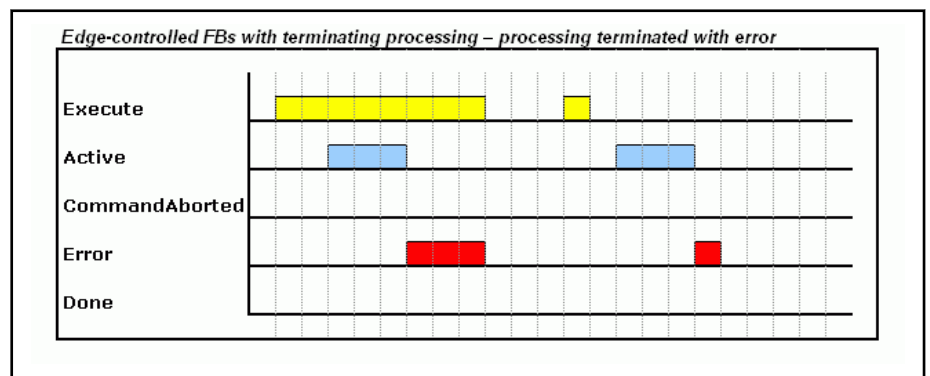


Fig. 2-8: I/O of edge-controlled function blocks, definitive processing without "Shutdown". Processing completed with error

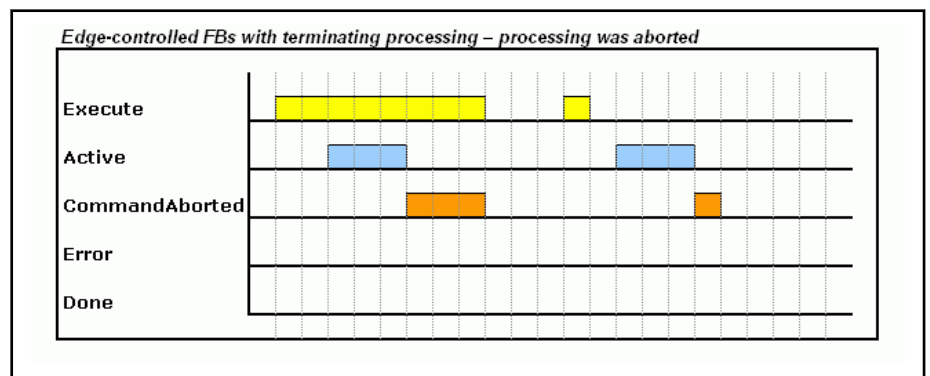


Fig. 2-9: I/O of edge-controlled function blocks, definitive processing without "Shutdown". Processing interrupted

Standard Interfaces at Function Blocks

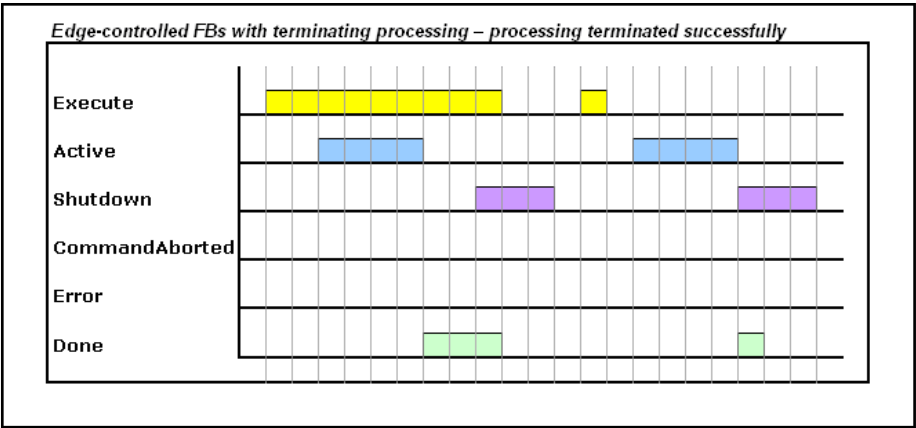


Fig. 2-10: I/O of edge-controlled function blocks, definitive processing with "Shutdown". Processing successfully completed

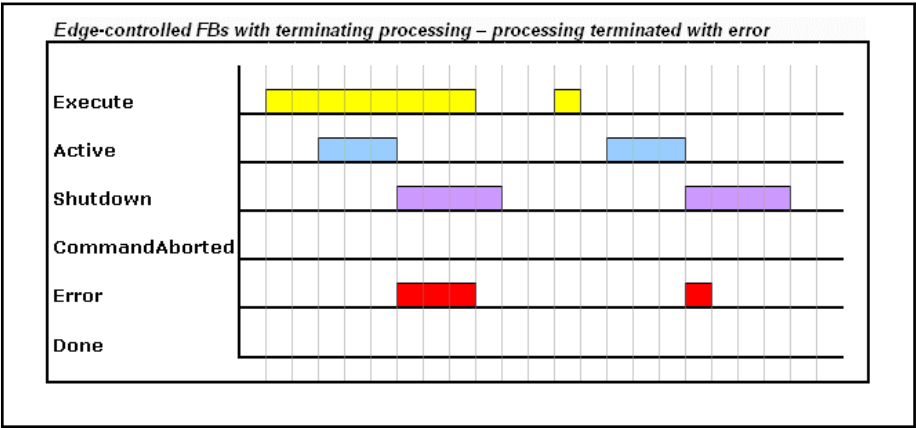


Fig. 2-11: I/O of edge-controlled function blocks, definitive processing with "Shutdown". Processing completed with error

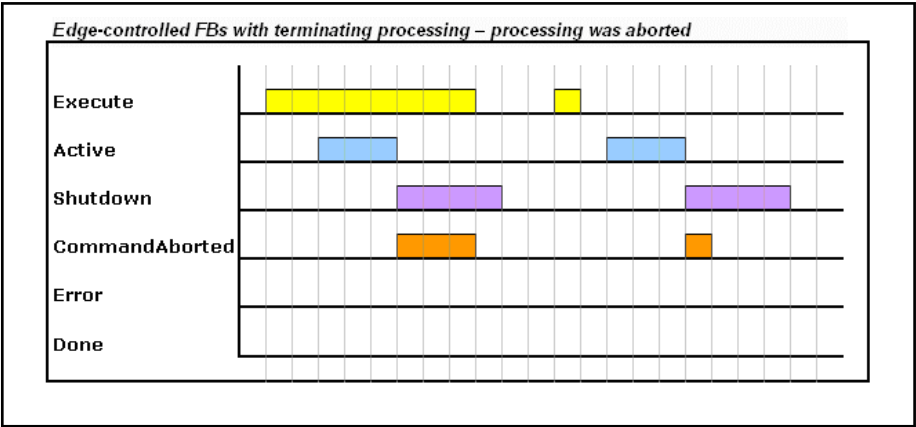


Fig. 2-12: I/O of edge-controlled function blocks, definitive processing with "Shutdown". Processing interrupted

2.6 Inputs and Outputs of State-controlled Function Blocks, Permanent Processing

Control/ Processing	I/O	Variable name	Description
State-controlled/ permanent	E	Enable	The input variables are registered with a positive edge at "Enable". New input values do not become effective until there is the next positive edge at "Enable". "Enable" has to remain TRUE while the function block is processed! If "Enable" is disabled, processing is interrupted and the "In...", "Active", "CommandAborted" and "Error" outputs are set to FALSE. If required, "Shutdown" signalizes that further signals are necessary for the FB to reach its final state defined.
	A	In....	If "In...." is TRUE, the function block has reached its goal but is still in engagement in order to "keep" what it has achieved and is thus in a continuous final state. Data outputs are valid now. Since the FB remains in engagement, "Active" must also remain TRUE. "Error" and "CommandAborted" are FALSE! As long as "Enable" is TRUE, "In...." remains TRUE as well.
	A	Active	If "Active" is TRUE, the function block processes its actual task. Possible pre-processing are not marked with this output! Since the FB remains in continuous engagement, "Active" remains TRUE until the FB is switched off via control input "Enable" or until it is completed by "Error" or "CommandAborted". As long as "Active" is TRUE, "Error" or "CommandAborted" must be FALSE.
	A	Shutdown (optional)	If a "Shutdown" output is available, the FB needs several cycles of calculation time when "Enable" is deactivated in order to reach a defined final state and to release the resources used. The FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	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 FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	Error	If "Error" is TRUE, the function block was canceled due to an error and is now in a final state. "Error" remains TRUE until 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 FB must be called as long as necessary for "Shutdown" to become FALSE.

Tab. 2-5: I/O of state-controlled function blocks, terminating processing



If specific inputs need to be applied not only with an edge at "Enable" but cyclically while the function block is being processed, this has to be explicitly documented. It is not sufficient to specify this by means of variable names!



If a function block is provided with "Shutdown", the behavior of this function block when "Enable" is set to TRUE while "Shutdown" remains set (retrigger), has to be precisely defined.

Standard Interfaces at Function Blocks

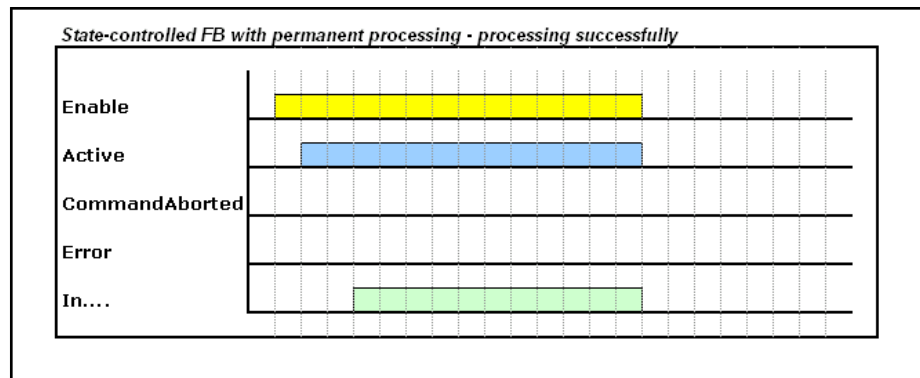


Fig. 2-13: I/O of state-controlled function blocks, continuous processing without "Shutdown". Processing successful

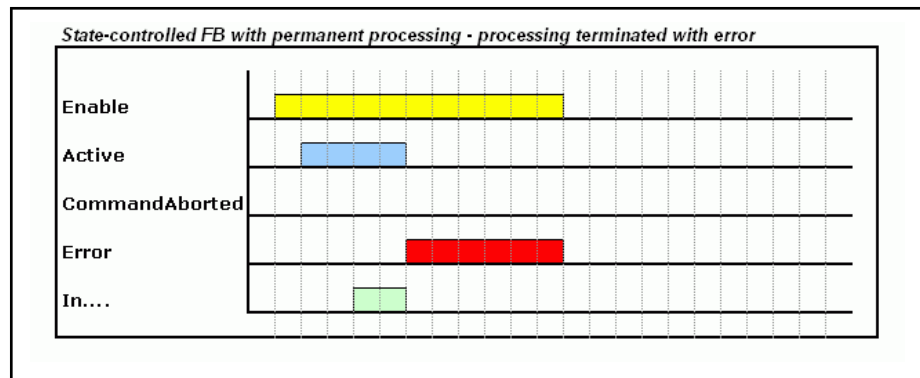


Fig. 2-14: I/O of state-controlled function blocks, continuous processing without "Shutdown". Processing aborted with errors.

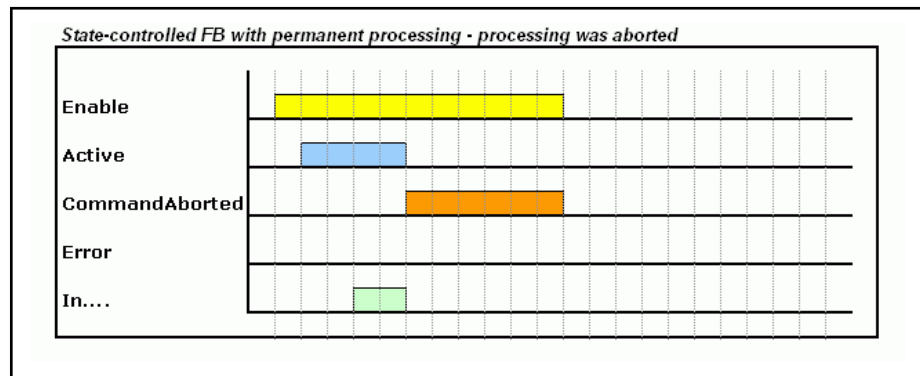


Fig. 2-15: I/O of state-controlled function blocks, continuous processing without "Shutdown". Processing interrupted

Standard Interfaces at Function Blocks

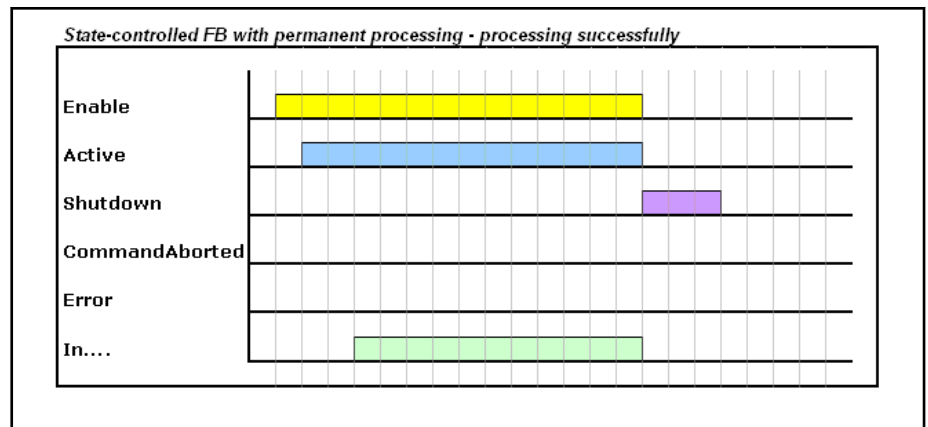


Fig. 2-16: I/O of state-controlled function blocks, continuous processing with "Shutdown". Processing successful

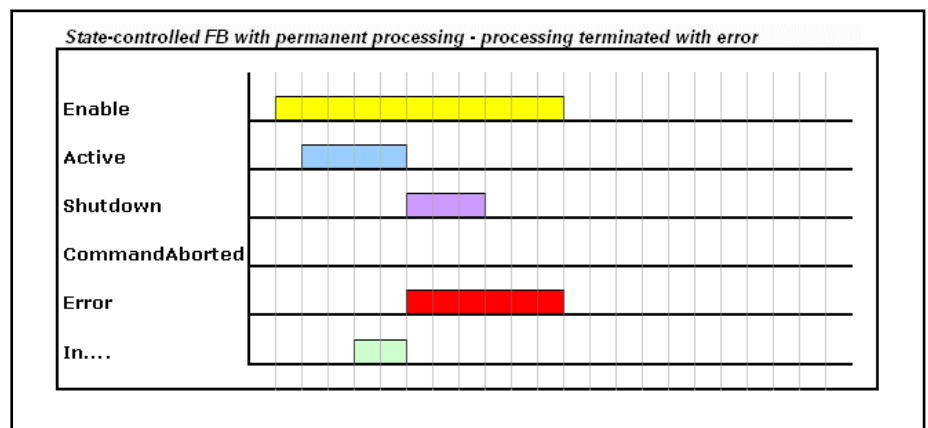


Fig. 2-17: I/O of state-controlled function blocks, continuous processing with "Shutdown". Processing aborted with errors.

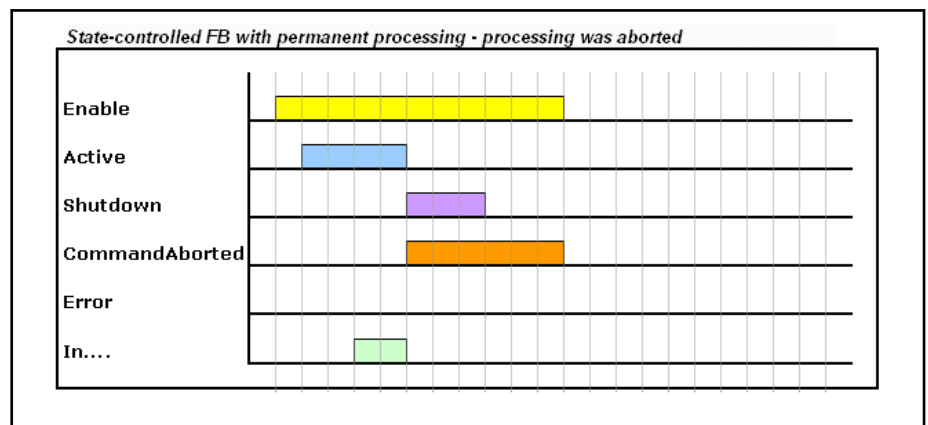


Fig. 2-18: I/O of state-controlled function blocks, continuous processing with "Shutdown". Processing interrupted

Standard Interfaces at Function Blocks

2.7 Inputs and Outputs of Edge-controlled Function Blocks, Permanent Processing

Control/ Processing	I/O	Variable name	Description
Edge-controlled/ permanent	E	Execute	The input variables are registered with a positive edge at "Execute". New input values do not become effective until there is the next positive edge at "Execute". The edge change at "Execute" is sufficient to start the function block. The state of "Execute" is irrelevant for further processing. With a new edge change 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 FB is provided with a "Shutdown" output, retriggering is perhaps not allowed.
	A	In....	If "In...." is TRUE, the function block has reached its goal but is still active in order to keep the results and is thus in a permanent final state. Data outputs are valid now. Since the FB remains in engagement, "Active" must also remain TRUE. This even applies when "In...." drops again. "Error" and "CommandAborted" are FALSE! If 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.
	A	Active	If "Active" is TRUE, the function block processes its actual task. Possible pre-processing are not marked with this output! Since the FB remains in continuous engagement, "Active" remains TRUE until the FB is completed by "Error" or "CommandAborted". As long as "Active" is TRUE, "Error" or "CommandAborted" as well as "Shutdown" must be FALSE.
	A	Shutdown (optional)	If a "Shutdown" output is available, the FB needs several cycles of calculation time after having aborted its task ("CommandAborted" or "Error" = TRUE) in order to reach a defined final state and to release the resources used. The FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	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 control input "Execute" is FALSE when the order is completed, "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 FB must be called as long as necessary for "Shutdown" to become FALSE.
	A	Error	If "Error" is TRUE, the function block has been interrupted due to an error and is now in a final state. If 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 FB must be called as long as necessary for "Shutdown" to become FALSE.

Tab. 2-6: I/O of edge-controlled function blocks, permanent processing

Standard Interfaces at Function Blocks



If certain inputs are applied not only with the 0/1 edge at "Execute", but cyclically as long as the FB is processed, this must be explicitly documented; it is not allowed to specify this by means of variable names!



If a function block is provided with "Shutdown", the behavior of this FB when an edge is redetected at "Execute" while "Shutdown" remains set (retrigger), must be precisely defined.

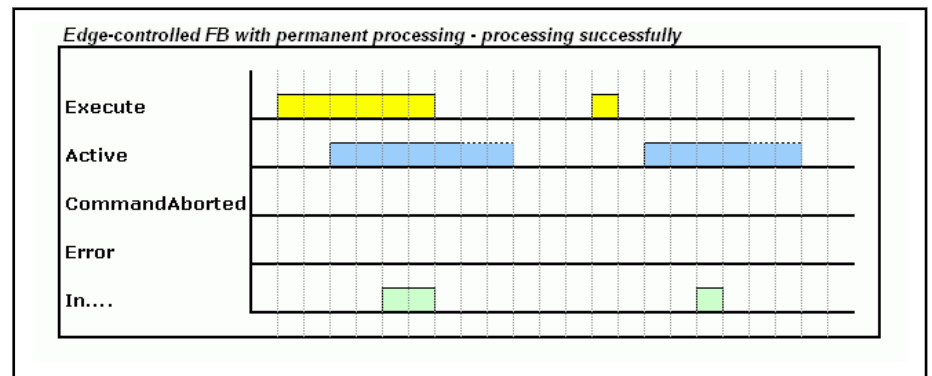


Fig. 2-19: I/O of edge-controlled function blocks, continuous processing without "Shutdown". Processing successful

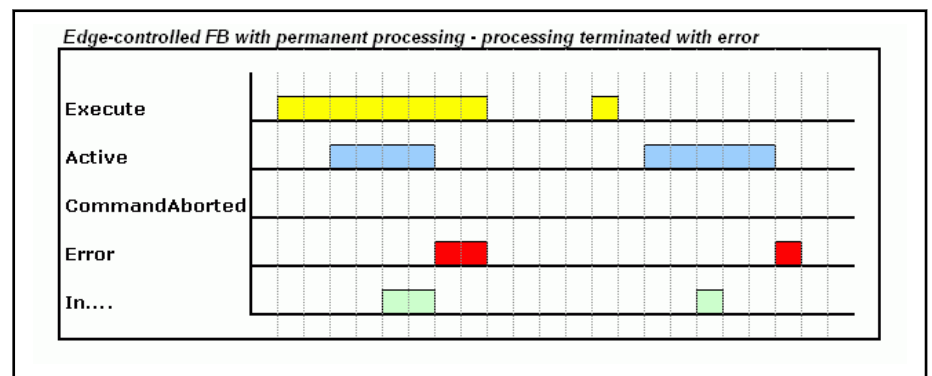


Fig. 2-20: I/O of edge-controlled function blocks, continuous processing without "Shutdown". Processing aborted with errors.

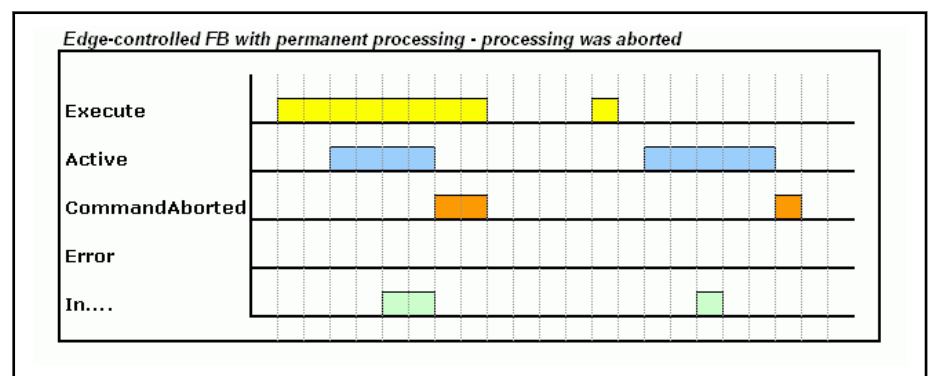


Fig. 2-21: I/O of edge-controlled function blocks, continuous processing without "Shutdown". Processing interrupted

Standard Interfaces at Function Blocks

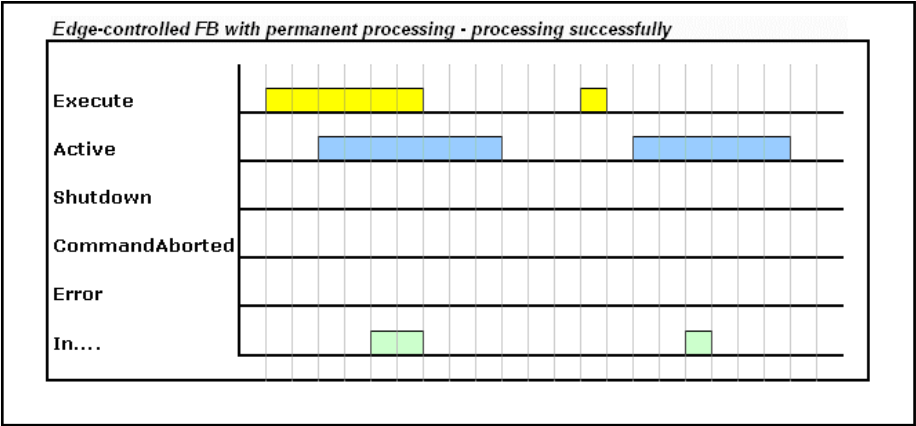


Fig. 2-22: I/O of edge-controlled function blocks, continuous processing with "Shutdown". Processing successful

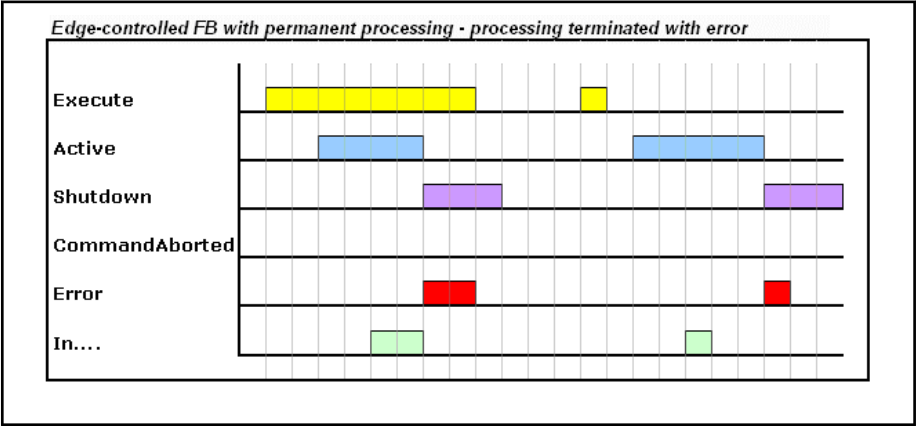


Fig. 2-23: I/O of edge-controlled function blocks, continuous processing with "Shutdown". Processing aborted with errors.

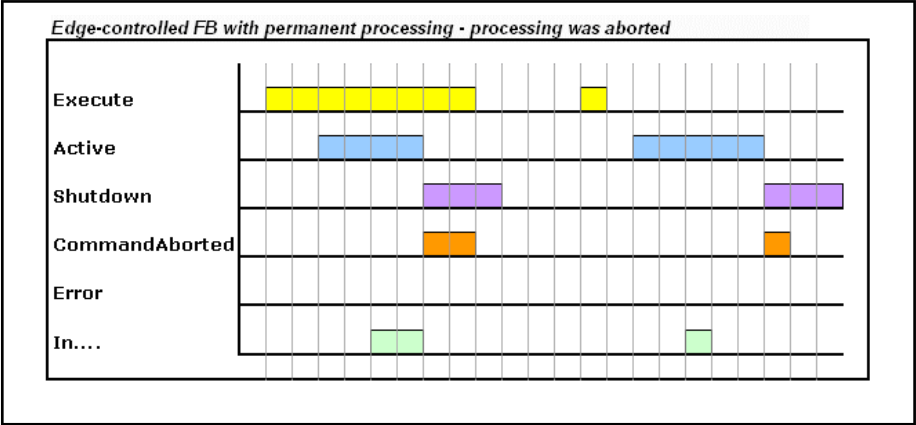


Fig. 2-24: I/O of edge-controlled function blocks, continuous processing with "Shutdown". Processing interrupted

3 RIL_CommonTypes.library

3.1 RIL_CommonTypes, Overview

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

Library structure:

Global_Variables

Global_Variables

- [IL_NULL](#), page 38, NIL pointer.

sercos attributes

- The global variables list contains the sercos attributes, including a comment on their use.

Data types

Data types of logical addresses

- [CONTROLS](#), page 39, (ENUM),
- [KIN_OBJECTS](#), page 41, (ENUM),
- [OBJECTS](#), page 40, (ENUM),
- [TP_OBJECTS](#), page 42, (ENUM),

PLCopen data types

- [AXIS_GROUP_REF](#), page 41, (STRUCT),
- [AXIS_REF](#), page 39, (STRUCT),
- [MC_CAM_ID](#), page 43, (ENUM),
- [MC_START_MODE](#), page 68, (ENUM),
- [MC_STEP_MODE](#), page 44, (ENUM),
- [MC_SYNC_DIRECTION](#), page 70, (ENUM),
- [TOUCHPROBE_REF](#), page 42, (STRUCT),

Data types of POU diagnostics

- [ERROR_STRUCT](#), page 72, (STRUCT)

Includes

- [ERROR_CODE](#), page 71, (ENUM)
- [ERROR_TABLE](#), page 72, (ENUM)

Time data types

- [EXT_SYS_TIME_DATE](#), page 74, (STRUCT)
- [SYS_TIME64](#), page 73, (STRUCT)
- [SYS_TIME_DATE](#), page 74, (STRUCT)

TouchProbe

- [MB_DIFF_EVAL](#), page 75, (ENUM)
- [MB_EXPECT_WINDOW_MODE](#), page 76, (ENUM)

RIL_CommonTypes.library

- [MB_INTERPOLATION_MODE](#), page 77, (ENUM)
- [MB_POSITION_EVAL](#), page 77, (ENUM)
- [MB_PROBE_DATA](#), page 78, (ENUM)
- [MB_PROBE_INPUT](#), page 78, (ENUM)
- [MB_PROBE_MODULE](#), page 77, (ENUM)
- [MB_PROBE_CONFIG](#), page 79, (ENUM)

It was declared to define a uniform access to different IndraMotion 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.

Library	Range
RIL_CommonTypes	Global_Variables

Tab. 3-1: Library assignment

Program:

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

3.3 Data Types for Referencing Objects of IndraMotion Blocks

3.3.1 Referencing of Objects, General

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

Information is transferred via VAR_IN_OUT to the respective function block.

The REF structure contains two 16 bit address elements:

- The control address (element of [CONTROLS](#), page 39) and
- the object address, element of the enumeration type.

REF structure	Enumeration type
AXIS_REF , page 39, axes, currently still programmable limit switches	OBJECTS , page 40
AXIS_GROUP_REF , page 41, axis groups/kinematics	KIN_OBJECTS , page 41
TOUCHPROBE_REF , page 42, touch probes	TP_OBJECTS , page 42

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



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

- the system,
- the control type/drive type and
- the firmware version.

3.3.2 CONTROLS

Brief description

The enumeration contains all controls that can be addressed via IndraMotion function blocks in the IndraMotion network:



In case of a physical address, the "local control" can be reached in two ways, number "0", locally, and via its own address, e.g. CTRL_3.

Library	Range
RIL_CommonTypes.library	Data types of logical addresses

Tab. 3-3: Library assignment

Declaration

```
// 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 IndraMotion function blocks in the IndraControl link:

- Number 0000 is enabled for the local control (the control itself),
- The area of numbers 0001 .. 0999 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, camshaft gear, oscilloscope...).

Library	Range
RIL_CommonTypes	Data types of PLCopen

Tab. 3-4: 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 Motion Control Function Blocks. *)
TYPE AXIS_REF :
STRUCT
    CntrlNo: CONTROLS := LOCAL_CNTRL; (* Control number init: local control *)
    AxisNo: OBJECTS := NO_OBJECT; (* Axis reference number, init: no object *)
END_STRUCT
END_TYPE
```

Functional description

Information is transferred via VAR_IN_OUT to the respective PLC function block.

The AXIS_REF structure contains two 16 bit address elements:

- The control address (element of [CONTROLS page 39](#)) and
- the object address (element of [OBJECTS page 40](#)).

The elements can be addressed via the logical or the 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).

RIL_CommonTypes.library

3.3.4 OBJECTS

Brief description This enumeration of available axes contains all the objects of an IndraMotion link which can be addressed by means of IndraMotion FBs via "AXIS_REF".

Library	Range
RIL_CommonTypes	Data types of logical addresses

*Tab. 3-5: Library assignment
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 deactivated axes.



The allocation of the name and the numbers is carried out when the respective axis/the object is created in the IndraWorks suite (IndraLogic, GlobalMlcVariables).

3.3.5 AXIS_GROUP_REF

Short Description The structure contains information on the desired control and the corresponding axis group.

Library	Range
RIL_CommonTypes	PLCopen data types

Tab. 3-6: Library assignment

Type Definition

```
// Structure of PLCopen element AXIS_GROUP_REF.
// The AXIS_GROUP_REF is a structure that contains information on the axis group.
(* It is used as a VAR_IN_OUT in Motion Control Function Blocks.
TYPE AXIS_GROUP_REF:
STRUCT
  CntrlNo: CONTROLS := LOCAL_CNTRL; // control reference number
                                     // Control number init: local control
  GroupNo: KIN_OBJECTS := NO_KINEMATICS; // object reference number
                                     // axis group reference number, init: NO_KINEMATICS
END_STRUCT
END_TYPE
```

Functional description Information is transferred via VAR_IN_OUT to the respective PLC function block defined in the "ML_PLCopen.library" and related to a group of axes.

The AXIS_GROUP_REF structure contains two 16 bit address elements:

- The control address (element of [CONTROLS page 39](#)) and
- the object address (element of [KIN_OBJECTS page 41](#)).

The elements can be addressed via the logical or the 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).

3.3.6 KIN_OBJECTS

Brief description The enumeration of available kinematics (axis groups) contains all the objects of an IndraMotion network which can be addressed by means of Coordinated Motion blocks, also refer to ([AXIS_GROUP_REF page 41](#)).

Library	Range
RIL_CommonTypes	Data types of logical addresses

Tab. 3-7: Library assignment

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,
  (* no object specified
   *)
  (* synchronize axis group 1 *)
  (* synchronize axis group 2 *)
  (* synchronize axis group 3 *)
  (* synchronize axis group 4 *)
  *)
```


RIL_CommonTypes.library

```

KINEMATICS_5  := 05,      (* synchronize axis group 5 *)
KINEMATICS_6  := 06,      (* synchronize axis group 6 *)
KINEMATICS_7  := 07,      (* synchronize axis group 7 *)
KINEMATICS_8  := 08,      (* synchronize axis group 8 *)
KINEMATICS_9  := 09,      (* synchronize axis group 9 *)
KINEMATICS_10 := 10,      (* synchronize axis group 10 *)
KINEMATICS_11 := 11,      (* synchronize axis group 11 *)
KINEMATICS_12 := 12,      (* synchronize axis group 12 *)
KINEMATICS_13 := 13,      (* synchronize axis group 13 *)
KINEMATICS_14 := 14,      (* synchronize axis group 14 *)
KINEMATICS_15 := 15,      (* synchronize axis group 15 *)
KINEMATICS_16 := 16,      (* synchronize axis group 16 *)
);
END_TYPE

```



The allocation of the name and the numbers is carried out when the object is created in the IndraWorks suite (IndraLogic, Global-MlcVariables)



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

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).

Library	Range
RIL_CommonTypes	Data types of PLCopen

Tab. 3-8: Library assignment

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 *)
  TouchProbeNo: TP_OBJECTS := NO_TOUCHPROBE (* object reference number *)
END_STRUCT
END_TYPE

```

Functional description Information is transferred via VAR_IN_OUT to the respective PLC function block.

The TOUCHPROBE_REF structure contains two 16 bit address elements:

- The control address (element of [CONTROLS page 39](#)) and
- the object address (element of [TP_OBJECTS page 42](#)).

3.3.8 TP_OBJECTS

Brief description The enumeration of available touch probes (Touch Probe) contains all the objects of an IndraMotion network which can be accessed by means of touch probe blocks, also refer to ([TOUCHPROBE_REF page 42](#)).

Library	Range
RIL_CommonTypes	Data types of logical addresses

Tab. 3-9: Library assignment

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

```



The allocation of the name and the numbers is carried out when the respective touch probe is manually created in the IndraWorks suite (IndraLogic, GlobalMlcVariables)



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

3.4 Data Types for the Support of Cams, MotionProfiles and FlexProfiles

3.4.1 Data Type for Switching between Cams, General

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. Selection and switching can take place via the function block MC_CamIn (CamTableID: [MC_CAM_ID](#) [page 43](#)) 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 desired cam table ([Data Types for the Support of Cams, MotionProfiles and FlexProfiles](#) , [page 43](#)).

Library	Area
RIL_CommonTypes	PLCopen data types

Tab. 3-10: Library assignment

Program:

```

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

```

Functional description

Cam table switching is possible at runtime. The selection and switching can be carried out via the function block MC_CamIn, ML_PLCopen.library or via the parameter control "A-0-2740, ELS, CamIn, cam table preselection".

RIL_CommonTypes.library

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.

Library	Area
RIL_CommonTypes	PLCopen data types

Tab. 3-11: Library assignment

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, (* 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, *)
                                         (* also Flex motion step *)
    MOTION_IN_MOTION_POLY7          := 16#2100, (* FlexProfile, *)
                                         (* 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 *)

    (* 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, *)
                                         (* also Flex motion step *)
    X_FIT_VEL_TRAPEZE_ALIM              := 16#7000, (* FlexProfile *)
    X_FIT_SINE_TRAPEZE_ALIM             := 16#7100, (* FlexProfile *)
    X_FIT_ACC_TRAPEZE_JLIM              := 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, *)

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

```

Functional description



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

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

The following modes are available:

- R-R: Rest-in-rest, Polynomial 5th order,
- R-R: Rest-in-rest, inclined sine curve,
- R-V: Rest-in-velocity, Polynomial 5th order,
- V-R: Velocity-in-rest, Polynomial 5th order,
- V: 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)	Automatic	Automatic

Tab. 3-12: Marginal values for the motion law - Polynomial 5th order

RIL_CommonTypes.library

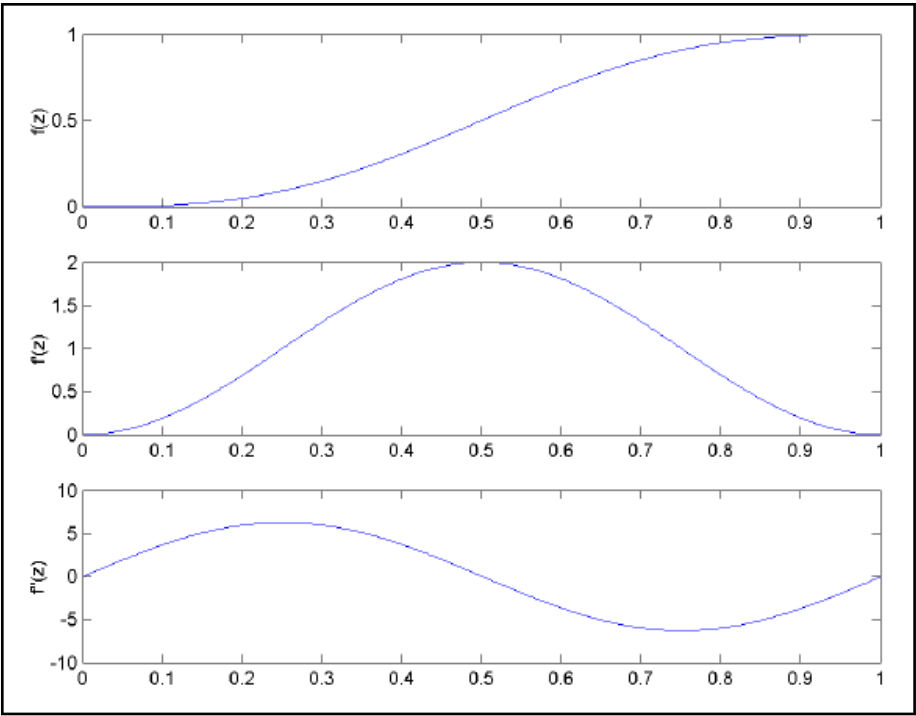


Fig. 3-2: Rest-in-rest, Polynomial 5th order

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

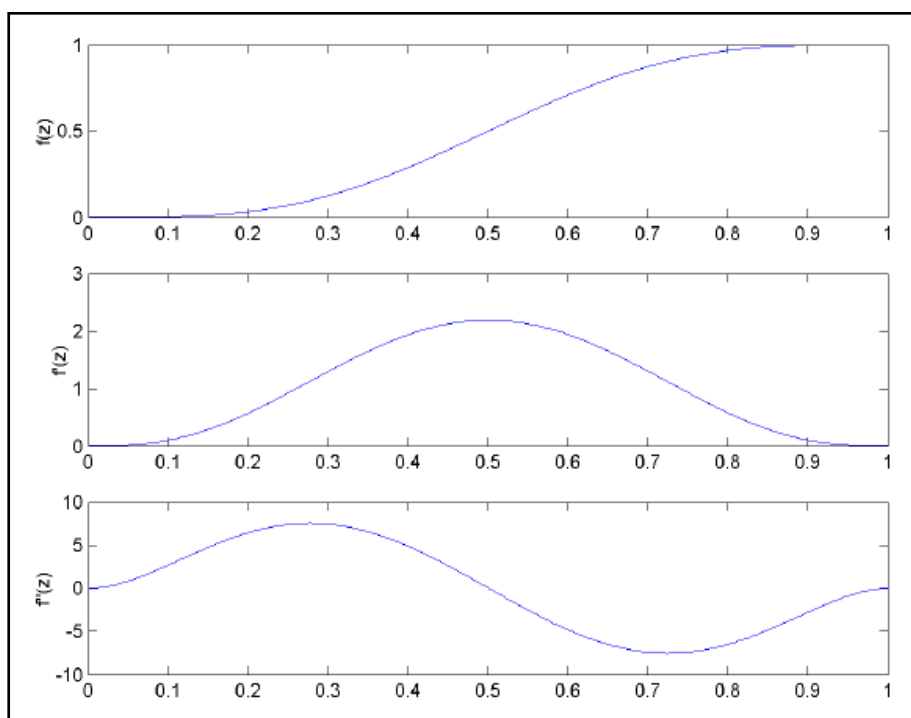


Fig. 3-4: Rest-in-rest, Polynomial 7th order

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)	Automatic	Automatic

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

RIL_CommonTypes.library

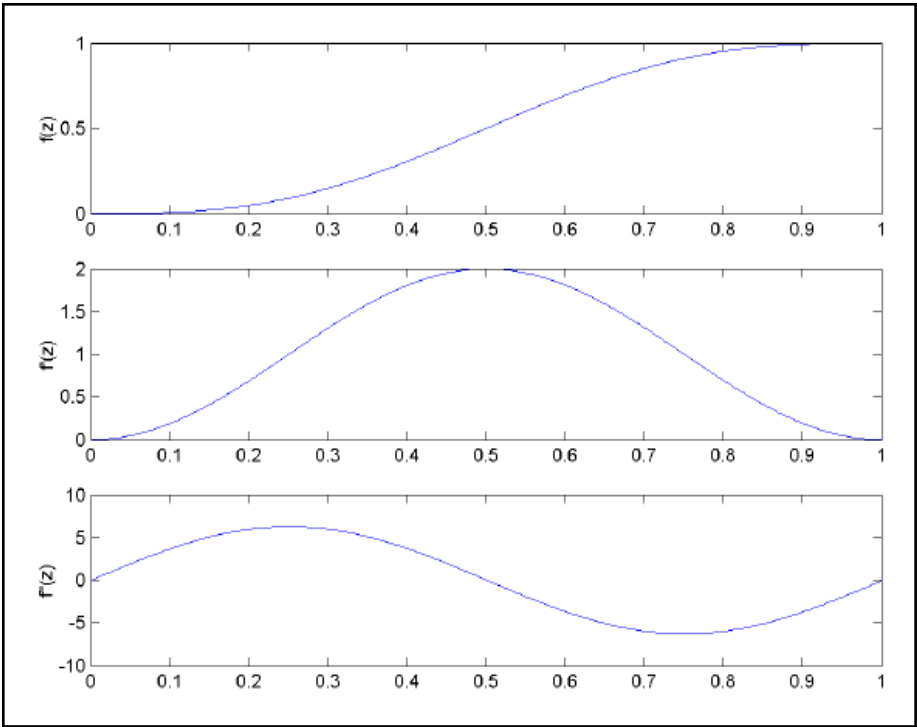


Fig. 3-6: Rest-in-rest, Inclined sine curve (BESTEHORN)

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

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

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)	Automatic	Automatic
3. Order (jerk)	Automatic	Automatic

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

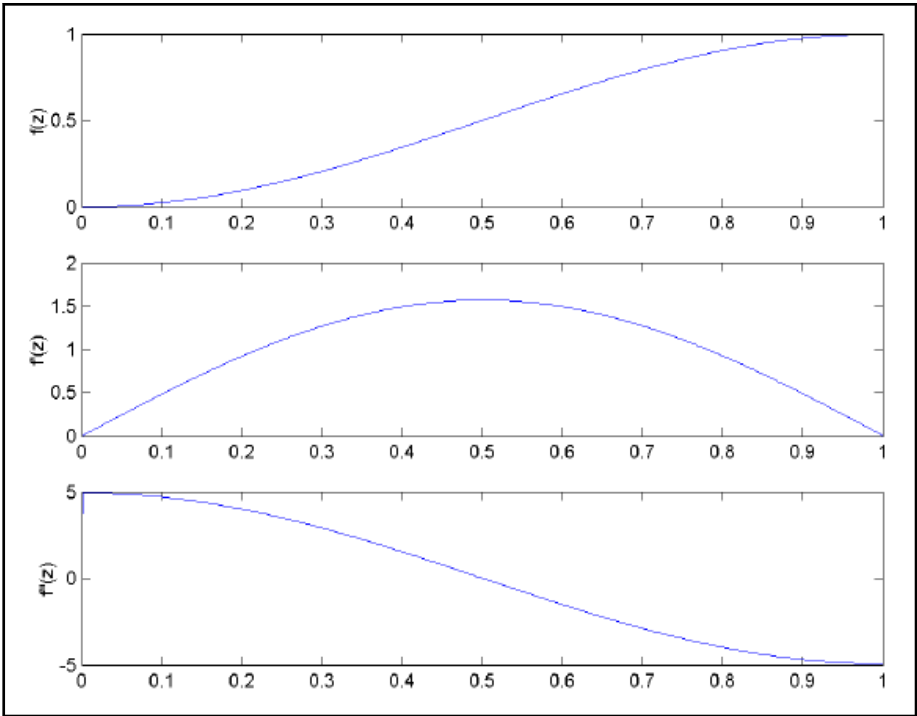


Fig. 3-8: Rest-in-rest, Sine curve (simple sinoside)

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

RIL_CommonTypes.library

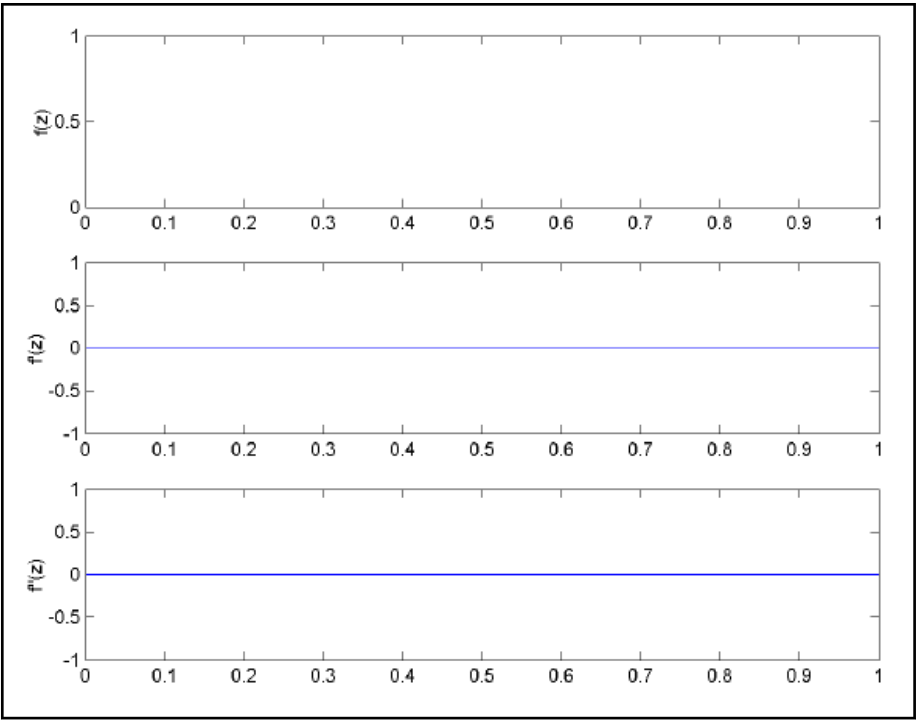


Fig. 3-10: Linear rest

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 sinoide

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)	Automatic	Automatic

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

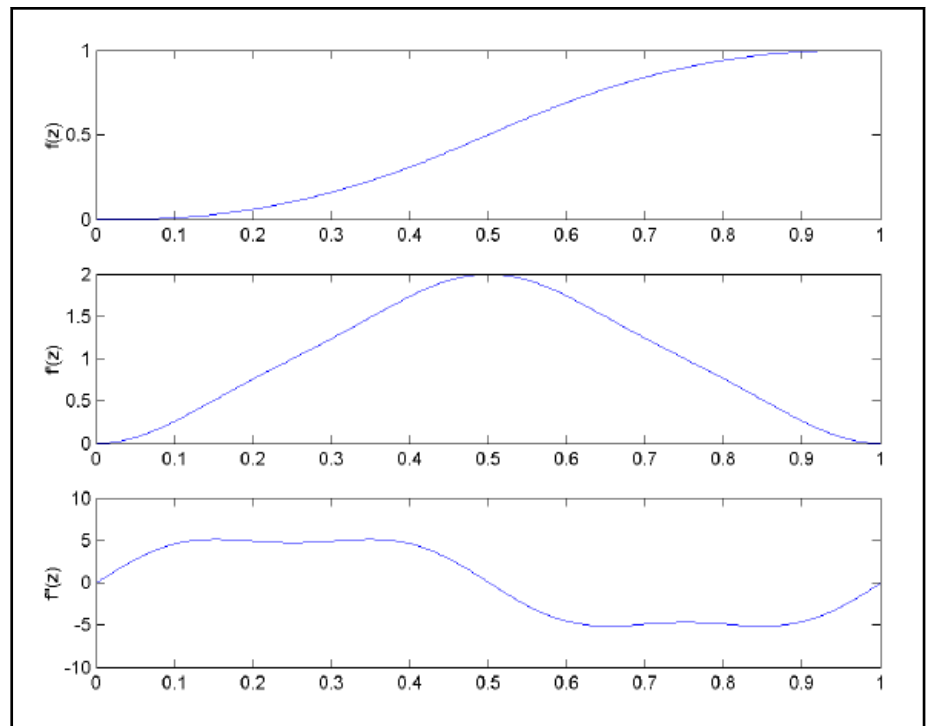


Fig. 3-12: Rest-in-rest, Gutman sine

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: Inclined sine curve optimum for acceleration

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)	Automatic	Automatic

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

RIL_CommonTypes.library

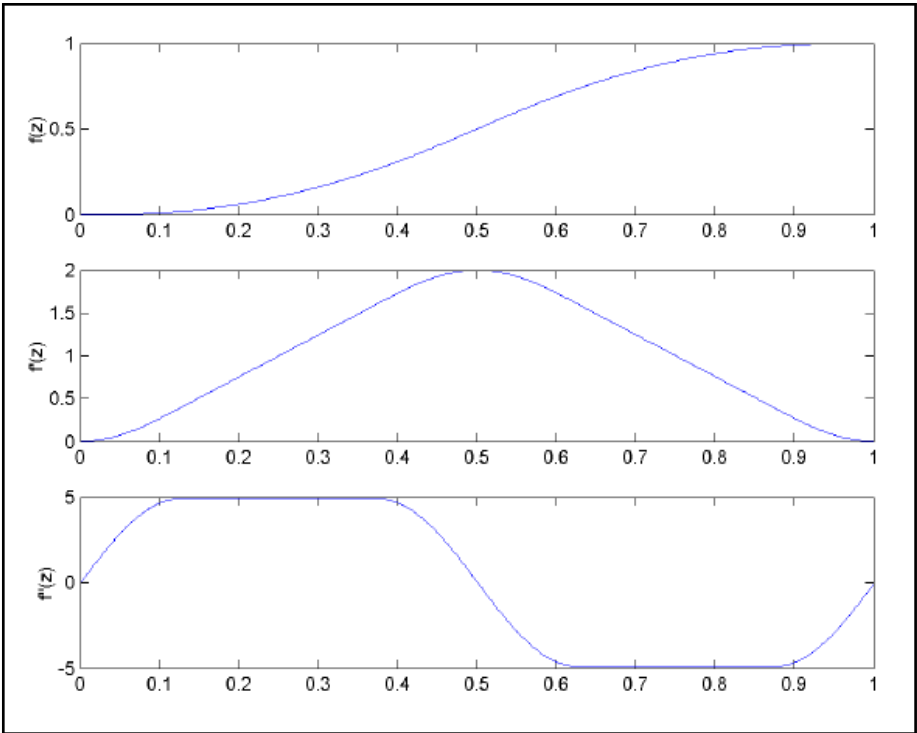


Fig. 3-14: *Rest-in-rest, acceleration-optimum inclined sine curve*

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: *Inclined sine curve optimum for the torque*

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)	Automatic	Automatic

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

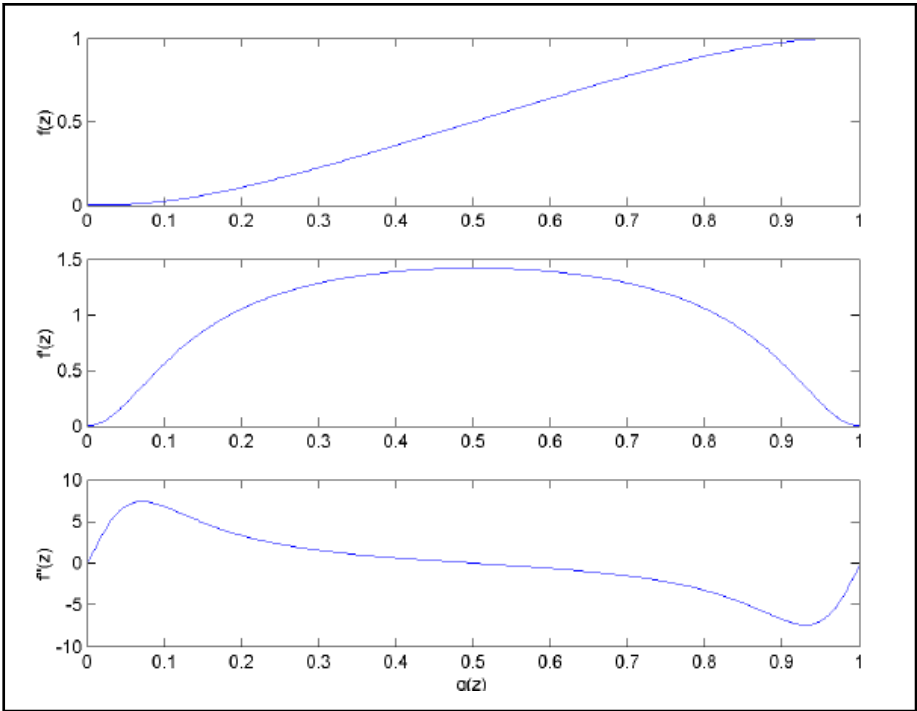


Fig. 3-16: Rest-in-rest, torque-optimum inclined sine curve
The profile for the "Modified sinoide" is determined by the following function:

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

Fig. 3-17: Modified sinoide
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)	Automatic	Automatic

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

RIL_CommonTypes.library

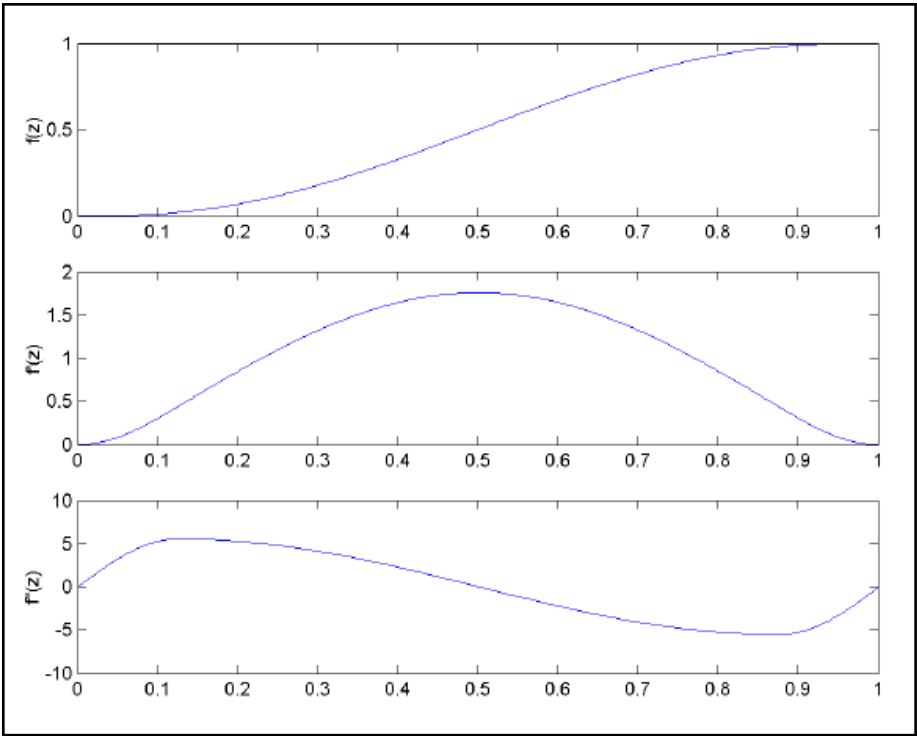


Fig. 3-18: Rest-in-rest, Modified sinoide

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

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

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

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

$$\frac{1}{2} \leq z \leq 1 : f(z) = 1 - f(1 - z)$$

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)	Automatic	Automatic

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

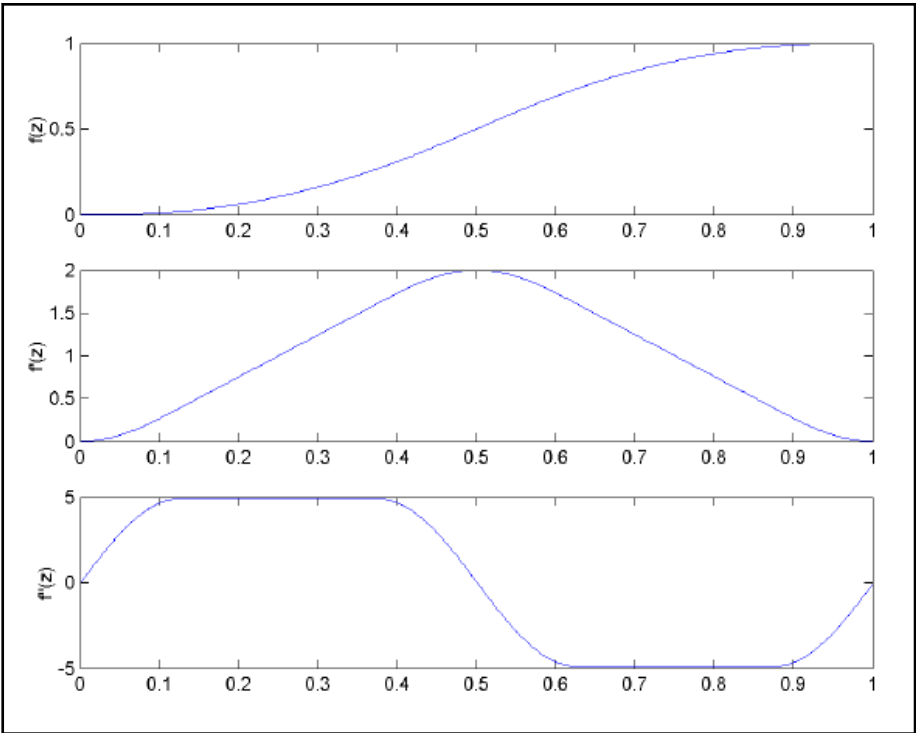


Fig. 3-20: Rest-in-rest, Modified acceleration trapezoid

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":

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

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)	Automatic	Automatic

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

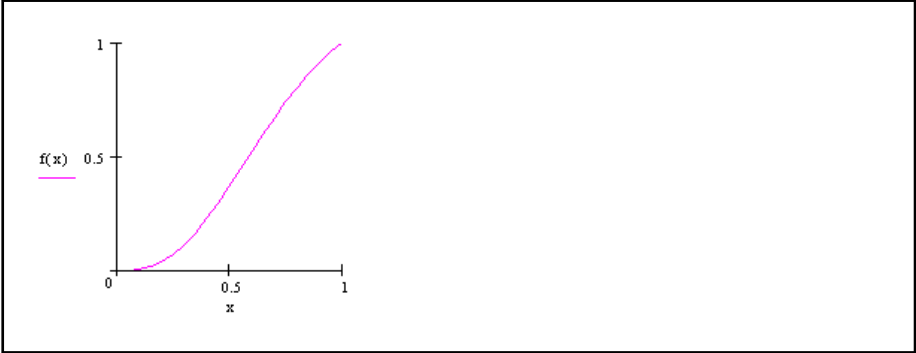


Fig. 3-22: Rest-in-velocity, Polynomial 5th order

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

RIL_CommonTypes.library

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

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)	Automatic	Selectable

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

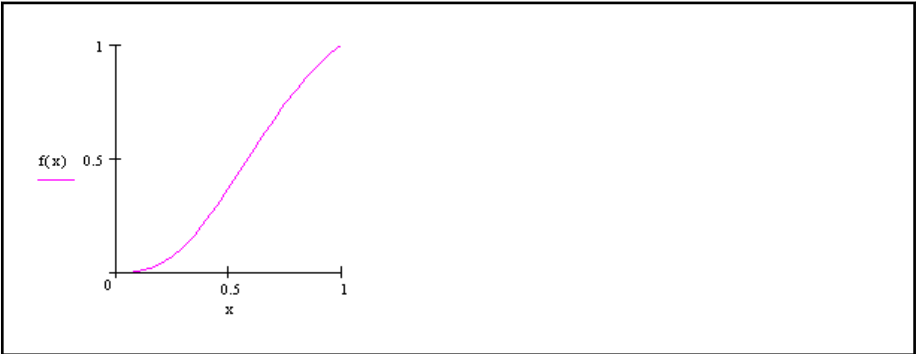


Fig. 3-24: Rest-in-velocity, Polynomial 7th order

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":

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

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)	Automatic	0

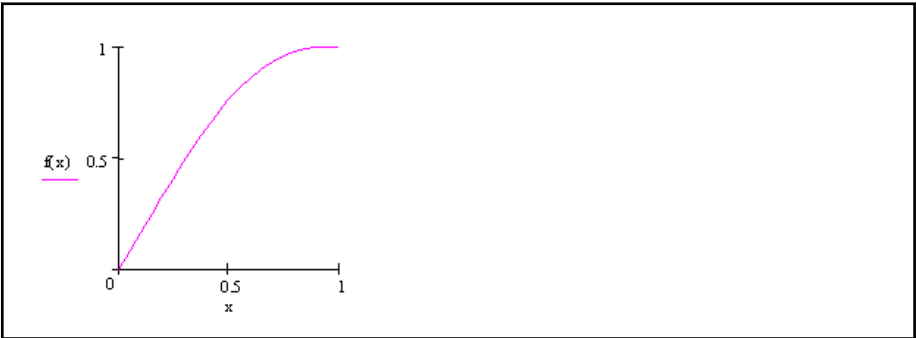


Fig. 3-26: Velocity-in-rest, Polynomial 5th order

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

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

$$f(z) = A_7 z^7 + A_6 z^6 + A_5 z^5 + A_4 z^4 + A_3 z^3 + A_2 z^2 + A_1 z$$

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

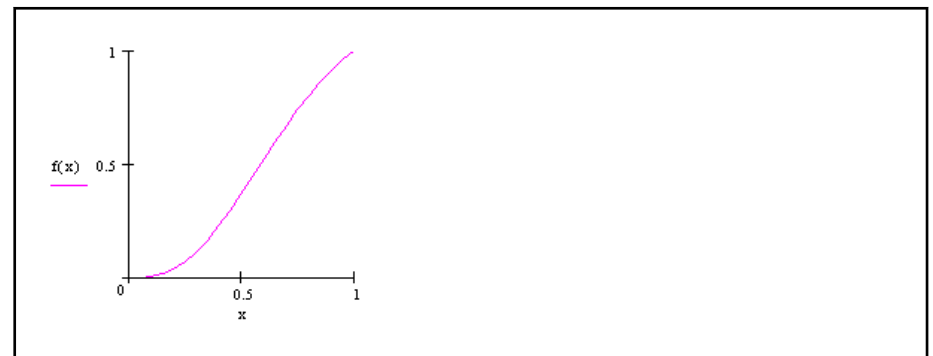


Fig. 3-28: Rest-in-velocity, Polynomial 7th order

V: 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)	Automatic	Automatic
2. Order (acceleration)	Automatic	Automatic
3. Order (jerk)	Automatic	Automatic

RIL_CommonTypes.library

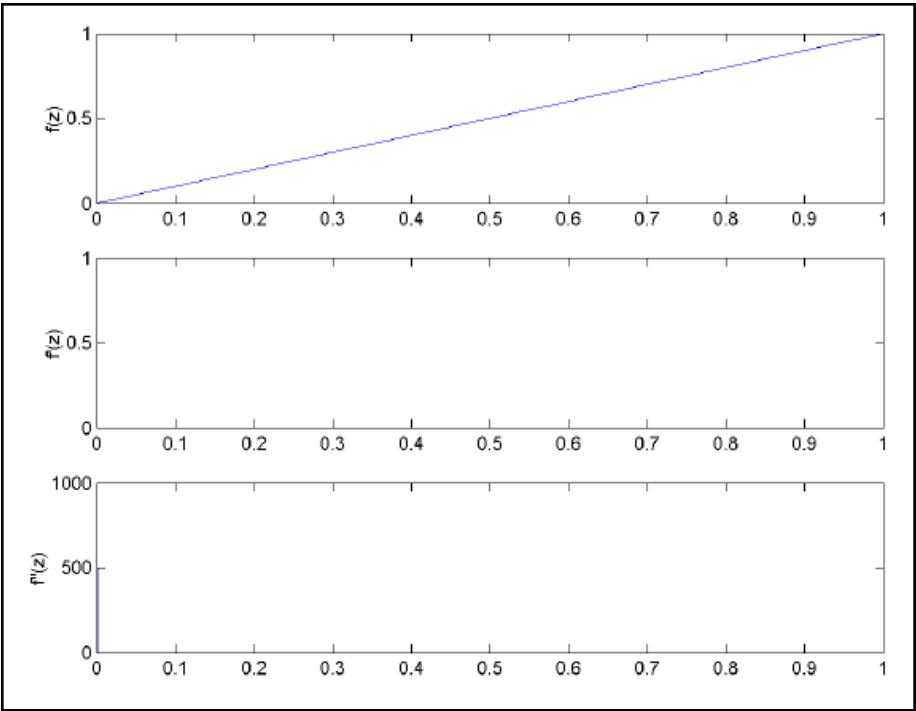


Fig. 3-29: Constant velocity

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)	Automatic	Automatic

Tab. 3-25: Marginal values for the motion law - Polynomial 5th order

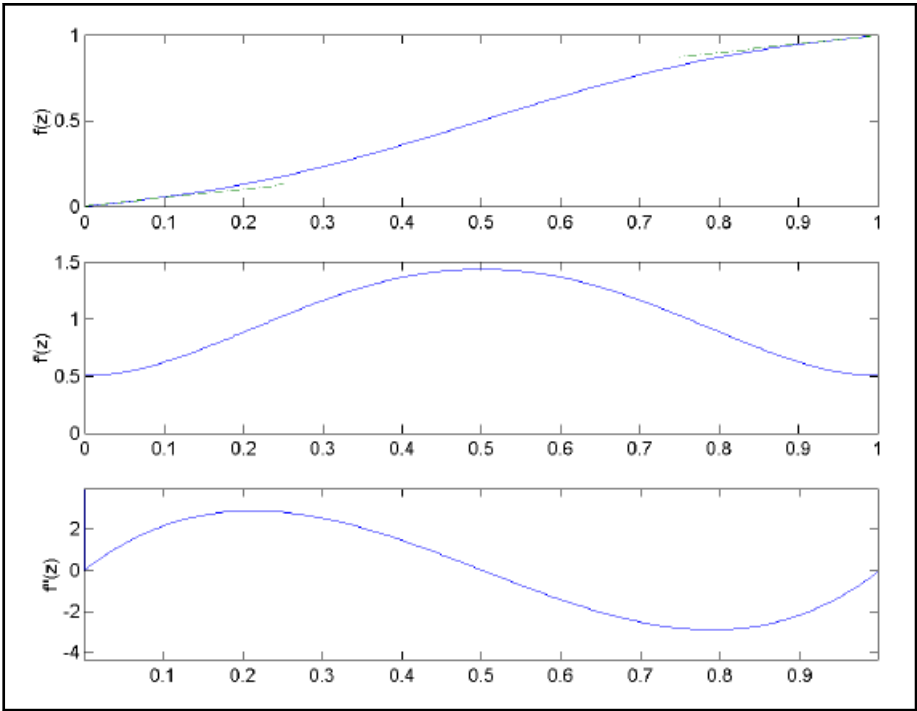


Fig. 3-31: Velocity-in-velocity, Polynomial 5th order

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

$$f(z) = A_7 z^7 + A_6 z^6 + A_5 z^5 + A_4 z^4 + A_3 z^3 + A_2 z^2 + A_1 z$$

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

RIL_CommonTypes.library

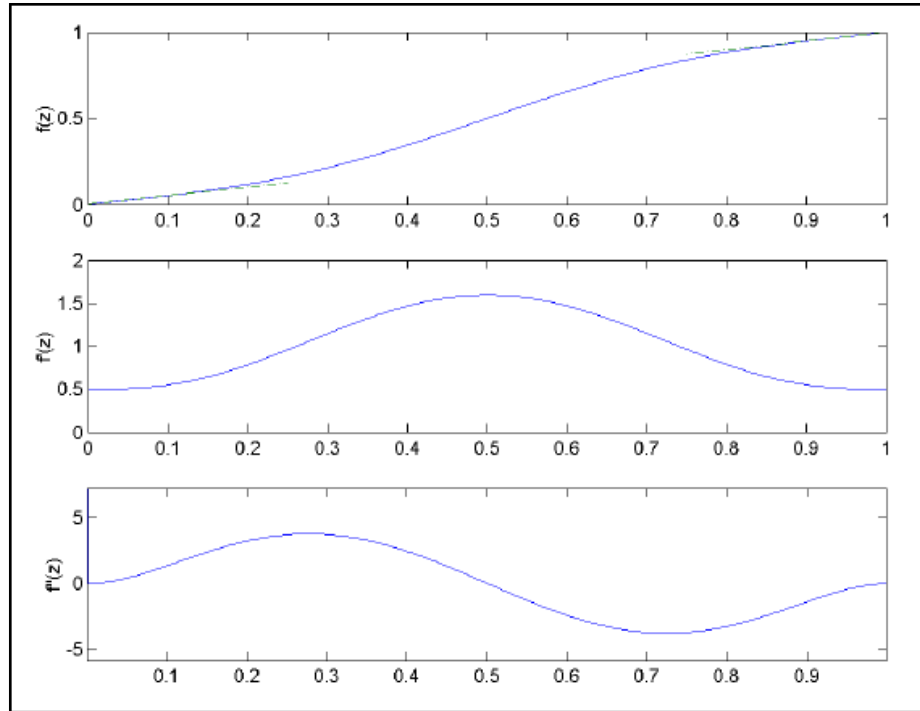


Fig. 3-33: Velocity in velocity, Polynomial 7th order

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

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

Fig. 3-34: Modified sine curve

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 range	$v_{\max}$	$a_{\max}$	$j_{\max}$
No overshoot					
Polynomial 5th order	Selectable	Selectable	Selectable	Automatic	Automatic
Velocity-limited					
Polynomial 5th order	Selectable	Selectable	Selectable	Automatic	Automatic
Acceleration-limited Motion	Selectable	Automatic	Selectable	Selectable	Automatic
Velocity-limited Sinoide profile	Selectable	Automatic	Selectable	Selectable	Automatic
Jerk-limited motion	Selectable	Automatic	Selectable	Selectable	Selectable

Tab. 3-28: Extended Motion Laws

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

The 5th order polynomial 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 of the 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

RIL_CommonTypes.library

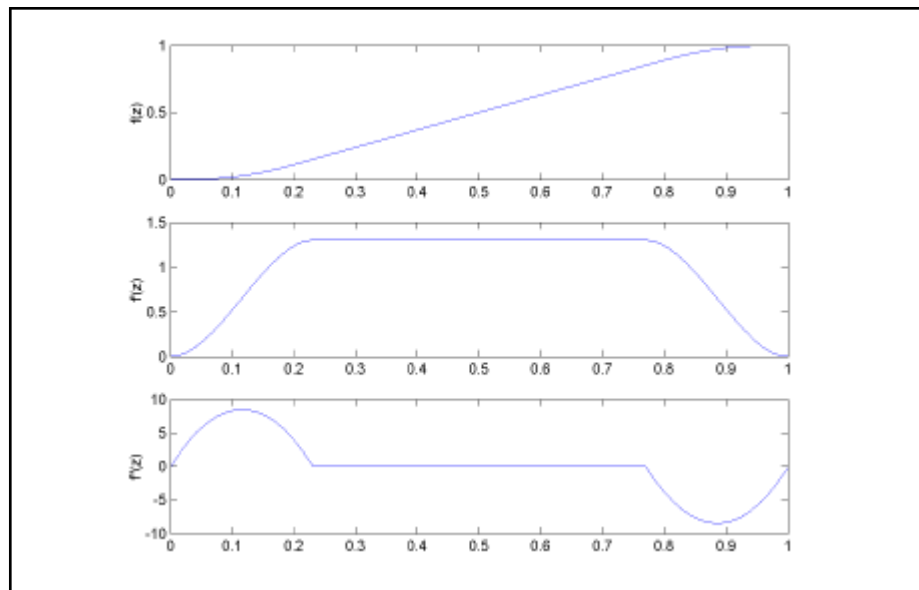


Fig. 3-35: Velocity-limited polynomial 5th order

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 marginal 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 $V_1 \leq 1$

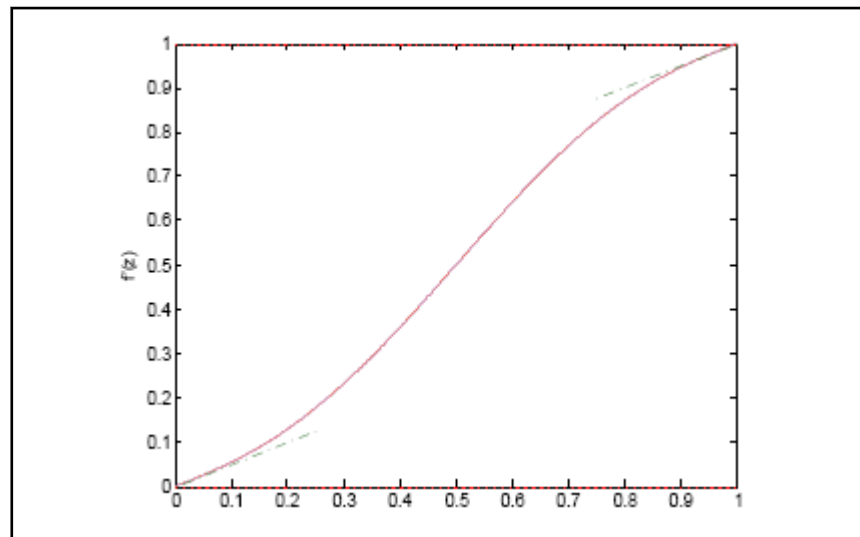


Fig. 3-36: Case 1

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

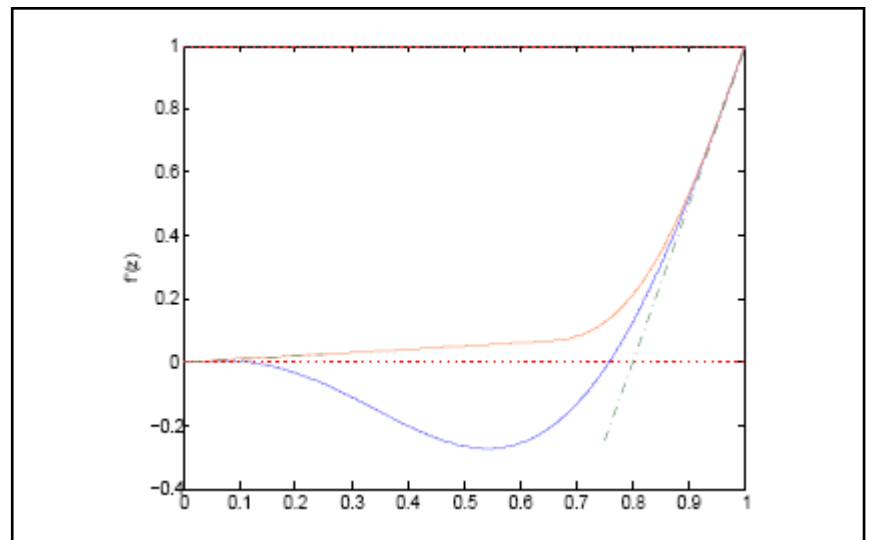


Fig. 3-37: Case 2

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

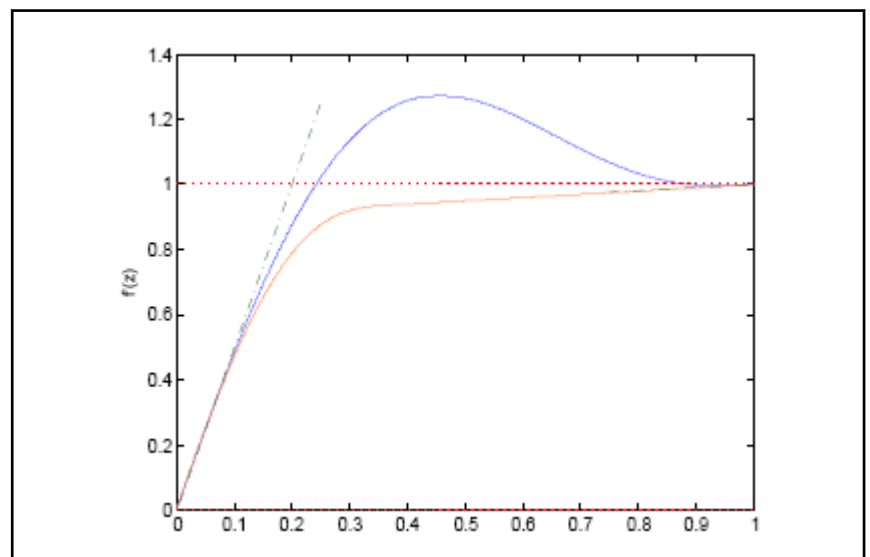


Fig. 3-38: Case 3

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

RIL_CommonTypes.library

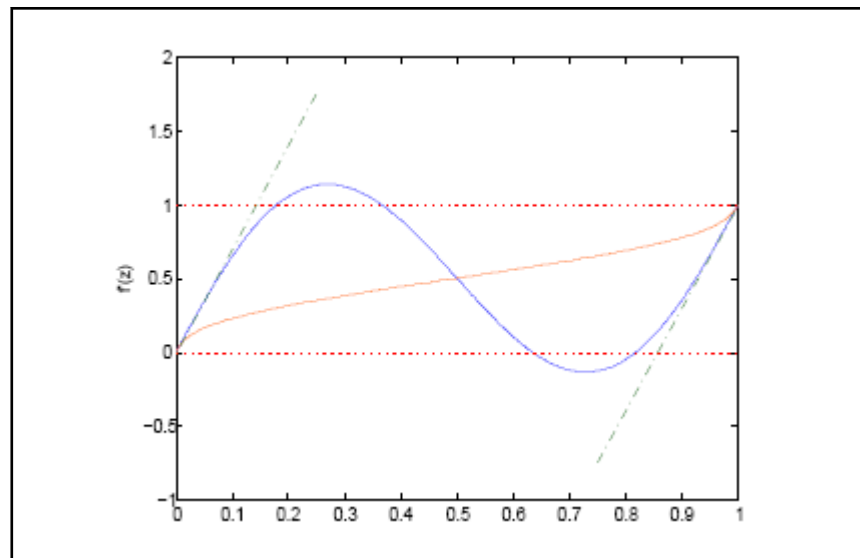


Fig. 3-39: Case 4

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

The trapezoid profile, often also known as a ramp profile, can be regarded 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.

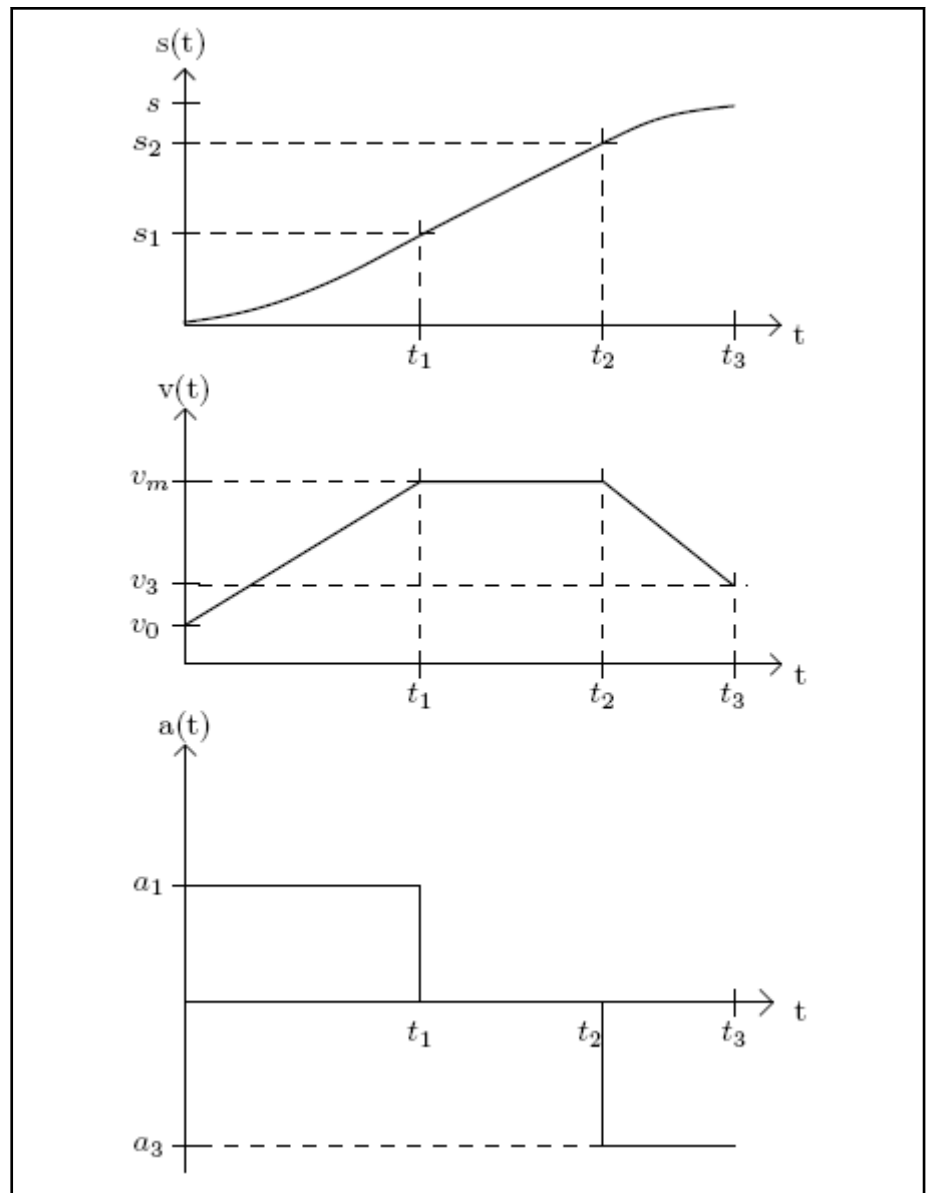


Fig. 3-40: Acceleration-limited motion

The possible marginal values for the motion law are:

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

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

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

With the sine 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 .

RIL_CommonTypes.library

The possible marginal values for the motion law are:

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

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

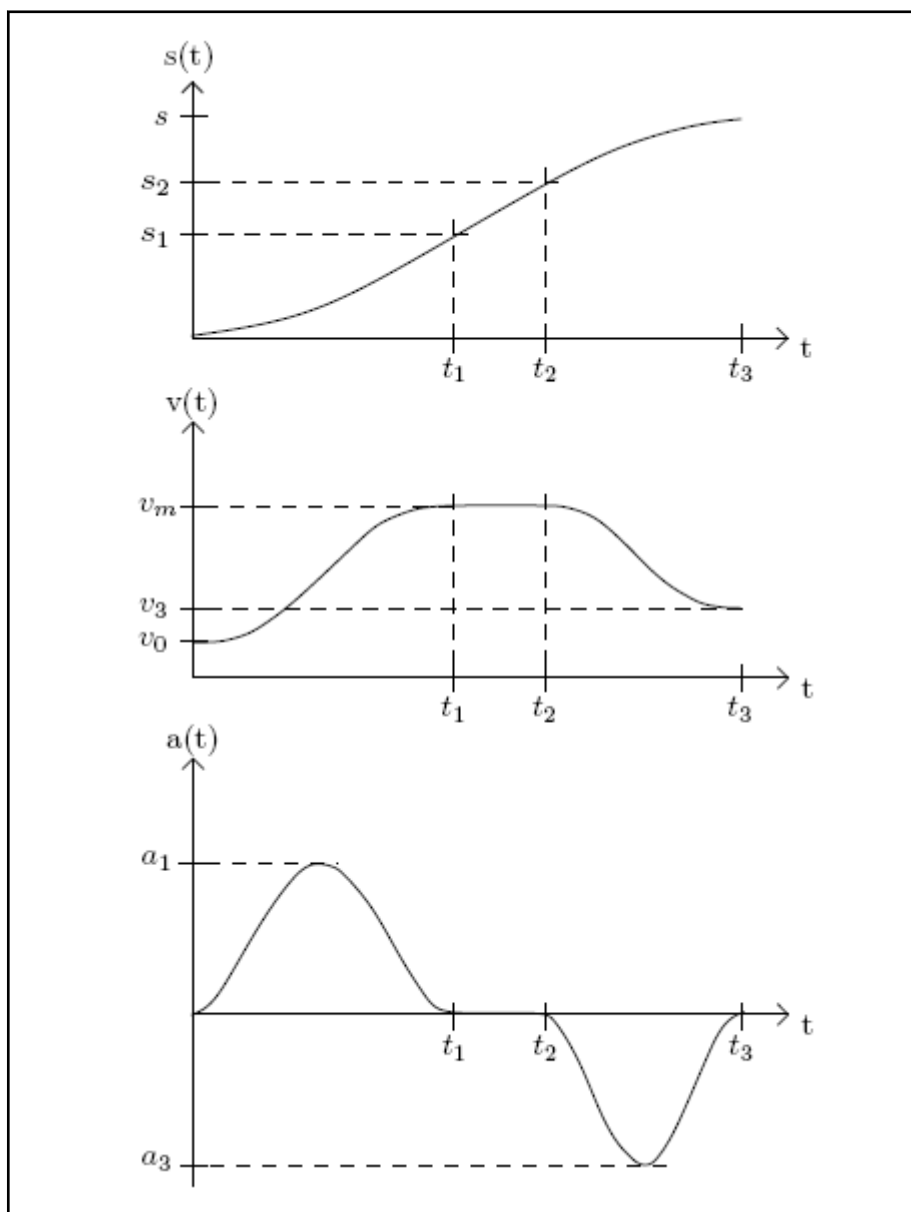


Fig. 3-41: Acceleration-limited motion with sine profile

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.

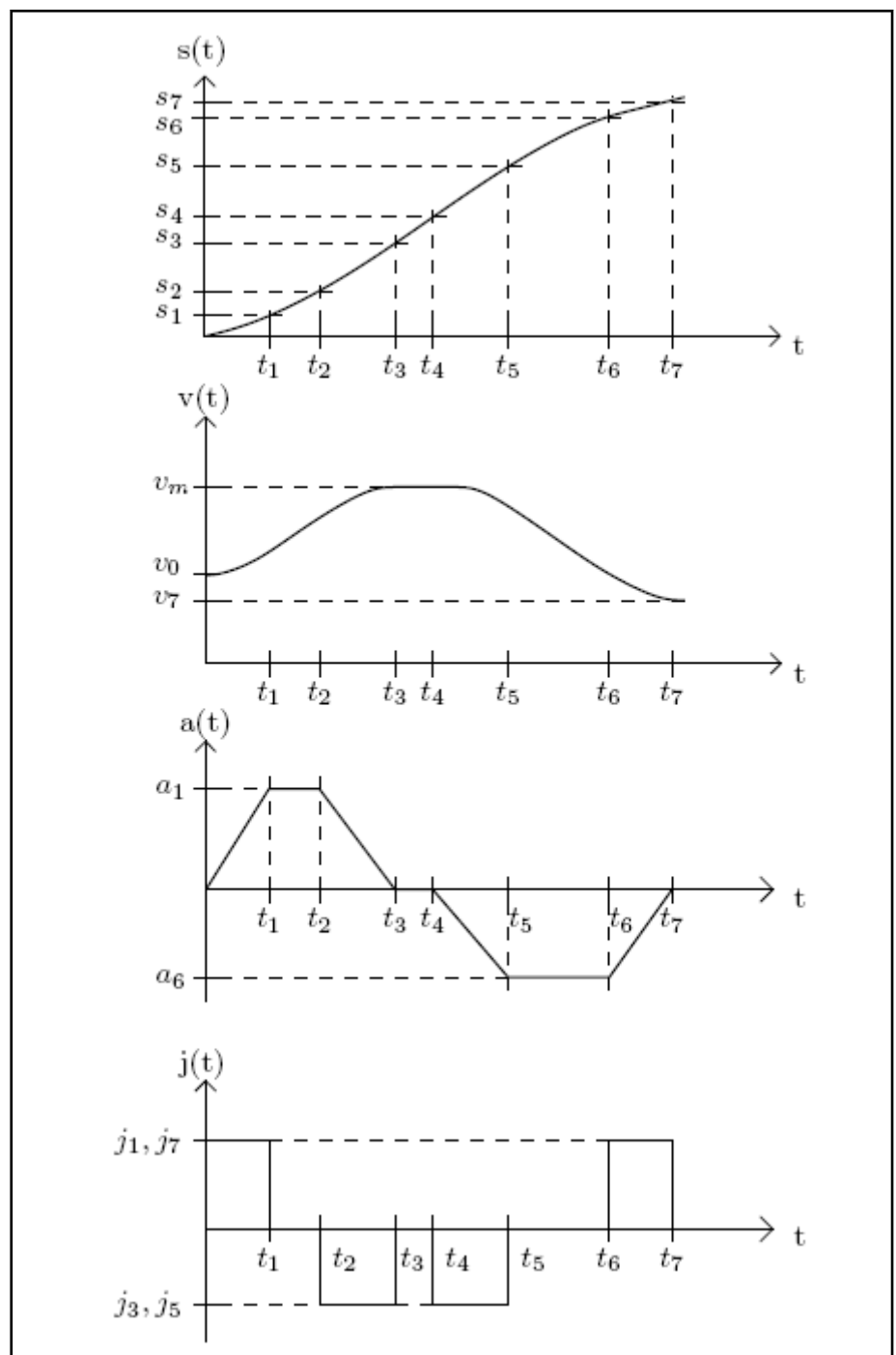


Fig. 3-42: Jerk-limited motion

3.5 Data Types for Synchronization in the Position-controlled Synchronous Operation Modes

3.5.1 General

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:

RIL_CommonTypes.library

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 68):

- 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 70).

3.5.2 MC_START_MODE

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

Library	Range
RIL_CommonTypes	Data types of PLCopen

Tab. 3-31: Library assignment

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

```

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.

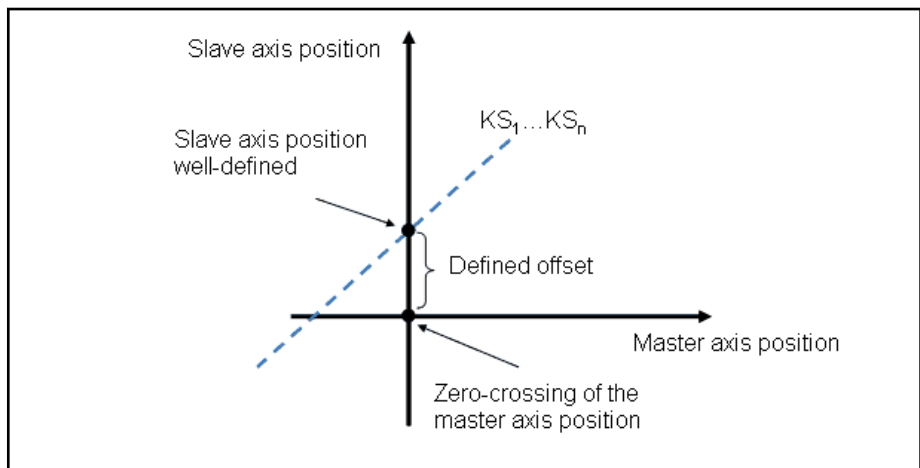


Fig. 3-43: Absolute synchronization

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.

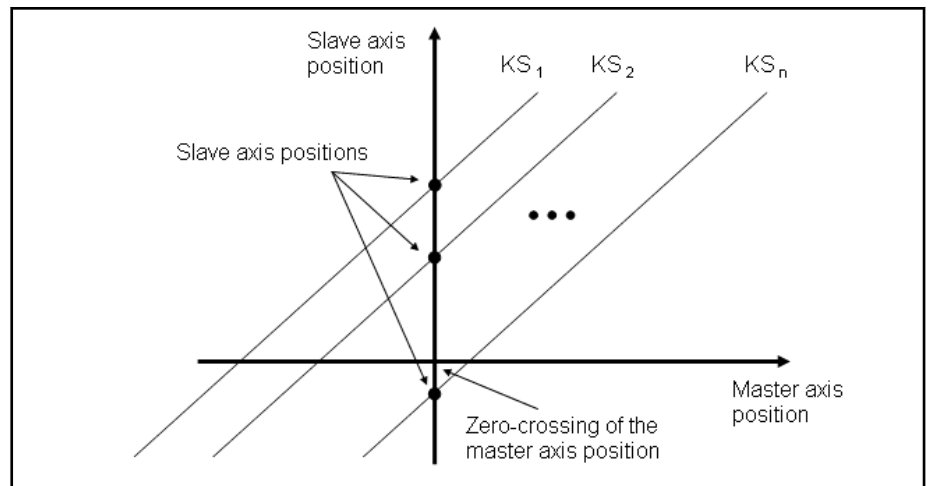


Fig. 3-44: Relative synchronization

"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,

The changeover is carried out after the crossing of the master axis change-over position (A-0-2900) by the key axis.

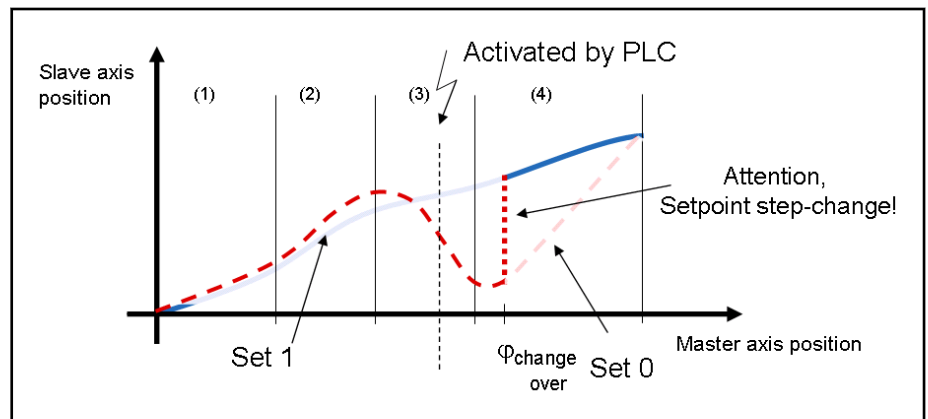


Fig. 3-45: Hard changeover

"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,

RIL_CommonTypes.library

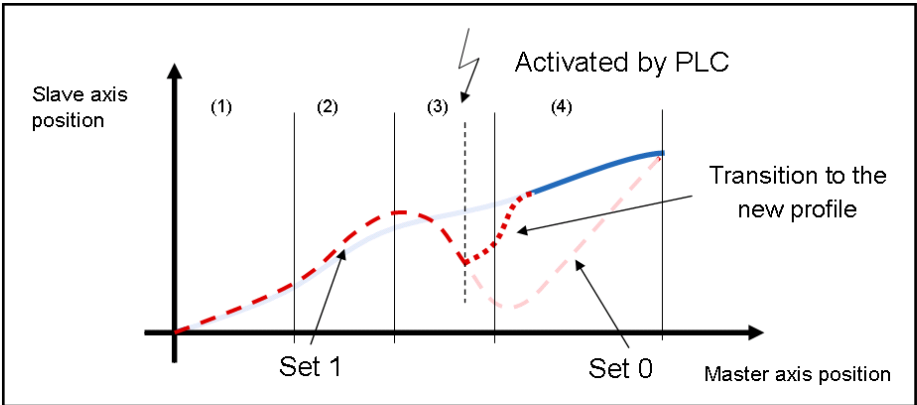


Fig. 3-46: Soft changeover
The changeover is carried out immediately after starting the FB. It is not waited for the crossing of the master axis changeover position (A-0-2900) by the master axis.

 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.

Library	Area
RIL_CommonTypes	PLCopen data types

Tab. 3-32: Library assignment
Type definition

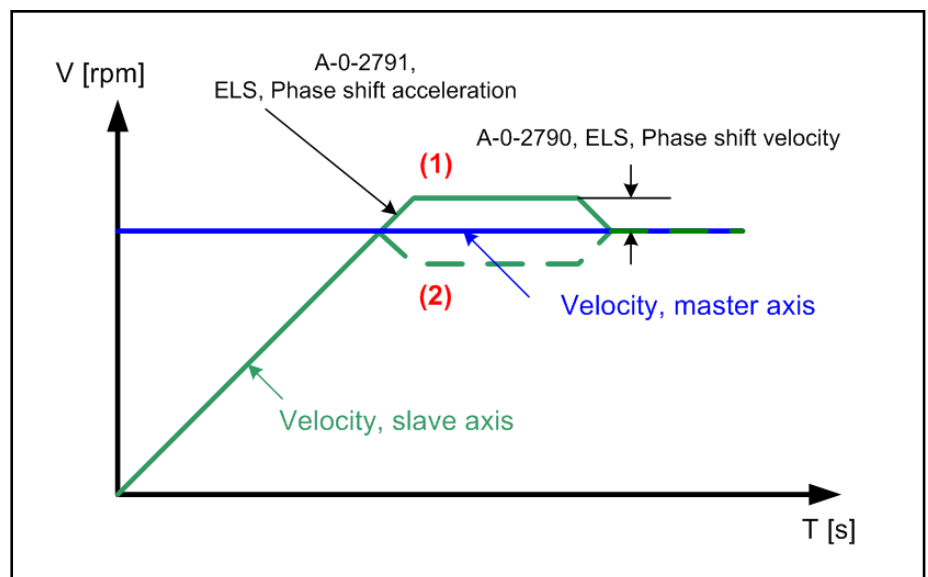
```

TYPE MC_SYNC_DIRECTION :
(
  SHORTEST_WAY      := 0,  (* shortest distance      *)
  CATCH_UP          := 1,  (* 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. Then, 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, the two parameters "A-0-2790, ELS, synchronization velocity" and "A-0-2791, ELS, synchronization acceleration" are active.

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

3.6 Data Types for Fault Diagnosis with IndraMotion 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.

Library	Range
RIL_CommonTypes	Data types of POU diagnosis -> includes

Tab. 3-33: Library assignment

Type Definition

```
(* Enumeration of valid types of diagnostics *)
TYPE ERROR_CODE :
(
  NONE_ERROR           := 16#0000, (* No error code available, FB without errors *)
  INPUT_INVALID_ERROR  := 16#0001, (* Invalid input, error due to invalid input values *)
  COMMUNICATION_ERROR  := 16#0002, (* Error while establishing a communication *)
  RESOURCE_ERROR       := 16#0003, (* Resource error (e.g. parameter) does not exist *)
  ACCESS_ERROR         := 16#0004, (* Access error, resource exists, but cannot be accessed *)
  STATE_MACHINE_ERROR  := 16#0005, (* Invalid state State - Machine *)
  INPUT_RANGE_ERROR    := 16#0006, (* Error because Input out of range *)
  (* Example: Parameter should be written in phase 4 *)
  (* but can only be written in phase 2 *)
  (* The current state of the state machine *)
  (* does not allow a further processing *)
  (* During processing, setpoint values are generated by *)
  (* one or several input values. *)
  (* These generated setpoint values are not defined *)
  (* too high or too low. *)
)
```

RIL_CommonTypes.library

```

CALCULATION_ERROR := 16#0007, (* Error while calculation *)
(* a calculation error occurred, e.g. division by 0 *)
DEVICE_ERROR      := 16#0008, (* Device error of overriding importance occurred, *)
(* e.g. IndraDrive: *)
(* cam operating mode F2005, invalid cam *)
OTHER_ERROR       := 16#7FFE, (* General, undefined Error *)
SYSTEM_ERROR      := 16#7FFF (* system error occurred, e.g. operating system error *)
);
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. Determination of the table number ([ERROR_TABLE page 72](#)). This identifies the origin of the error.
2. Use this finding to evaluate "Additional1". The content is the diagnosis number which can be found in the particular block descriptions (also refer to Troubleshooting Guide, MLC). Check whether "Additional2" = 0; if yes, diagnosis is completed; if no, proceed.
3. Evaluation of "Additional2" (not yet used for MLC_Table).

Library	Range
RIL_CommonTypes	Data types of POU diagnosis

Tab. 3-34: Library assignment

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 *)
(* indicates the list required for *)
(* error identification of "Additional1" *)
  Additional1 : DWORD
(* Additional diagnosis number1 *)
(* does not contain specified error number *)
(* corresponding to the given table. *)
  Additional2 : DWORD
(* Additional diagnosis number2 *)
(* can contain supplements to "Additional1". *)
END_STRUCT
END_TYPE

```



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

3.6.3 ERROR_TABLE

Brief description The **ERROR_TABLE** provides the first element of the **ERROR_STRUCT** [page 72](#). 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 the respective block description get their information from this table.

Blocks can use the table SERCOS_TABLE, 16#0010, if errors occur when accessing a parameter.

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

Library	Range
RIL_CommonTypes	Data types of POU diagnostics - including

Tab. 3-35: Library assignment

Type Definition

```
(* Enumeration of valid tables in additional information *)
TYPE ERROR_TABLE:
(
  NO_TABLE_USED      := 16#0000, (* reserved *)
  SERCOS_TABLE       := 16#0010, (* Sercos Error: Additional = 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 for the Indication of the Date and Absolute Time Designations for IndraMotion Blocks

3.7.1 Indication of the Date and Absolute Time Designations, General

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

- [SYS_TIME64 page 73](#), system time in microseconds since 01.01.1970, 00:00 o'clock.
- [EXT_SYS_TIME_DATE page 74](#), system time in microseconds since 01.01.1970, 00:00 o'clock, additional information year, month, day, hours, minutes, seconds, milliseconds and week day are calculated from the system time.
- [SYS_TIME_DATE page 74](#), year, 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.

Library	Range
RIL_CommonTypes	Data types of time

Tab. 3-36: Library assignment

RIL_CommonTypes.library

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

Library	Range
RIL_CommonTypes	Data types of time

Tab. 3-37: Library assignment

Type definition

```
(* Extended structure of system time *)
TYPE EXT_SYS_TIME_DATE :
STRUCT
  LowValue:  DWORD      (* Low value [us] *)
  HighValue:  DWORD      (* High value [us] *)
  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 system time in microseconds since 01.01.1970, 00:00 o'clock and of the additional information year, month, day, hours, minutes, seconds, milliseconds and week day which are calculated from the system time.

Library	Range
RIL_CommonTypes	Data types of time

Tab. 3-38: Library assignment

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 for the Support of Touch Probe Functions

3.8.1 Data Types for the Support of 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 75, setting of the methods for differential measuring
- [MB_EXPECT_WINDOW_MODE](#), page 76, configuration and activation of the expectation window
- [MB_INTERPOLATION_MODE](#), page 77, activation of the interpolation
- [MB_POSITION_EVAL](#), page 77, selection and activation of the trigger signal
- [MB_PROBE_MODULE](#), page 77, selection of the trigger source for the touch probe
- [MB_PROBE_INPUT](#), page 78, selection of the desired input on the selected module
- [MB_PROBE_DATA](#), page 78, 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 79

3.8.2 MB_DIFF_EVAL

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

Library	Range
RIL_CommonTypes	TouchProbe

Tab. 3-39: Library assignment

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 measurement values, depending on the activated edge.

Example: both edges are activated, the difference pos/neg edge, then neg/pos edge, then pos/neg edge is registered.

RIL_CommonTypes.library

- In the following image, the extended methods 2..5 of difference measurement are displayed:

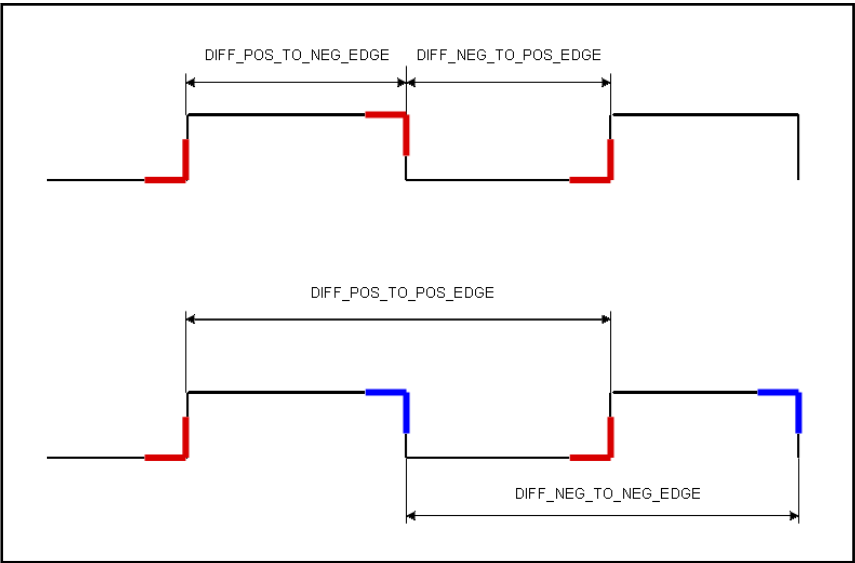


Fig. 3-48: Methods of difference measurement

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.

Library	Range
RIL_CommonTypes	TouchProbe

Tab. 3-40: Library assignment
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. The start position and the 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.

Library	Range
RIL_CommonTypes	TouchProbe

Tab. 3-41: Library assignment

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.

With these enumeration values, it can be selected whether triggering shall be carried out on the positive or the negative edge and/or on both edges.



The default value "Zero" blocks triggering!

Library	Range
RIL_CommonTypes	TouchProbe

Tab. 3-42: Library assignment

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.

Library	Range
RIL_CommonTypes	TouchProbe

Tab. 3-43: Library assignment

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) *)
);

```

RIL_CommonTypes.library

```
);
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 IO modules have the inputs 1..16 as a trigger source.

Library	Range
RIL_CommonTypes	TouchProbe

Tab. 3-44: Library assignment

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 XLC/MLC (A-parameters) and for the touch probes of the drive (S-/P-parameter).



Up to the XLC/MLC 13V04, only the parameter A-0-0100, "Actual position value of the axis, encoder 1" was supported by the touch probes of the control.

Library	Range
RIL_CommonTypes	TouchProbe

Tab. 3-45: Library assignment

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!!!
  TP_ELS_CMD_POS := 10, // P-0-0053 / A-0-0300, Master axis, actual position value
  // ----- / 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.

Library	Range
RIL_CommonTypes	TouchProbe

Tab. 3-46: Library assignment

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 *)
  DifferenceMeasurement :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

```

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 type for position data, bit 7 absolute/modulo
ProbePositionEval	MB_POSITION_EVAL, page 77	Activation positive/negative edge, M-0-0007, control word
Differential measuring	MB_DIFF_EVAL, page 75	Differential measuring, M-0-0007, control word
ExpectWindowPosMode	MB_EXPECT_WINDOW_MODE, page 76	Activation and configuration of the expectation window of the positive edge M-0-0007, control word
ExpectWindowNegMode	MB_EXPECT_WINDOW_MODE, page 76	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

RIL_CommonTypes.library

Structure element	Data type	Parameters
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 ML_Base, Overview

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

- IndraLogic XLC with or without sercos interface and
- IndraMotion MLC

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

The name of the "ML_Base" library is thus extended by the specified reference, that is

"ML_Base_XLC_L45": IndraLogic XLC, control CML45 or

"ML_Base_MLC_L25": IndraMotion MLC, control CML25.



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, page 83](#) defines the parameter length (system-dependent)
 - Remaining parameters: Internal use
- **Flex Structs**
 - [MB_CHANGESTEP_MODE, page 84](#), (ENUM), valid change options within a FlexProfile.
 - [MB_EVENT_ACTION, page 84](#), (ENUM), actions to be triggered by an event.
 - [MB_EVENT_TRIGGER, page 86](#), (ENUM), describes the trigger which should cause a desired action in case of an event.
 - [MB_EXECUTION_MODE, page 86](#), (ENUM), one-time or cyclic processing of the FlexProfile
 - [MB_FLEX_EVENT, page 87](#), (STRUCT), configuration of conditions which can be included in a FlexProfile at a specified step
 - [MB_MASTER_TYPE, page 87](#), (ENUM), dependence of the motion step, current master axis (MASTER_AXIS0) or time (MASTER_TIME).
 - [MB_PROFILE_START_POINT, page 87](#), (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, page 89](#), (STRUCT), configuration of a motion step in a FlexProfile

ML_Base.library

- [MB_STEP_TYPE](#), [page 89](#), (ENUM) characteristic of the motion step, fix (STEP_FIX) step or FlexStep
- [MB_SYNC_TYPE](#), [page 90](#), (ENUM), synchronization direction of the axis and type of switching
- **Read Write parameter types**
 - [MB_SERCOS_DATA_STATUS](#), [page 90](#), (ENUM), status representation of a sercos-parameter access
 - Remaining parameters: Internal use
- **Shared memory types**
 - [ML_AXISDATA_SM](#), [page 118](#), (STRUCT), axis data via shared memory
 - [ML_CONTROLDATA_SM](#), [page 120](#), (STRUCT), control data via shared memory
 - [ML_LINK](#), [page 121](#), validity bit of the control in the MLC link
 - [ML_PLSDATA_SM](#), [page 120](#), (STRUCT), camshaft via shared memory
 - [ML_PROBEDATA_SM](#), [page 121](#), (STRUCT), probe data via shared memory
 - [ML_SYNC](#), [page 122](#) contains the four input bits of the control which are exchanged in the MLC link
 - [SM_EXT_TYPES](#), [page 123](#), (STRUCT) for the type-correct interpretation of formatted data
 - [SM_TYPES](#), [page 123](#), (STRUCT) for the type-correct interpretation of formatted data

Error/diagnostic memory

- [Error/diagnostic memory - supporting user diagnostics and logbook access on page 124](#)

Global Variable List

- [GVL_Base](#) on [page 130](#)
- [GVL_BaseConstant](#) on [page 131](#)
- **Global Variable index**

Declaration files for global constants; C2C control link

- [GVL_A_GlobalVarIdx](#) on [page 132](#)
- **Parameter ID**
 - Declaration files for Global Constants*
 - [GVL_A_Param_ID](#) on [page 135](#)
 - [GVL_C_Param_ID](#) on [page 135](#)
 - [GVL_M_Param_ID](#) on [page 136](#)
 - [GVL_N_Param_ID](#) on [page 136](#)
 - [GVL_O_Param_ID](#) on [page 137](#)
 - [GVL_P_Param_ID](#) on [page 137](#)
 - [GVL_S_Param_ID](#) on [page 138](#)

Interface

- [Interface](#), [page 138](#), (access to sercos parameters)

IP settings

- [IP Configuration](#), [page 144](#), (IP settings),

Project information

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

SystemTime

- [System time, page 152](#)

Target

- [Target system, page 156](#)

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 a system-specific parameter length. These are 64 bits for the parameters of the IndraLogic XLC or IndraMotion MLC. S- and P-parameters are automatically filled with up to 64 bits.

Library	Range
ML_Base	CommonTypes

Tab. 4-1: Library assignment

Program:

```

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

ML_Base.library

4.2.2 Data Types/ Flex Structs

Data Types for the Support of Motion and FlexProfile Functions, General

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

- [MB_CHANGESTEP_MODE](#), [page 84](#), admissible modification possibilities within a FlexProfile,
- [MB_EVENT_ACTION](#), [page 84](#), actions to be triggered by an event,
- [MB_EVENT_TRIGGER](#), [page 86](#), describes the trigger which should cause a desired action to be carried out in case of an event.
- [MB_EXECUTION_MODE](#), [page 86](#), one-time or cyclic processing of the FlexProfile,
- [MB_FLEX_EVENT](#), [page 87](#), configuration of conditions which might be included in a FlexProfile at a predetermined step,
- [MB_MASTER_TYPE](#), [page 87](#), dependence of the motion step, current master axis (MASTER_AXIS0) or time (MASTER_TIME),
- [MB_PROFILE_START_POINT](#), [page 87](#), describes how the origin of the profile has to be interpreted in case of a profile start or change of profile; it can be related to the current position or the zero point for the master as well as the slave axis,
- [MB_PROFILE_STEP](#), [page 89](#), configuration of a motion step in a FlexProfile.
- [MB_STEP_TYPE](#), [page 89](#), characteristic of the motion step, fix (STEP_FIX) step or Flex step.
- [MB_SYNC_TYPE](#), [page 90](#), 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 within a FlexProfile.

Library	Range
ML_Base.library	FlexTypes

Tab. 4-3: Library assignment

Functional Description

Program:

```
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.

Library	Range
ML_Base.library	FlexType

Tab. 4-4: Library assignment

Functional description

Program:

```

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 *)
  ACTION_STATUSBIT1_INSTANT_NEXTSTEP := 16#0301, (* set bit1 immediately and *)
  ACTION_STATUSBIT2_INSTANT_NEXTSTEP := 16#0302, (* set bit2 immediately and *)
  ACTION_STATUSBIT3_INSTANT_NEXTSTEP := 16#0303, (* set bit3 immediately and *)
  ACTION_STATUSBIT4_INSTANT_NEXTSTEP := 16#0304, (* set bit4 immediately and *)
  ACTION_STATUSBIT5_INSTANT_NEXTSTEP := 16#0305, (* set bit5 immediately and *)
  ACTION_STATUSBIT6_INSTANT_NEXTSTEP := 16#0306, (* set bit6 immediately and *)
  ACTION_STATUSBIT7_INSTANT_NEXTSTEP := 16#0307, (* set bit7 immediately and *)

```

ML_Base.library

```
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 The enumeration data type MB_EVENT_TRIGGER describes the trigger which should cause a desired action ([MB_EVENT_ACTION](#), page 84,) to be carried out in case of an event.

Library	Range
ML_Base.library	FlexTypes

Tab. 4-5: Library assignment

Functional Description .
Program:

```
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.

Library	Range
ML_Base.library	FlexType

Tab. 4-6: Library assignment

Functional Description .
Program:

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

```
);
END_TYPE
```

MB_FLEX_EVENT

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

Library	Range
ML_Base.library	FlexType

Tab. 4-7: Library assignment

Functional Description

Program:

```
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_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.

Library	Area
ML_Base.library	FlexTypes

Tab. 4-8: Library assignment

Functional Description

Program:

```
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.

Library	Range
ML_Base.library	FlexType

Tab. 4-9: Library assignment

ML_Base.library

Functional description

Program:

```
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
```

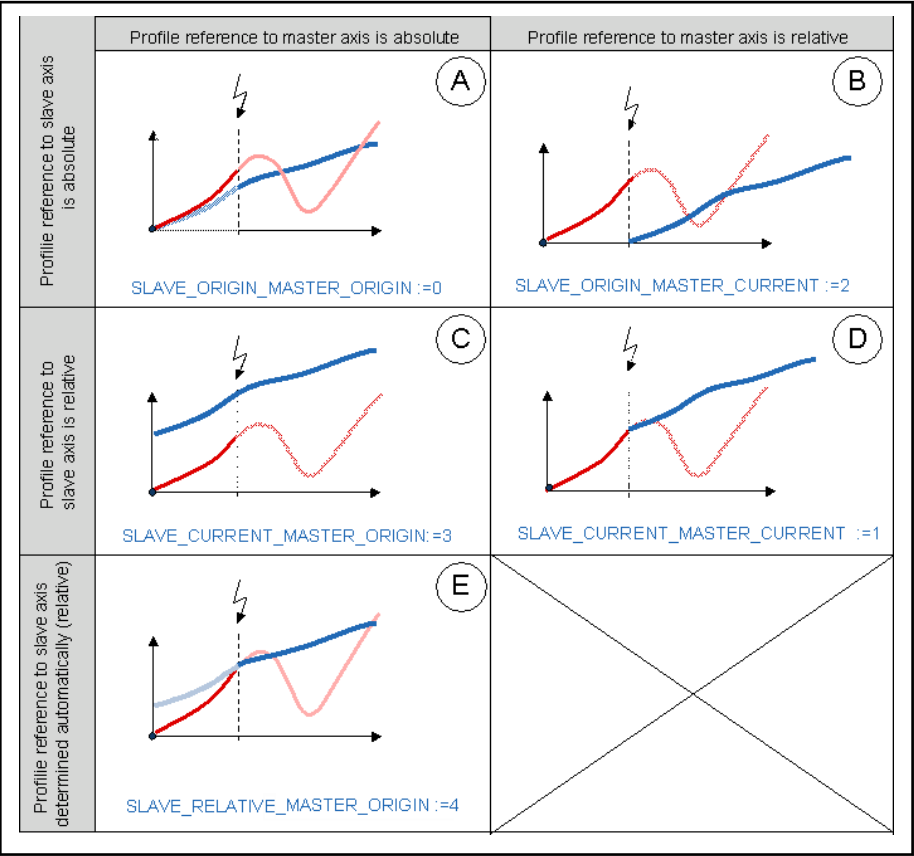


Fig. 4-1: Functional description

- 

Describes the motion profile 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. Then the profile entry is relative.
- 

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



The position and start of the motion profile 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.

Library	Area
ML_Base.library	FlexTypes

Tab. 4-10: Library assignment

Functional description

Program:

```

TYPE MB_PROFILE_STEP :
STRUCT
    Master:MB_MASTER_TYPE := MASTER_AXIS0; (* selection of master axis sources: *)
                                           (* time or position*)
    Hub:LREAL;                          (* stroke *)
    Range:LREAL;                        (* definition of master axis section *)
    MotionLaw:MC_STEP_MODE;              (* definition of motion law *)
    StepType:MB_STEP_TYPE := STEP_FIX;  (* step type: standard (fixed) or Flex *)
    StartVel: LREAL := 0.0;              (* slave axis start velocities *)
    StartAcc: LREAL := 0.0;              (* slave axis start acceleration *)
    StartJrk: LREAL := 0.0;              (* slave axis start jerk *)
    EndVel: LREAL := 0.0;                (* slave axis final velocity *)
    EndAcc: LREAL := 0.0;                (* slave axis final acceleration *)
    EndJrk: LREAL := 0.0;                (* slave axis final jerk *)
    TravelVel:LREAL := 0.0;              (* slave axis traveling velocity,*)
                                           (* not taken into account by all motion *)
                                           (* laws *)
    LimitAcc: LREAL := 0.0;              (* slave axis traveling acceleration, *)
                                           (* not taken into account by all motion *)
                                           (* laws *)
    LimitJrk: LREAL := 0.0;              (* slave axis traveling jerk not taken into *)
                                           (* account by all motion laws*)
    TurningPointDisplacement: LREAL := 0.5; (* turning point displacement for symmetric *)
                                           (* 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.

Library	Area
ML_Base.library	FlexTypes

Tab. 4-11: Library assignment

ML_Base.library

Type definition

```

TYPE MB_STEP_TYPE:
(
STEP_FIX           := 0, (*standard step*)
STEP_FLEX_REL_REL := 1, (*Flex step with relative distance and relative master axis section*)
STEP_FLEX_REL_ABS := 2, (*Flex step with relative distance and absolute master axis section*)
STEP_FLEX_ABS_REL := 3, (*Flex step with absolute slave axis position and relative master axis section*)
STEP_FLEX_ABS_ABS := 4, (*Flex step with absolute slave axis position and absolute 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.

Library	Range
ML_Base.library	FlexTypes

Tab. 4-12: Library assignment

Functional Description

Program:

```

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".

Library	Range
ML_Base.library	RDWR_Parameter

Tab. 4-13: Library assignment

Program:

```

TYPE MB_SERCOS_DATA_STATUS:
(* parameter status*)
(
    MB_SERCOS_DATA_NOT_SET           := 0, (* Not set and not enabled *)
    MB_SERCOS_DATA_SETTING           := 1, (* Set and enabled *)
    MB_SERCOS_DATA_END_OK            := 3, (* Command correctly executed *)
    MB_SERCOS_DATA_INTERRUPTED       := 5, (* Command execution interrupted *)
    MB_SERCOS_DATA_IN_PROCESS        := 7, (* In process *)
    MB_SERCOS_DATA_BREAK_ERROR       := 15, (* Error, command execution impossible*)
    MB_SERCOS_DATA_DATUM_FAULT       := 256 (* Data invalid *)
);
END_TYPE

```



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 Shared Memory between Motion and PLC

- [ML_AXISDATA_SM, page 118](#), (STRUCT), axis data via shared memory.
- [ML_CONTROLDATA_SM, page 120](#), (STRUCT), control data via shared memory.
- [ML_LINK, page 121](#), , validity bit of the control in the MLC link.
- [ML_PLSDATA_SM, page 120](#), (STRUCT), programmable limit switch via shared memory.
- [ML_PROBEDATA_SM, page 121](#), (STRUCT), touch probe data via shared memory.
- [ML_SYNC, page 122](#), contains four input bits of the control which are exchanged in the MLC link.
- [SM_EXT_TYPES, page 123](#), (STRUCT), for the type-correct interpretation of data.
- [SM_TYPES, page 123](#), (STRUCT), for the type-correct interpretation of 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 provi-

ML_Base.library

sion) 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.

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 93](#). The definition is part of the global variable list "GVL_Base" in the "ML_Base_xxx_yyy" 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
```



The "AxisData", "ControlData", "PlsData" and "ProbeData" addresses can be taken from the library manager of ml_Base_xxx_yyy, GVL_Base, documentation.

xxx: mlc | xlc

yyy: l65 | l45 | l25

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_xxx_yyy" library, see [DV_Axis\[\]](#), [page 95](#).

Function variables of the axis data:

```
VAR_GLOBAL
...
DV_Axis: ARRAY [MB_MIN_AXIS_NUMBER..MB_MAX_AXIS_NUMBER] OF ML_DirectVarAxis;
...
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_xxx_yyy" library, see [DV_Axis\[\]](#), [page 98](#).

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

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_xxx_yyy" library.

Definition of the global array "AxisData[]"

```

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

```



The "AxisData", "ControlData", "PlsData" and "ProbeData" addresses can be taken from the library manager of ml_Base_xxx_yyy, GVL_Base, documentation.

xxx: mlc | xlc

yyy: l65 | l45 | l25

Also refer to

- [ML_AXISDATA_SM, page 118](#)

ML_Base.library

**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.

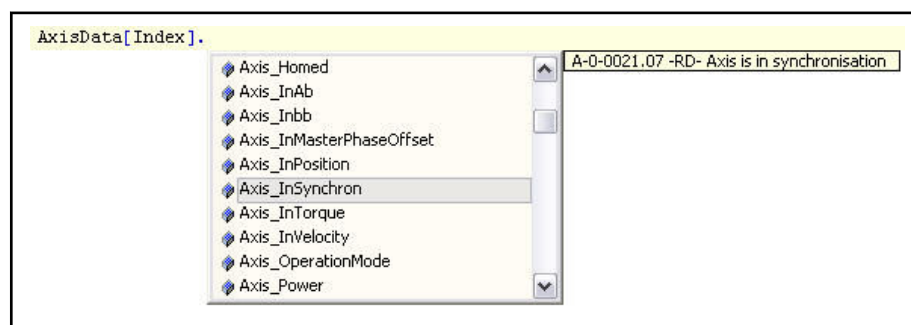


Fig. 4-2: IntelliSense of a variable in the shared memory



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 $\neq 0$, 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.

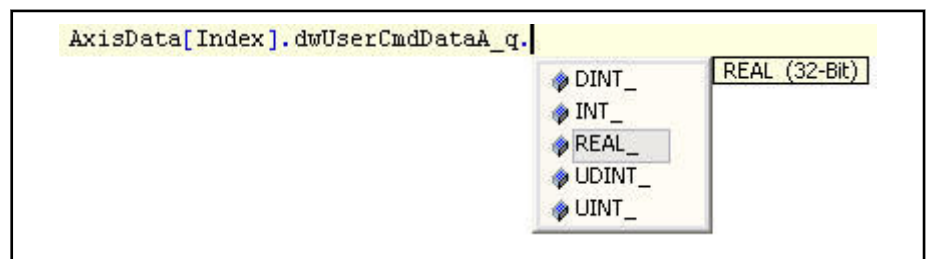


Fig. 4-3: Data type specification via point operator

Also refer to

- [SM_Types](#), page 123,
- [ML_AXISDATA_SM](#), page 118

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 131).

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[]")

ML_Base.library



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.

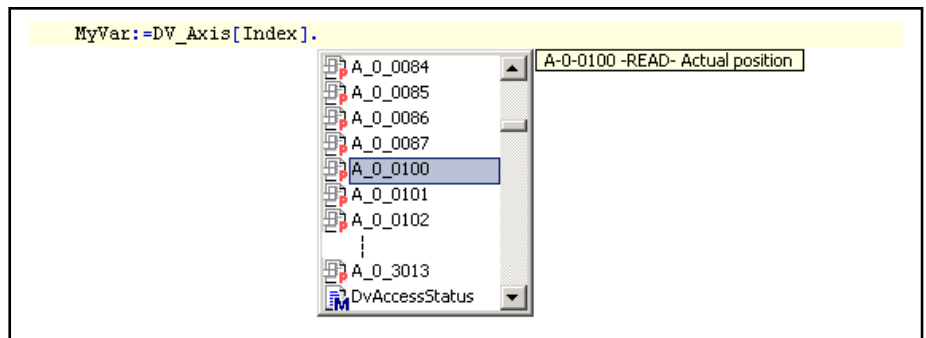


Fig. 4-4: Function variable access using the IntelliSense functionality



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 135](#)
- [GVL_C_Param_ID, page 135](#)
- [GVL_M_Param_ID, page 136](#)
- [GVL_N_Param_ID, page 136](#)
- [GVL_O_Param_ID, page 137](#)

Status (extract, for further information, see SERCOS_TABLE)

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

ML_Base.library

- 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_xxx_yyy" 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.

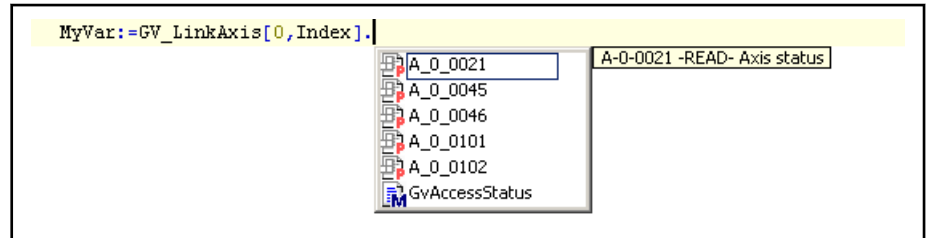


Fig. 4-5: Function variable access synchronous in the C2C link using the IntelliSense functionality



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 132.

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_Base.library

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 date 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 date can be modified due to task changes.

Shared memory

Control data in shared memory is declared as global array "ControlData[]" of the data structure [ML_CONTROLDATA_SM](#), [page 101](#).. The definition is part of the global variable list "GVL Base" in the "ML Base xxx yyy" 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
```



The "AxisData", "ControlData", "PlsData" and "ProbeData" addresses can be taken from the library manager of ml_Base_xxx_yyy, GVL_Base, documentation.

xxx: mlc | xlc

yyy: 165 | 145 | 125

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_xxx_yyy" library, see [DV_Control\[\], page 103](#).

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).

ControlData[] - Control Data in Shared Memory

Brief description

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 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_xxx_yyy" 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
```

ML_Base.library



The "AxisData", "ControlData", "PlsData" and "ProbeData" addresses can be taken from the library manager of ml_Base_xxx_yyy, GVL_Base, documentation.

xxx: mlc | xlc

yyy: l65 | l45 | l25

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.

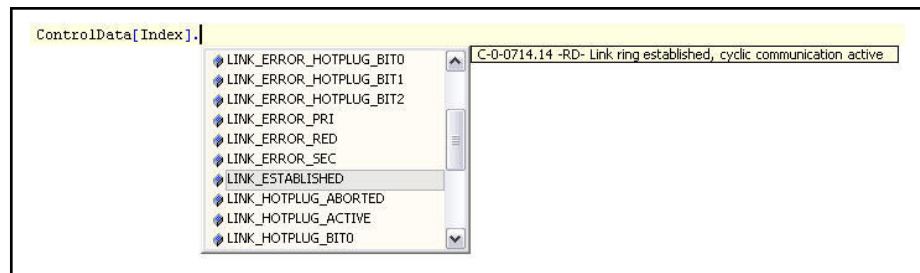


Fig. 4-6: IntelliSense of a variable in the shared memory



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 120

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 131](#)) control data 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_xxx_yyy" 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.

Definition of 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;
```

ML_Base.library

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

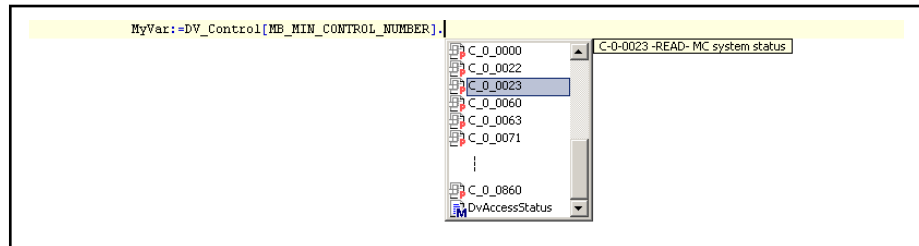


Fig. 4-7: Function variable access using the IntelliSense functionality



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 135
- [GVL_C_Param_ID](#), page 135
- [GVL_M_Param_ID](#), page 136
- [GVL_N_Param_ID](#), page 136
- [GVL_O_Param_ID](#), page 137

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 data 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 data can be modified due to task changes.

Shared memory

Programmable limit switch data in shared memory is declared as global array "PLSData[]" of the data structure [ML_PLSDATA_SM](#), [page 106](#),. The definition is part of the global variable list "GVL_Base" in the "ML_Base_xxx_yyy" 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
```



The "AxisData", "ControlData", "PlsData" and "ProbeData" addresses can be taken from the library manager of ml_Base_xxx_yyy, GVL_Base, documentation.

xxx: mlc | xlc

yyy: l65 | l45 | l25

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_xxx_yyy" library, also refer to [DV_Pls\[\], page 108](#).

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_xxx_yyy" 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

```



The "AxisData", "ControlData", "PlsData" and "ProbeData" addresses can be taken from the library manager of ml_Base_xxx_yyy, GVL_Base, documentation.

xxx: mlc | xlc

yyy: l65 | l45 | l25

Also refer to

- [ML_PLSDATA_SM, page 120](#)

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.

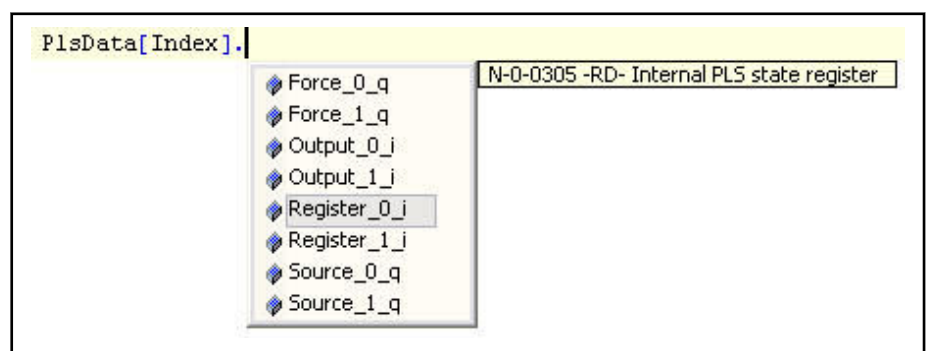


Fig. 4-8: IntelliSense of a variable in the shared memory



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];

```

ML_Base.library

END_VAR

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

See also:

- [ML_PLSDATA_SM, page 120](#)

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 131](#)), 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 the 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_xxx_yyy" 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.

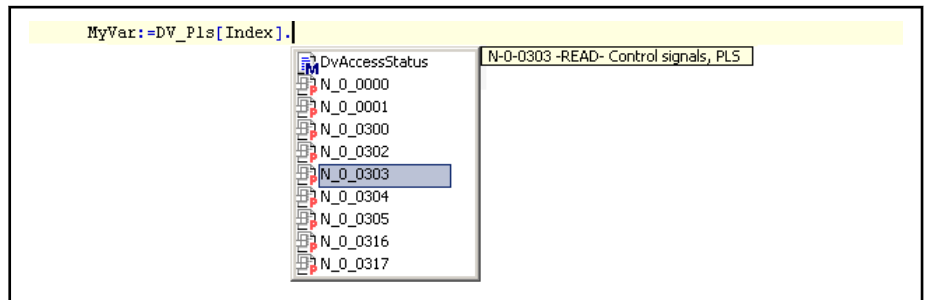


Fig. 4-9: Function variable access using the IntelliSense functionality



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_Osci[Index].DvAccessStatus(FP_O_0_0021);
```

Declaration files for global constants

- [GVL_A_Param_ID, page 135](#)
- [GVL_C_Param_ID, page 135](#)
- [GVL_M_Param_ID, page 136](#)
- [GVL_N_Param_ID, page 136](#)

ML_Base.library

- [GVL_O_Param_ID, page 137](#)

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.

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. 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 data 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 data can be modified due to task changes.

Shared memory Touch probe data in shared memory is declared as global array "ProbeData[]" of the data structure [ML_PROBEDATA_SM, page 111](#). The definition is part of the global variable list "GVL_Base" in the "ML_Base_xxx_yyy" library. The touch probes are numbered and the name of the touch probe is not important for this access type.²⁾

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
```



The "AxisData", "ControlData", "PlsData" and "ProbeData" addresses can be taken from the library manager of ml_Base_xxx_yyy, GVL_Base, documentation.

xxx: mlc | xlc

yyy: l65 | l45 | l25

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_xxx_yyy" library, also refer to [DV_Probe\[\], page 113..](#)

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[]")

.

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

ML_Base.library



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 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_xxx_yyy" library.

Defining the global array "ProbeData[]"

```
VAR_GLOBAL
```

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



The "AxisData", "ControlData", "PlsData" and "ProbeData" addresses can be taken from the library manager of ml_Base_xxx_yyy, GVL_Base, documentation.

xxx: mlc | xlc

yyy: l65 | l45 | l25

See also:

- [ML_PROBEDATA_SM, page 121](#)

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.

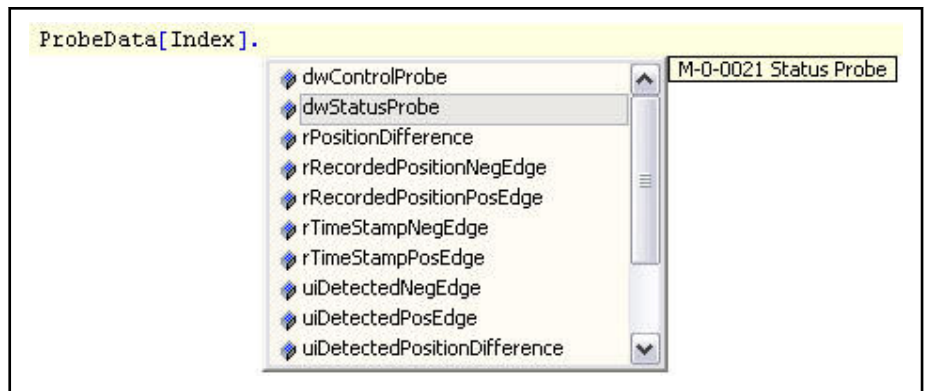


Fig. 4-10: IntelliSense of a variable in the shared memory



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.

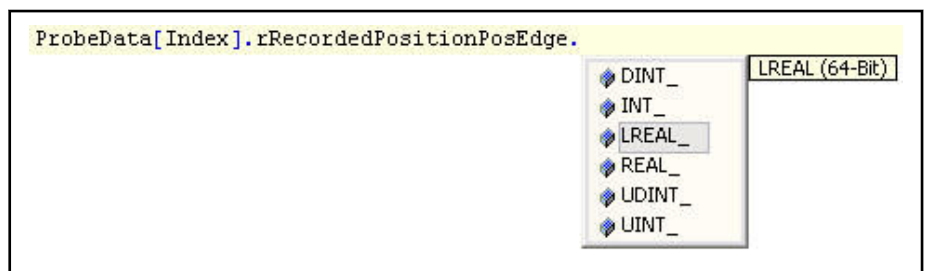


Fig. 4-11: Data type specification via point operator

Also refer to

- [SM_EXT_TYPES](#), page 123,
- [ML_PROBEDATA_SM](#), page 121

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

ML_Base.library

([GVL_BaseConstant](#), page 131), the touch probe data can be accessed 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 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_xxx_yyy" library.

The function block "ML_DirectVarProbe" 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 direct variable depends on the 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.

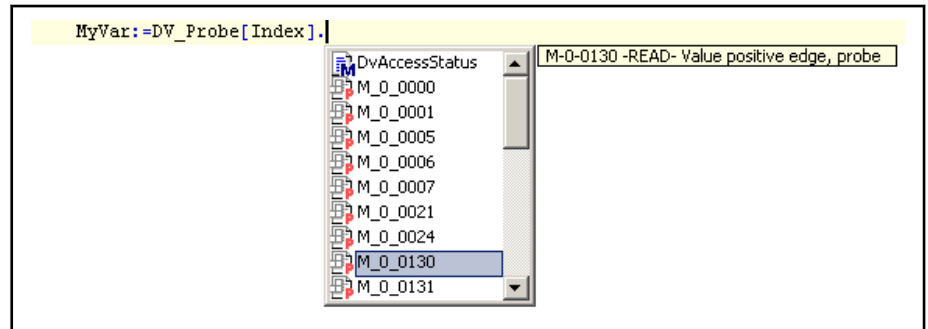


Fig. 4-12: Function variable access using the IntelliSense functionality



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 135](#)
- [GVL_C_Param_ID, page 135](#)
- [GVL_M_Param_ID, page 136](#)
- [GVL_N_Param_ID, page 136](#)
- [GVL_O_Param_ID, page 137](#)

Status (extract, for further information, see SERCOS_TABLE)

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

ML_Base.library

- 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_xxx_yyy" library, also refer to [DV_Osc](#), page 116.

Function variables of oscilloscope data

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



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 131), 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.

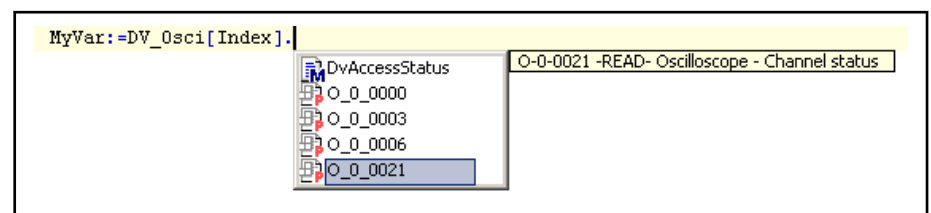


Fig. 4-13: Function variable access using the IntelliSense functionality



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];
```

ML_Base.library

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_Osci[Index].DvAccessStatus(FP_O_0_0021);
```

Declaration files for global constants

- [GVL_A_Param_ID](#), page 135
- [GVL_C_Param_ID](#), page 135
- [GVL_M_Param_ID](#), page 136
- [GVL_N_Param_ID](#), page 136
- [GVL_O_Param_ID](#), page 137

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**.

Library	Range
ML_Base	Data types/shared memory

Tab. 4-14: Library assignment

Data structure design 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 velocity
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 position
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_PLCCControlled :BIT; // A-0-0021.27 -RD- Interpolation influenced by PLC

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

```

ML_Base.library

```
END_STRUCT
END_TYPE
```

See also:

- [AxisData\[\] - Axis Data in Shared Memory, page 93](#)
- [SM_TYPES, page 123.](#)

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**.

Library	Range
ML_Base	Data types/shared memory

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

```

See also:

- [ML_LINK, page 121,](#)
- [ML_SERCOS, page 122](#)
- [ML_SYNC, page 122,](#)

ML_PLSDATA_SM

Short Description

An instance of the structure ML_PLSDATA_SM contains the information of the corresponding camshaft gear which can be accessed via **Shared Memory**.

Library	Range
ML_Base	Data Types/ Shared Memory

Tab. 4-16: Library assignment

Data Type Declaration 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

Short Description An instance of the structure ML_PROBEDATA_SM contains the information of the corresponding touch probe which can be accessed via **Shared Memory**.

Library	Range
ML_Base	Data Types/ Shared Memory

Tab. 4-17: Library assignment

Data Type Declaration 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 123.

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.

ML_Base.library

Library	Range
ML_Base	Shared memory 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.

Library	Range
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.

Library	Range
ML_Base	Shared memory 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

Short 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 completed separately by a ".".

Library	Range
ML_Base	Data Types/ Shared Memory 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.

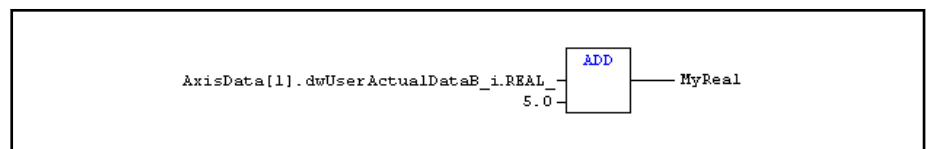


Fig. 4-14: Use of the cyclic SERCOS data channel for reading

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 ".".

Library	Range
ML_Base	Data Types/ Shared Memory Types

Tab. 4-22: Library assignment

Data 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

```

ML_Base.library

Example:

M-0-0130, the measurement value of the positive edge, touch probe, is a date of the length of 4 byte/float.

`ProbeData[3].rValuePositiveEdgeProbe.REAL_`

means that the measurement 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

Short description The ML_ReadDiagLog function block reads entries from the diagnostic logbook.

Library	Range
ML_Base_xxx_yyyy	Error/diagnostic memory

Tab. 4-23: Library assignment

Interface description

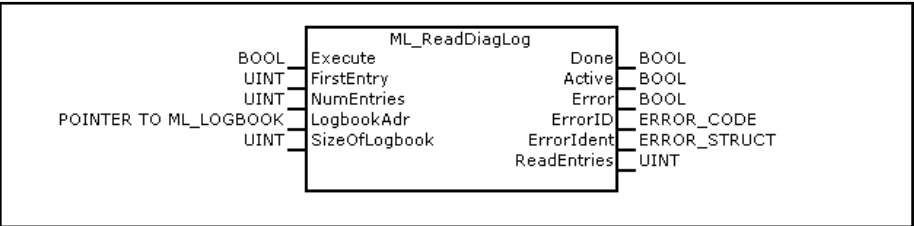


Fig. 4-15: ML_ReadDiagLog function block

	Name	Type	Comment
VAR_INPUT	Execute	UDINT	Processing release at a rising edge
	FirstEntry	UINT	Start index from which the diagnostic logbook is to be saved
	NumEntries	UINT	Maximum number of entries to be read
	LogbookAdr	POINTER TO ML_LOGBOOK	Address on which the diagnostic logbook entries are to be saved See chapter 4.3.7 "ML_LOGBOOK" on page 129
	SizeOfLogbook	UINT	Size of data range in bytes created in LogbookAdr
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 Also refer to "Error handling" on page 125
	ReadEntries	UINT	Number of read logbook entries

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"
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 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.

At the "LogbookAdr" input, the array address of a the ML_LOGBOOK data type has to be assigned. The addresses can be determined with the ADR() operator. The array size in byte is assigned at the "SizeOfLogbook" input. This size can be determined with the SIZEOF() operator.

In case of 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 at the array transferred at the "LogbookAdr" input are valid. "ReadEntries" contains the number of the logbook entries transmitted to the array.

If an error occurs while processing, the "Error" output is set. "ErrorID" and "ErrorIdent" contain detailed information on the error.

Error handling

The ML_ReadDiagLog returns the following error codes of F_RELATED_TABLE, ERROR_TABLE, table number 16#0170:

ErrorID	Additional1	Additional2	Description
INPUT_INVALID_ERROR	16#00002800	16#00000000	Input LogbookAdr is 0
INPUT_INVALID_ERROR	16#00002801	16#00000000	Input SizeOfLogbook is 0
INPUT_INVALID_ERROR	16#00002801	16#00000001	SizeOfLogbook is too small for the required number of logbook entries
INPUT_INVALID_ERROR	16#00002801	16#00000002	SizeOfLogbook is not a multiple of the ML_LOGBOOK structure
INPUT_INVALID_ERROR	16#00002802	16#00000000	Input FirstEntry is larger than 999
INPUT_INVALID_ERROR	16#00002803	16#00000000	Input NumEntries is larger than 1000
INPUT_INVALID_ERROR	16#00002803	16#00000001	Input NumEntries is smaller 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 appears on the display of the control.

ML_Base.library

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, as an entry is made into the logbook with every call and overruns it!

Library	Range
ML_Base_xxx_yyyy	Error/diagnostic memory

Tab. 4-27: Library assignment

Interface description

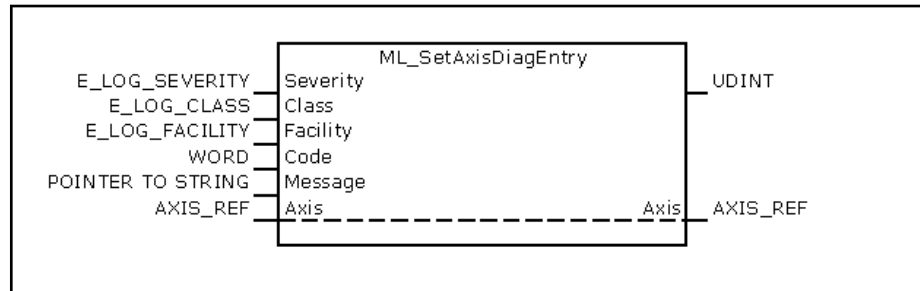


Fig. 4-16: ML_SetAxisDiagEntry function

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the desired control and the axis controlled by it.
VAR_INPUT	Severity	E_LOG_SEVERITY	Entry type: MESSAGE: 16#A, WARNING: 16#E, ERROR: 16#F
	Class	E_LOG_CLASS	Class 0..9, determines the error reaction of the control
	Facility	E_LOG_FACILITY	Facility for user-defined entries permissible: 16#003B0000
	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

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 appears on the display of the control.

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
 - A logbook entry is generated, a 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 appears on the display of the control.



The function may **not** be called cyclically in the PLC program, as an entry is made into the logbook with every call and overruns it!

Library	Range
ML_Base_xxx_yyyy	Error/diagnostic memory

Tab. 4-29: Library assignment

Interface description

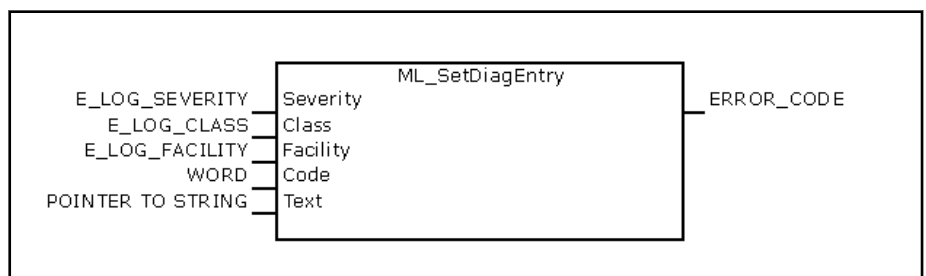


Fig. 4-17: ML_SetDiagEntry function

	Name	Type	Comment
VAR_INPUT	Severity	E_LOG_SEVERITY	Entry type: MESSAGE: 16#A, WARNING: 16#E, ERROR: 16#F
	Class	E_LOG_CLASS	Class 0..9, determines the error reaction of the control
	Facility	E_LOG_FACILITY	Facility for user-defined entries permissible: 16#003B0000
	Code	WORD	User code 16#0000 16#FFFF

ML_Base.library

	Name	Type	Comment
	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 appears on the display of the control.

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.

Meaningful 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
 - A logbook entry is generated, a message is shown on the display.
 - **All** axes are disengaged as best as 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 IndraMotion MLC as well as of the IndraLogic XLC.

Library	Range
ML_Base_xxx_yyy	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_FACILITY describes the diagnostic groups of the diagnostic system of the IndraMotion MLC as well as of the IndraLogic XLC enabled for user diagnostics.

Library	Range
ML_Base_xxx_yyy	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 IndraMotion MLC as well as of the IndraLogic XLC enabled for user diagnostics.

Library	Range
ML_Base_xxx_yyy	Error/diagnostic memory

Tab. 4-33: Library assignment

E_LOG_SEVERITY (ENUM)

```

TYPE E_LOG_FACILITY:
(
    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. The structure is used by the ML_ReadDiagLog function block.

Library	Range
ML_Base_xxx_yyyy	Data types/Info

Tab. 4-34: Library assignment

Structure element	Type	Description
ErrorNumber	DWORD	Error number
State	UINT	Diagnostic input status
LogTime	SYS_TIME_DATE	Diagnostic point in time
DiagText	STRING(123)	Diagnostic text

Tab. 4-35: ML_LOGBOOK data structure

ML_Base.library

4.4 Global Variable List

4.4.1 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 in **GVL_Base** are listed in the global list [GVL_BaseConstant](#), page 131, .

Library	Range
ML_Base	Global variables lists

Tab. 4-36: 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
```



The "AxisData", "ControlData", "PlsData" and "ProbeData" addresses can be taken from the library manager of ml_Base_xxx_yyy, GVL_Base, documentation.

xxx: mlc | xlc

yyy: l65 | l45 | l25

NoAxis	Default initial value
NoTouchProbe	Default initial value
GV_LinkAxis	GV_LinkAxis[] - Axis Data via Synchronous Function Variable Access in C2C Link , page 98
GlobalVarDirectVarData	Internal use
Axis Data via Shared Memory and Function Variable Access, page 91	
AxisData	AxisData[] - Axis Data in Shared Memory, page 93
DV_Axis	DV_Axis[] - Axis Data via Function Variable Access, page 95
Control Data via Shared Memory and Function Variable Access, page 100	
ControlData	ControlData[] - Control Data in Shared Memory, page 101
DV_Control	ControlData[] - Control Data in Shared Memory, page 101

Programmable Limit Switch Data via Shared Memory and Function Variable Access, page 105	
PlsData	PlsData[] - PLS Data in Shared Memory, page 106
DV_Pls	DV_Pls[] - PLS Data via Function Variable Access, page 108
Touch Probe Data via Shared Memory and Function Variable Access, page 110	
ProbeData	ProbeData[] - Touch Probe Data in Shared Memory , page 111
DV_Probe	DV_Probe[] - Touch Probe Data via Function Variable Access, page 113
Oscilloscope Data via Function Variable Access, page 116	
DV_Osci	DV_Osci[] - Oscilloscope Data via Function Variable Access, page 116

Tab. 4-37: Components of the "GVL_Base"

4.4.2 GVL_BaseConstant

Brief description	The global list GVL_BaseConstant contains the essential global constants of an • IndraLogic XLC with or without sercos interface and • IndraMotion MLC.

Library	Range
ML_Base_xxx_yyy ³⁾	Global variables lists

Tab. 4-38: Library assignment

3) xxx: XLC or MLC; yyy: L25, L45, L65 or Vxx

ML_Base.library

VAR_GLOBAL CONSTANT	Type	Comment	XLC			MLC		
			L25	L45	L65	L25	L45	L65
IL_TARGET_TYPE	UINT	XLC // MLC	1	1	1	1	1	1
IL_TARGET_VERSION	UINT	XLC 13 VRS // MLC 13VRS	13	13	13	13	13	13
IL_SYSTEM_MLC // IL_SYSTEM_XLC	UINT	System XLC // System MLC	1	1	1	1	1	1
MB_MIN_OSCI_NUMBER	INT	min. number of oscilloscopes	1	1	1	1	1	1
MB_MAX_OSCI_NUMBER	INT	max. number of oscilloscopes	16	16	16	16	16	16
MB_NUMBER_OF_OSCI	INT		16	16	16	16	16	16
MB_MIN_PROBE_NUMBER	INT	min. number of touch probes	1	1	1	1	1	1
MB_MAX_PROBE_NUMBER	INT	max. number of touch probes	100	100	100	100	100	100
MB_NUMBER_OF_PROBE	INT		100	100	100	100	100	100
MB_MIN_PLS_NUMBER	INT	min. number of PLS FMs	1	1	1	1	1	1
MB_MAX_PLS_NUMBER	INT	max. number of PLS FMs	2	4	4	2	4	4
MB_NUMBER_OF_PLS	INT		2	4	4	2	4	4
MB_MIN_CONTROL_NUMBER	INT	min. number of controls	0	0	0	0	0	0
MB_MAX_CONTROL_NUMBER	INT	max. number of controls	0	0	0	0	0	0
MB_NUMBER_OF_CONTROL	INT		1	1	1	1	1	1
MB_MIN_AXIS_NUMBER	INT	min. number of axis	1	1	1	1	1	1
MB_MAX_AXIS_NUMBER	INT	max. number of axis	16	32	64	16	32	64
MB_NUMBER_OF_AXIS	INT		16	32	64	16	32	64
MB_MIN_GLOBAL_CONTROL_NUMBER	INT	min. number of global controls	0	0	0	0	0	0
MB_MAX_GLOBAL_CONTROL_NUMBER	INT	max. number of global controls	0	0	0	64	64	64
MB_NUMBER_OF_GLOBAL_CONTROL	INT		0	0	0	65	65	65
MB_MIN_GLOBAL_AXIS_NUMBER	INT	min. number of global axis	0	0	0	1	1	1
MB_MAX_GLOBAL_AXIS_NUMBER	INT	max. number of global axis	0	0	0	64	64	64
MB_NUMBER_OF_GLOBAL_AXIS	INT		0	0	0	64	64	64
ML_MAX_NUMBER_OF_APPLICATIONS	UINT	max. number of plc applications	1	1	1	1	1	1
ML_MIN_IDX_ARRAY_EXT_ATTRIBUTE	UDINT		0	0	0	0	0	0
ML_MAX_IDX_ARRAY_EXT_ATTRIBUTE	UDINT		3	3	3	3	3	3
ML_LOG_ID_PLC_BASE	UDINT		1	1	1	1	1	1
ML_LOG_ID_PLC_POU	UDINT		2	2	2	2	2	2
ML_LOCAL_SERCOS_IFC ¹⁾	INT	onboard sercos interface	0	0	0	0	0	0
ML_MIN_SERCOS_IFC ¹⁾	INT	min. number of sercos interfaces	0	0	0	0	0	0
ML_MAX_SERCOS_IFC ¹⁾	INT	max. number of sercos interfaces	0	0	0	0	0	0
ML_NUMBER_OF_SERCOS_IFC ¹⁾	INT	one sercos interface available	1	1	1	1	1	1
END_VAR								

¹⁾ Under development

Fig. 4-18: Declaration of the global list "GVL_BaseConstant", initial values of the systems

4.4.3 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.

Library	Range
ML_Base	Global Variable Lists/Global Variable Index

Tab. 4-39: 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

```

4.4.4 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 (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.

Declaration files for Global Constants

- [GVL_A_Param_ID, page 135,](#)
- [GVL_C_Param_ID, page 135,](#)
- [GVL_M_Param_ID, page 136,](#)

ML_Base.library

- [GVL_N_Param_ID](#), page 136,
- [GVL_O_Param_ID](#), page 137,
- [GVL_P_Param_ID](#), page 137,
- [GVL_S_Param_ID](#), page 138

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. 4-40: Parameter offsets

Several axes can be actuated with the control. That is why the blocks have 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.
 - ContrlNo:= LOCAL_CONTRL or CNTRL_1...

– AxisNo:= NO_OBJECT.

References at axis input, example: MB_ReadParameter

```

VAL:AXIS_REF:=(CntrlNo:=LOCAL_CNTRL,AxisNo:=AXIS_1)
                (*A parameters, axis VAL, 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

Short description The global constant list contains the address constant assignment to the respective parameter.

The parameter constant name is composed as follows

A-0-0001 ⇒ FP_A_0_0001 etc. ...

Number and parameter constant name can be used optionally.

Library	Range
ML_Base	Global variables lists/ Parameter ID

Tab. 4-41: 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 83.](#)

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.

Library	Range
ML_Base	Global Variable Lists/parameter ID

Tab. 4-42: Library assignment

Declaring the global constant list GVL_C_Param_ID

```

VAR GLOBAL CONSTANT
  FP_C_0_0000: MB_IDN:= 8589934592 ; (* Dummy C parameter *)

```

ML_Base.library

```
FP_C_0_0001: MB_IDN:= 8589934593 ; (* Language selection *)
FP_C_0_0002: MB_IDN:= 8589934594 ; (* Name of parameter file *)
(* ..... *)
END_VAR
```

See also:

- [MB_IDN, page 83](#)

GVL_M_Param_ID

Short description The global constant list contains the address constant assignment to the respective parameter.
The parameter constant name is composed as follows
M-0-0001 ⇒ FP_M_0_0001 etc. ...
Number and parameter constant name can be used optionally.

Library	Range
ML_Base	Global variables lists/ Parameter ID

Tab. 4-43: 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 83.](#)

GVL_N_Param_ID

Short description The global constant list contains the address constant assignment to the respective parameter.
The parameter constant name is composed as follows
N-0-0001 ⇒ FP_N_0_0001 etc. ...
Number and parameter constant name can be used optionally.

Library	Range
ML_Base	Global variables lists/ Parameter ID

Tab. 4-44: 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 83.](#)

GVL_O_Param_ID

Short description The global constant list contains the address constant assignment to the respective parameter.
The parameter constant name is composed as follows
O-0-0001 ⇒ FP_O_0_0001 etc. ...
Number and parameter constant name can be used optionally.

Library	Range
ML_Base	Global variables lists/ Parameter ID

Tab. 4-45: 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 83.](#)

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.

Library	Range
ML_Base	Global Variable Lists/parameter ID

Tab. 4-46: 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 83](#)

ML_Base.library

GVL_S_Param_ID

Short description The global constant list contains the address constant assignment to the respective parameter.

 The parameter constant name is composed as follows

 S-0-0001 ⇒ FP_S_0_0001 etc. ...

 Number and parameter constant name can be used optionally.

Library	Range
ML_Base	Global variables lists/ Parameter ID

Tab. 4-47: 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 83.](#)

4.5 Interfaces, functions and function blocks

4.5.1 IL_SercosAttribute

Brief Description The function block IL_SercosAttribute serves for the breakdown of the elements of the Sercos attribute.

Interface Description

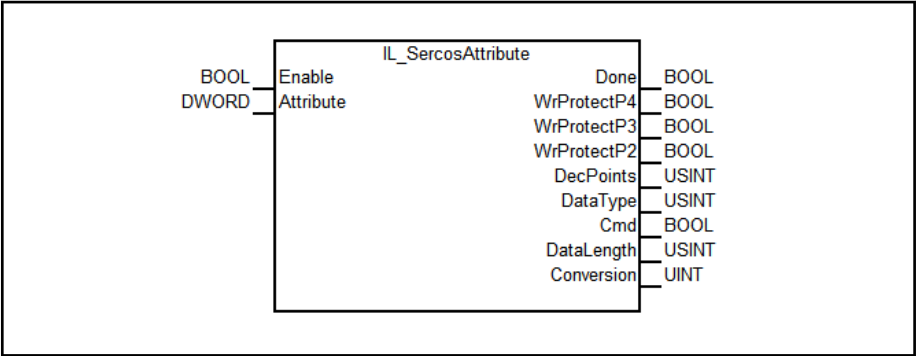


Fig. 4-19: Function block IL_SercosAttribute

	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

	Name	Data type	Description
	WrProtectP3	BOOL	Operation date of the parameter in Sercos phase 3 (P3) write-protected
	WrProtectP2	BOOL	Operation date of the parameter in the parameterization mode (Sercos phase 2 - P2) write-protected
	DecPoints	USINT	Number of decimal places of the operation date of the parameter
	DataType	USINT	Data type and display format of the operation date of the parameter
	Cmd	BOOL	Operation date calls for command function
	DataLength	USINT	Data length of the operation date of the parameter
	Conversion	UINT	Evaluation factor of the operation date of the parameter

Tab. 4-48: Interface IL_SercosAttribute

Sercos attribute The following table shows the structure of the Sercos attribute.

Bit	Interface variable	Description
30:	WrProtectP4	Write-protection of the operation date of the parameter in the operation mode (Sercos phase 4)
		0: Operation date can be written 1: Operation date is write-protected
29:	WrProtectP3	Write-protection of the operation date of the parameter (Sercos phase 3)
		0: Operation date can be written 1: Operation date is write-protected
28:	WrProtectP2	Write-protection of the operation date of the parameter in the parameterization mode (Sercos phase 2)
		0: Operation date can be written 1: Operation date is write-protected
27..24:	DecPoints	The decimal places define the position of the comma for the display and input of the respective operation date.
		0000: No decimal places 1111: 15 decimal places (maximum)
23:		Reserved

ML_Base.library

Bit	Interface variable	Description	
22..20:	DataType	Data type and display format are used to convert the operation 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 byte)
19:	Cmd	The function of the operation date indicates that a command function is called for in the drive with this operation date.	
		0: Operation 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: Operation date is 2 byte long	
		010: Operation 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. 4-49: Elements of the Sercos attribute

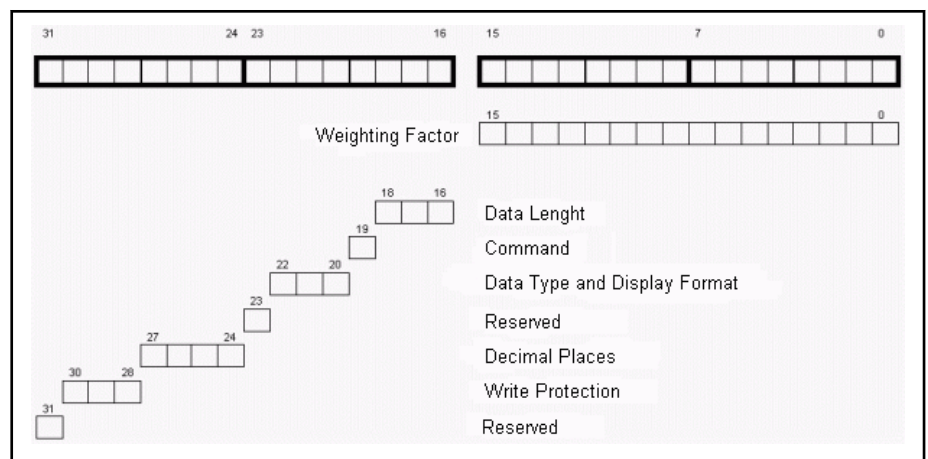


Fig. 4-20: Structure of the Sercos attribute

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 signalled by the output "Done".

Error Handling For this function block, no error treatment is available.

4.5.2 IL_SercosElementsToParaNo

Brief Description The function IL_SercosElementsToParaNo serves for the combination of the individual elements of a SERCOS parameter string and the simultaneous conversion to the SERCOS parameter ident number.

Interface description

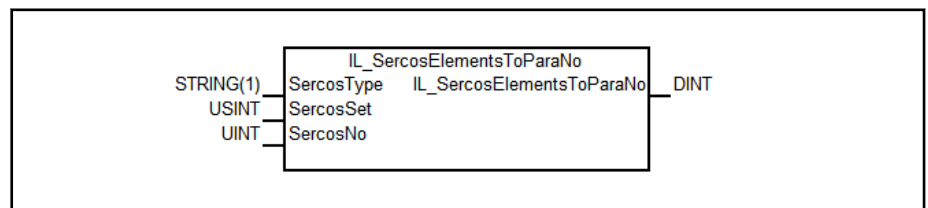


Fig. 4-21: Function IL_SercosElementsToParaNo

	Name	Data type	Description
VAR_INPUT	SercosType	STRING[1]	Parameter type
	SercosSet	USINT	Parameter set
	SercosNo	UINT	Parameter number
RETURN	IL_SercosElement-sToParaNo	UDINT	Parameter ident number

Tab. 4-50: 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.

ML_Base.library

Error handling

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".

Application Example

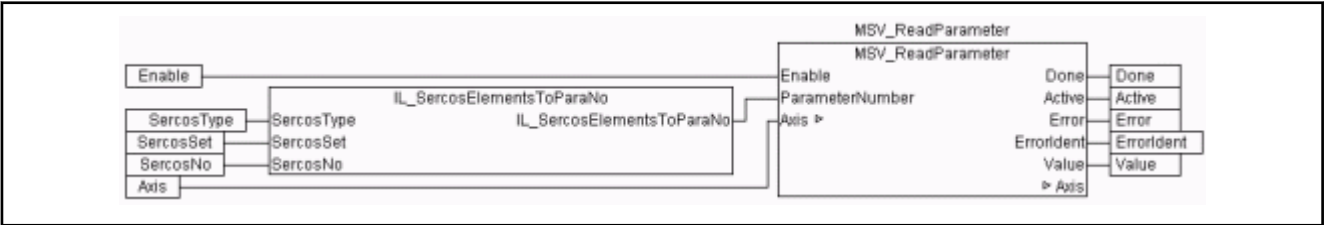


Fig. 4-22: Application of the function (CFC): IL_SercosElementsToParaNo

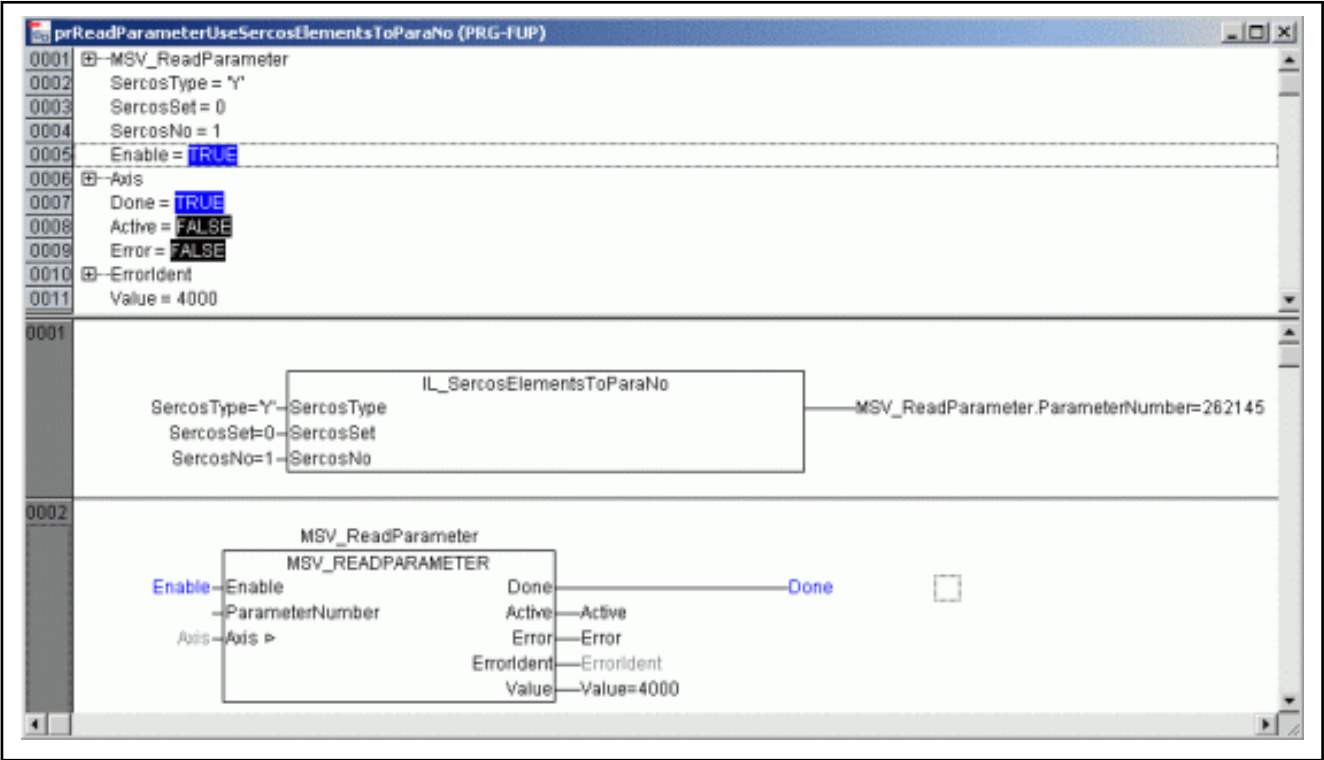


Fig. 4-23: Application of the function (FBD): IL_SercosElementsToParaNo

4.5.3 IL_SercosStringToParaNo

Brief Description

The function IL_SercosStringToParaNo serves for the conversion of a SERCOS parameter string to the SERCOS parameter ident number.

Interface description

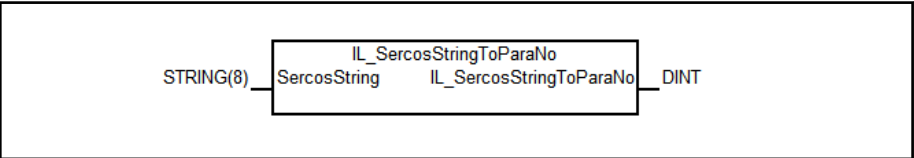


Fig. 4-24: Function IL_SercosStringToParaNo

	Name	Data type	Description
VAR_INPUT	SercosString	STRING[8]	SERCOS-parameter string
RETURN	IL_SercosStringToParaNo	UDINT	Parameter ident number

Tab. 4-51: Interface IL_SercosStringToParaNo

Specification The function IL_SercosStringToParaNo allows for the conversion of 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 puts the connected input elements together to get the corresponding SERCOS parameter ident number.

Error handling 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".

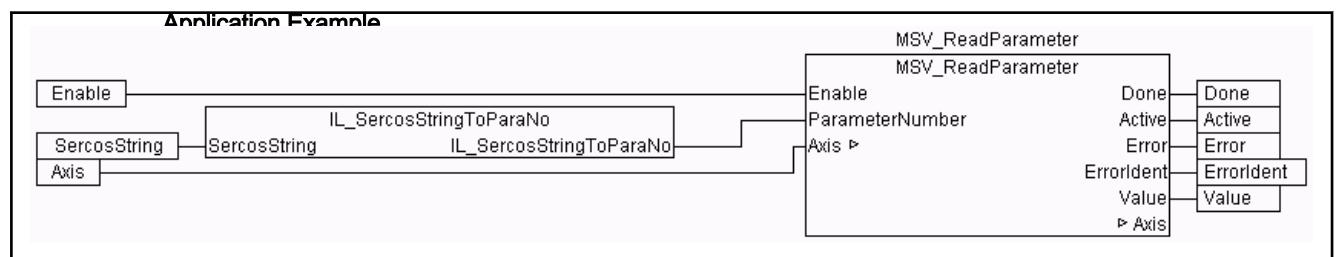


Fig. 4-25: Application of the function (CFC): IL_SercosStringToParaNo

ML_Base.library

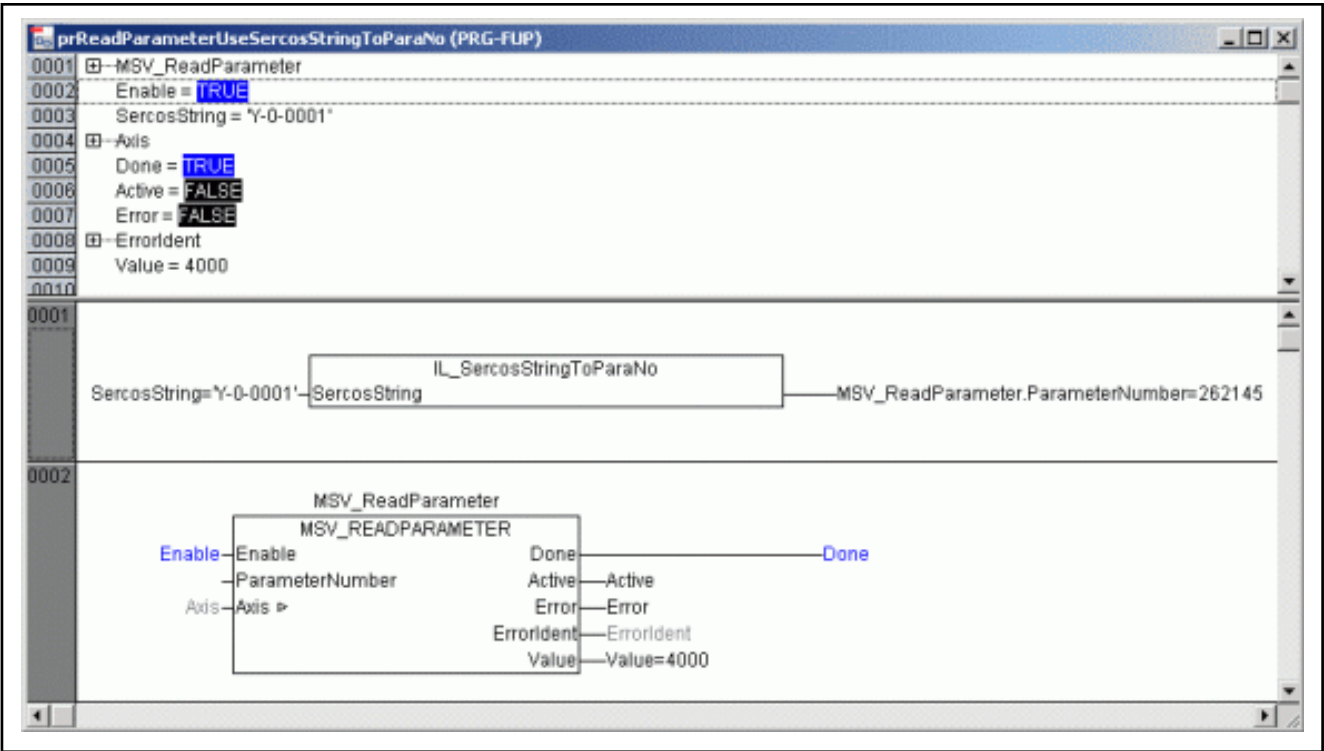


Fig. 4-26: Application of the function (FBD): IL_SercosStringToParaNo

4.6 IP settings - IP configuration

4.6.1 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.

Library	Range
ML_Base_xxx_yyyy	IP settings

Tab. 4-52: Library assignment

Interface Description

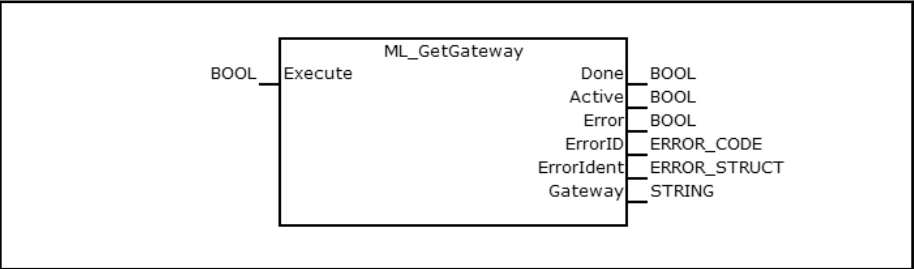


Fig. 4-27: ML_GetGateway function block

	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 the error cause

	Name	Type	Comment
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Gateway	STRING(255)	Read gateway address:

Tab. 4-53: ML_GetGateway interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous

Tab. 4-54: 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 343](#).

4.6.2 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.

Library	Range
ML_Base_xxx_yyyy	IP settings

Tab. 4-55: Library assignment

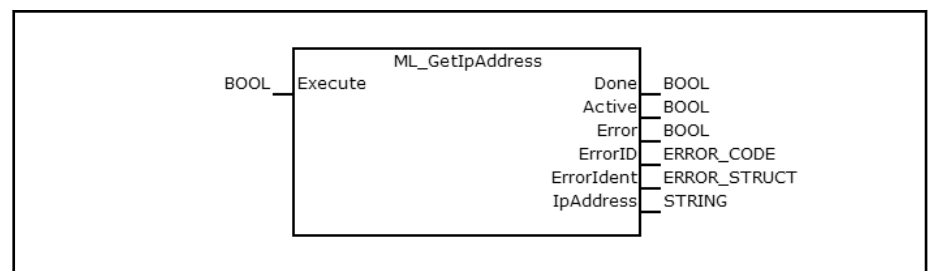
Interface Description

Fig. 4-28: 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

ML_Base.library

	Name	Type	Comment
	Error	BOOL	Indicates that an error occurred in the function block.
	ErrorID	ERROR_CODE	Brief note on the error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	IpAddress	STRING(255)	Read IP address

Tab. 4-56: ML_GetIpAddress Interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous

Tab. 4-57: 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 343](#).

4.6.3 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.

Library	Range
ML_Base_xxx_yyyy	IP settings

Tab. 4-58: Library assignment

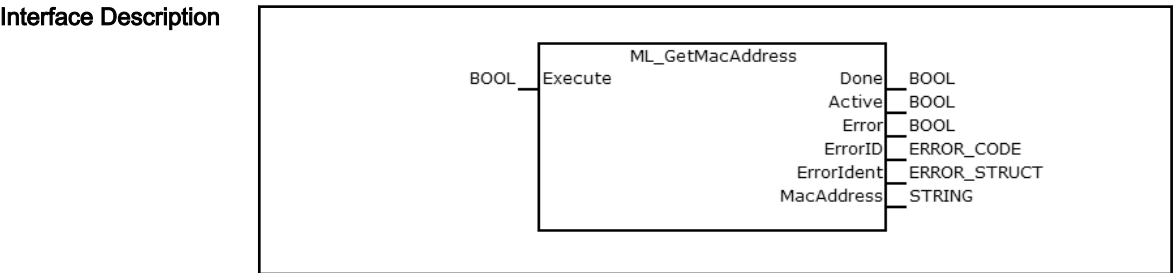


Fig. 4-29: ML_GetMacAddress function block

	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 the error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	MacAddress	STRING(255)	Read MAC address

Tab. 4-59: ML_GetMacAddress interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous

Tab. 4-60: 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 343](#).

4.6.4 ML_GetSubnetMask

Brief Description

This function block reads the subnet mask of the control. The subnet mask is determined by reading the parameter C-0-0205.

Library	Range
ML_Base_xxx_yyyy	IP settings

Tab. 4-61: Library assignment

ML_Base.library

Interface Description

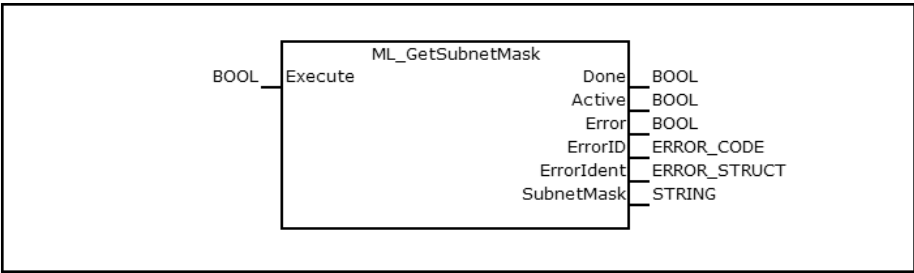


Fig. 4-30: ML_GetSubnetMask function block

	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 the error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	SubnetMask	STRING(255)	Read Subnet mask

Tab. 4-62: ML_GetSubnetMask interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous

Tab. 4-63: 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-0-0205.

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 343](#).

4.6.5 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.

Library	Range
ML_Base_xxx_yyyy	IP settings

Tab. 4-64: Library assignment

Interface Description

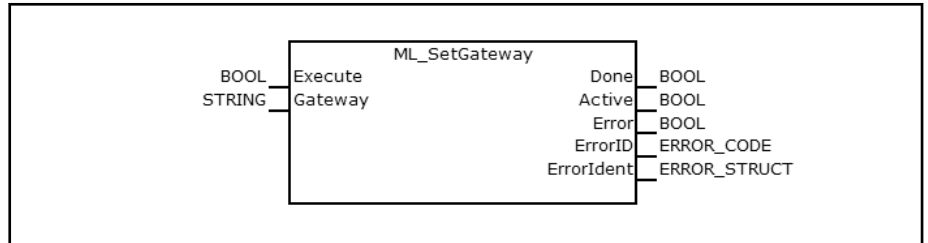


Fig. 4-31: ML_SetGateway function block

	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 the error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 4-65: ML_SetGateway interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
Gateway	STRING			"	Rising edge at "Execute"

Tab. 4-66: 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 343](#).

ML_Base.library

4.6.6 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.

Library	Range
ML_Base_xxx_yyyy	IP settings

Tab. 4-67: Library assignment

Interface Description

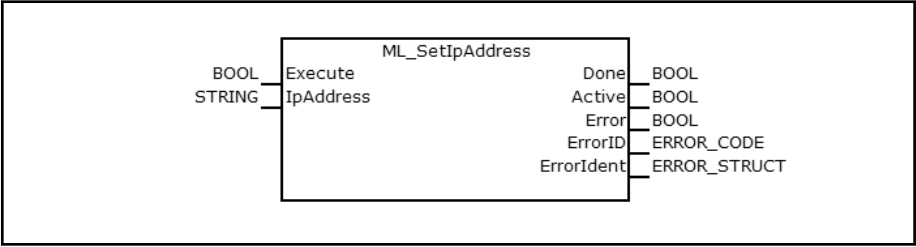


Fig. 4-32: ML_SetIpAddress function block

	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 the error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 4-68: ML_SetIpAddress Interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
IpAddress	STRING			"	Rising edge at "Execute"

Tab. 4-69: 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 343](#).

4.6.7 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.

Library	Range
ML_Base_xxx_yyyy	IP settings

Tab. 4-70: Library assignment

Interface Description

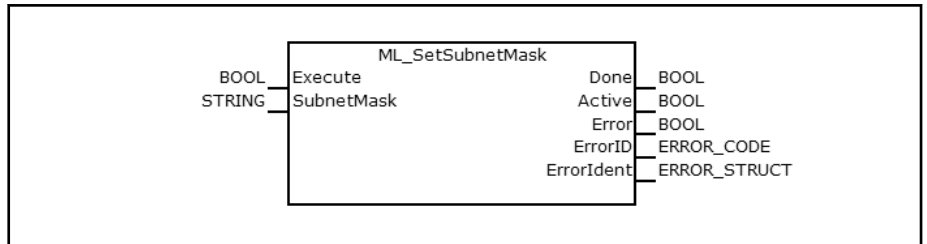


Fig. 4-33: ML_SetSubnetMask function block

	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 the error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 4-71: ML_SetSubnetMask interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
SubnetMask	STRING			"	Rising edge at "Execute"

Tab. 4-72: 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.

ML_Base.library

Example:

192.168.1.255

Error codes The function block uses the error table [MLC_TABLE](#), 16#0030 page 343.

4.7

SystemTime - system time

4.7.1

IL_SetDateTime

Brief Description This function block sets the system time of the control and of the connected drives.

Library	Range
ML_Base_xxx_yyyy	SystemTime

Tab. 4-73: Library assignment

Interface description

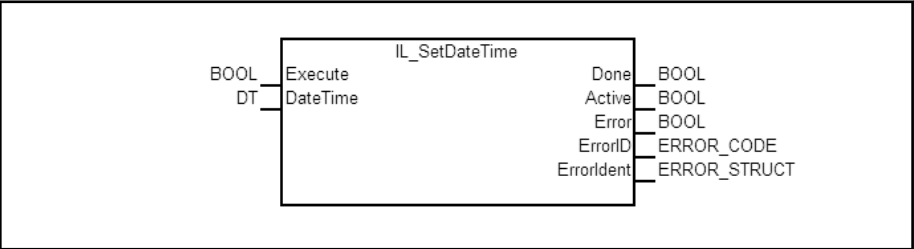


Fig. 4-34: IL_SetDateTime function block

	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. 4-74: IL_SetDateTime interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous
DateTime	DT	DT#1970-01-01-0 0:00:0		DT#1970-01-01-0 0:00:00	Rising edge at "Execute"

Tab. 4-75: 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 343](#) as well as the error code table F_RELATED_TABLE, 16#0170

4.7.2 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 .

Library	Range
ML_Base_xxx_yyyy	SystemTime

Tab. 4-76: Library assignment

Interface description

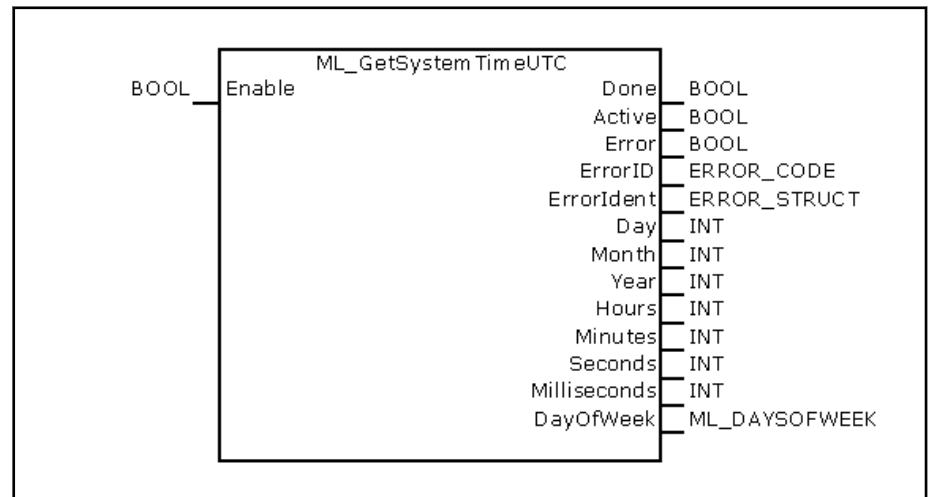


Fig. 4-35: Function block ML_GetSystemTimeUTC

	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

ML_Base.library

	Name	Type	Comment
	Milliseconds	INT	Milliseconds
	DayOfWeek	ML_DAYSOFWEEK	Week day

Tab. 4-77: ML_GetSystemTimeUTC interface

Name	Type	Min. value	Max. value	Default value	Applying data
Enable	BOOL			FALSE	Continuous

Tab. 4-78: 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", "Hour", "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 156](#),.

Example:

DayOfWeek: Friday

Error codes

The function block uses the error table [MLC_TABLE, 16#0030, page 343](#).

4.7.3 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 .

Library	Range
ML_Base_xxx_yyyy	SystemTime

Tab. 4-79: Library assignment

Interface Description

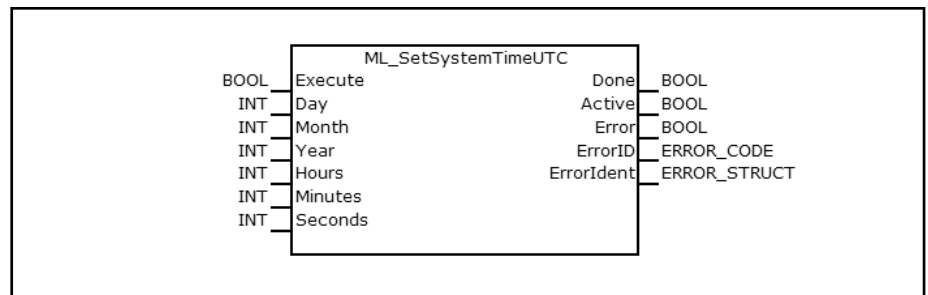


Fig. 4-36: ML_SetSystemTimeUTC function block

	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	Second
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 the error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 4-80: 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. 4-81: 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

ML_Base.library

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 343](#).

4.7.4 ML_DAYSOFWEEK

Library	Range
ML_Base_xxx_yyyy	SystemTime

Tab. 4-82: 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. 4-83: Definition of the ML_DAYSOFWEEK enumeration type

4.8 Target - Target system

4.8.1 ML_GetCpuLoad

Brief Description This function returns the current CPU load in percent.

Library	Range
ML_Base_xxx_yyyy	Target

Tab. 4-84: Library assignment

Interface Description

	Name	Type	Comment
RETURN	ML_GetCpuLoad	REAL	CPU load in percent

Tab. 4-85: 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 CPU load contained in percent.

4.8.2 ML_GetDiagnosisLog

Brief description

The ML_GetDiagnosisLog function block reads entries from the diagnostic logbook.

Library	Range
ML_Base_xxx_yyyy	Error/diagnostic memory

Tab. 4-86: Library assignment

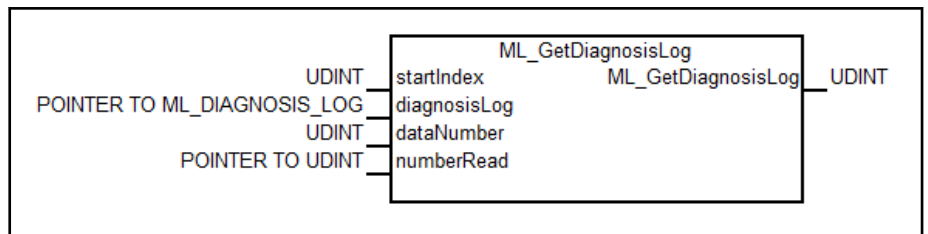
Interface description

Fig. 4-37: ML_GetDiagnosisLog function

	Name	Type	Comment
VAR_INPUT	startIndex	UDINT	Start index from which the diagnostic logbook is to be saved
	diagnosisLog	POINTER TO ML_DIAGNOSIS_LOG	Address on which the diagnostic logbook entries are to be saved See chapter 4.8.17 "ML_DIAGNOSIS_LOG" on page 163
	dataNumber	UDINT	Size of diagnosisLog (number of entries)
	numberRead	POINTER TO UDINT	Address of the number of read logbook entries
RETURN		UDINT	Brief note on error cause

Tab. 4-87: Interface of the FUN ML_GetDiagnosisLog

Functional description

The function reads a number of diagnostic logbook 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.

4.8.3 ML_GetFirmwareString

Brief Description

This function returns the firmware string of the control.

Library	Range
ML_Base_xxx_yyyy	Target

Tab. 4-88: Library assignment

Interface Description

ML_Base.library

	Name	Type	Comment
RETURN	ML_GetFirmwareString	STRING	Firmware ID of the control.

Functional Description

Tab. 4-89:Interface of the ML_GetFirmwareString

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-12V02.0072

4.8.4ML_GetHardwareInfoCtrl

Brief descriptionThis function returns information on the control hardware.

The function is not provided with any input, any return value, but four outputs (VAR_OUTPUT).

Library	Range
ML_Base_xxx_yyyy	Target

Interface description

Tab. 4-90:Library assignment

ML_GetHardwareInfoCtrl

TypeCode

STRING(40)

MaterialNo

STRING(40)

MaterialIndex

STRING(40)

SerialNo

STRING(40)

Fig. 4-38: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

Functional description

Tab. 4-91:Interface of the ML_GetHardwareInfoCtrl

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);
```

PlcProg[IndraMotionMlc1: Logic: Application]				
IndraMotionMlc1.Application.PlcProg				
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[CML65.1-3P], MaterialNo=> strMaterialNo[R911170900],				
2 MaterialIndex=> strMaterialIndex[301], SerialNo=> strSerialNo[006277436] ;RETURN				

Fig. 4-39: Response of the ML_GetHardwareInfoCtrl function

4.8.5 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.

Library	Range
ML_Base_xxx_yyyy	Target

Tab. 4-92: Library assignment

Interface Description

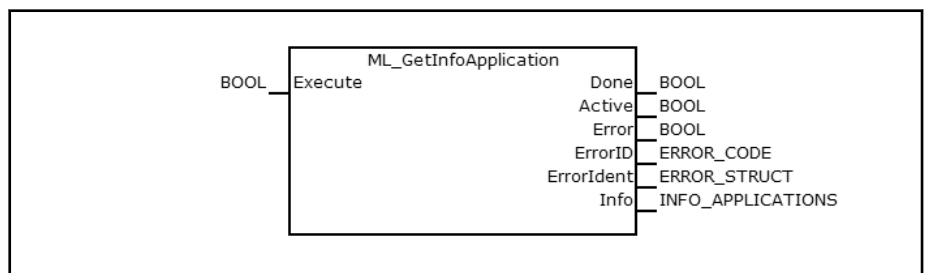


Fig. 4-40: ML_GetInfoApplication function block

	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 the error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Info	INFO_APPLICATIONS	Application information

Tab. 4-93: ML_GetInfoApplication interface

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Continuous

Tab. 4-94: 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.

ML_Base.library

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](#), [page 163](#), containing the number of applications, their names and statuses.

Error codes The function block uses the error table [MLC_TABLE](#), [16#0030](#), [page 343](#).

4.8.6 ML_GetMaxTemperature

Brief Description This function returns the highest temperature reached of the control.

Library	Range
ML_Base_xxx_yyyy	Target

Tab. 4-95: Library assignment

Interface Description

	Name	Type	Comment
RETURN	ML_GetMaxTemperature	REAL	Maximum temperature of the control

Tab. 4-96: 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](#), [page 161](#),.

The maximum temperature is reset when switching on the control.

4.8.7 ML_GetNewestDiagnosisIndex

The ML_GetNewestDiagnosisIndex function returns the latest diagnostic log-book index.

Library	Range
ML_Base_xxx_yyyy	Error/diagnostic memory

Tab. 4-97: Library assignment

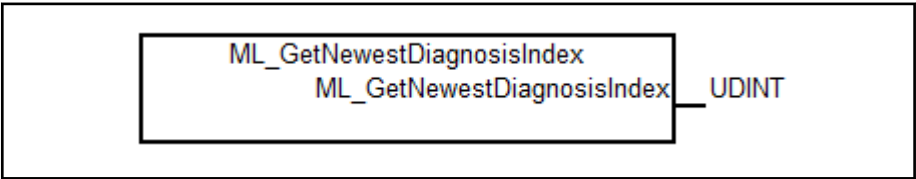


Fig. 4-41: ML_GetNewestDiagnosisIndex function

	Name	Type	Comment
RETURN		UDINT	Latest index of the diagnostic logbook

Tab. 4-98: Interface of the FUN ML_GetNewestDiagnosisIndex

4.8.8 ML_GetOhcFan

Brief Description This function returns the operating hours of the fan of the control.

Library	Range
ML_Base_xxx_yyyy	Target

Tab. 4-99: Library assignment

Interface Description

	Name	Type	Comment
RETURN	ML_GetOhcFan	REAL	Operating hours of the fan of the control

Tab. 4-100: 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.

4.8.9 ML_GetScalingType

Brief Description This function returns the unit of all temperature specifications of the control.

Library	Range
ML_Base_xxx_yyyy	Target

Tab. 4-101: Library assignment

Interface Description

	Name	Type	Comment
RETURN	ML_GetScalingType	E_SCALING_TYPE	Unit of all temperature specification of the control

Tab. 4-102: 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](#), page 163.

4.8.10 ML_GetTargetInfo

Brief Description This function returns information on the target system.

Library	Range
ML_Base_xxx_yyyy	Target

Tab. 4-103: Library assignment

Interface Description

	Name	Type	Comment
RETURN	ML_GetTargetInfo	TARGET_INFO	Information on the target system

Tab. 4-104: ML_GetTargetInfo interface

Functional Description This function returns information on the target system.
The return value of the function is of the enumeration type [TARGET_INFO](#), page 164.

4.8.11 ML_GetTemperature

Brief Description This function returns the current temperature of the control.

ML_Base.library

Library	Range
ML_Base_xxx_yyyy	Target

Tab. 4-105: Library assignment

Interface Description

	Name	Type	Comment
RETURN	ML_GetTemperature	REAL	Current temperature of the control

Tab. 4-106: 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](#), page 161,.

4.8.12 ML_GetWarningCpuLoad

Brief Description This function returns the maximum CPU load set in percent.

Library	Range
ML_Base_xxx_yyyy	Target

Tab. 4-107: Library assignment

Interface Description

	Name	Type	Comment
RETURN	ML_GetWarningCpuLoad	REAL	Maximum CPU load set in percent

Tab. 4-108: 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.

4.8.13 ML_SetWarningCpuLoad

Brief Description This function sets the maximum CPU load allowed in percent.

Library	Range
ML_Base_xxx_yyyy	Target

Tab. 4-109: Library assignment

Interface Description

	Name	Type	Comment
VAR_INPUT	CpuLoad	REAL	Maximum CPU load allowed in percent
RETURN	ML_GetWarningCpuLoad	REAL	

Tab. 4-110: 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.

4.8.14 APP_INFO

The data type "APP_INFO" is used in the structure [INFO_APPLICATIONS](#) [page 163](#).

The data type "APP_INFO" is a structure with the following entries:

Element	Type	Description
Name	STRING	Name of the PLC application
State	STRING	State of the PLC application

Tab. 4-111: Definition of the APP_INFO structure

4.8.15 E_SCALING_TYPE

The "E_SCALING_TYPE" data type is used in the function [ML_GetScaling-Type](#) [page 161](#).

Library	Range
ML_Base_xxx_yyyy	Target

Tab. 4-112: 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. 4-113: Definition of the enumeration type E_SCALING_TYPE

4.8.16 INFO_APPLICATIONS

The "INFO_APPLICATIONS" data type is used in the [ML_GetInfoApplication](#) [page 159](#).

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

Tab. 4-114: Definition of the INFO_APPLICATIONS structure

4.8.17 ML_DIAGNOSIS_LOG

Brief description This structure contains an entry of the diagnostic logbook.

Library	Range
ML_Base_xxx_yyyy	Data types/Info

Tab. 4-115: Library assignment

ML_Base.library

Structure element	Type	Description
number	UDINT	Error number
state	UDINT	Diagnostic input status
dateTime	SYS_TIME_DATE	Diagnostic point in time
logicalAddress	DINT	Logic device address (e.g. axis, kinematics, ...)
physicalAddress	DINT	Physical device address (e.g. drive)
dispatcher	UDINT	Diagnostic source
index	UDINT	Index of the diagnostics in the diagnostic logbook
text	STRING(123)	Diagnostic text

Tab. 4-116: ML_DIAGNOSIS_LOG data structure

4.8.18 TARGET_INFO

The data type "TARGET_INFO" is used in the function [ML_GetTargetInfo](#) [page 161](#).

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. 4-117: Definition of the TARGET_INFO structure

5 Function Blocks According to the 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 169](#), 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 353](#), includes function blocks and data types to activate the 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	-

Function Blocks According to the PLCopen Specification

Function block	ML_PLCopen	RMB_PLCopenFieldbus
ML_OpenCyclicAnalogChannel	X	-
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	-

Function Blocks According to the PLCopen Specification

Function block	ML_PLCOpen	RMB_PLCOpenFieldbus
MB_ReadStringParameter	X	X
MB_ReadUnit	X	-
MB_WriteLintParameter	X	-
MB_WriteListParameter	X	X
MB_WriteListSegParameterPtr	-	X
MB_WriteLrealParameter	X	-
MB_WriteParameter	X	X
MB_WriteRealParameter	X	X
MB_WriteStringParameter	X	X
MC_RealActualPosition	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_PLCOpen-FieldBus.library

6 ML_PLCopen.library

6.1 ML_PLCopen, General

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, VAR GLOBAL/EXTERNAL) are covered in the IEC standard EN 61131-3 or defined by the company in the following libraries:

- [RIL_CommonTypes](#), page 37, data types for general use in all IndraWorks systems,
- [ML_Base](#), page 81, data types of the IndraLogic XLC and IndraMotion MLC systems 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.

Global Variables

- [ProbeSelect](#), page 172, defines the field containing the data of the touch probes 1..100.

Data types

- *Data types of TouchProbe*
 - [PROBECONFIG](#), page 172, defines the data type for the data of the touch probes 1..100.

POUs

- *Controlled Motion*
 - [ML_ControlOff](#), page 264
 - [ML_ControlOn](#), page 264
 - [ML_ControlWriteActualData](#), page 265,
 - [ML_OpenCyclicAnalogChannelCtrl](#), page 266
 - [ML_OpenCyclicPositionChannelCtrl](#), page 268
 - [ML_OpenCyclicTorqueChannelCtrl](#), page 270
 - [ML_OpenCyclicVelocityChannelCtrl](#), page 272
 - [ML_WriteCyclicAnalogCtrl](#), page 267
 - [ML_WriteCyclicPositionCtrl](#), page 269
 - [ML_WriteCyclicTorqueCtrl](#), page 271
 - [ML_WriteCyclicVelocityCtrl](#), page 274
- *Motion*
 - [MB_ChangeCamData](#), page 223,
 - [MB_ChangeFlexEventSet](#), page 239,
 - [MB_ChangeFlexProfileSet](#), page 236,
 - [MB_ChangeProfileSet](#), page 216,
 - [MB_ChangeProfileStep](#), page 219,
 - [MB_GearInPos](#), page 245,
 - [MB_Home](#), page 196,
 - [MB_MotionProfile](#), page 210,

ML_PLCopen.library

- [MB_Phasing](#), page 201,
- [MB_PhasingSlave](#), page 203,
- [MB_SetPositionControlMode](#), page 195,
- [MC_AddAxisToGroup](#), page 248,



The "MC_AddAxisToGroup" and "MC_RemoveAxisFromGroup" function blocks have been moved from ML_Robot to ML_PLCopen, to make them available for IndraLogic XLC.

- [MC_CamIn](#), page 205,
- [MC_CamOut](#), page 209,
- [MC_GearIn](#), page 240,
- [MC_GearOut](#), page 243,
- [MC_MoveAbsolute](#), page 180,
- [MC_MoveAdditive](#), page 183,
- [MC_MoveRelative](#), page 185,
- [MC_MoveVelocity](#), page 189,
- [MC_Power](#), page 176,
- [MC_RemoveAxisFromGroup](#), page 249,



The "MC_AddAxisToGroup" and "MC_RemoveAxisFromGroup" function blocks have been moved from ML_Robot to ML_PLCopen, to make them available for IndraLogic XLC.

- [MC_Reset](#), page 177,
- [MC_Stop](#), page 178,
- [MC_TorqueControl](#), page 192,
- [ML_FlexProfile](#), page 224,
- [ML_OpenCyclicAnalogChannel](#), page 252,
- [ML_OpenCyclicPositionChannel](#), page 254,
- [ML_OpenCyclicTorqueChannel](#), page 258,
- [ML_OpenCyclicVelocityChannel](#), page 260,
- [ML_WriteCyclicAnalog](#), page 253,
- [ML_WriteCyclicPosition](#), page 255,
- [ML_WriteCyclicTorque](#), page 259,
- [ML_WriteCyclicVelocity](#), page 261,

- *Parameters*

- [MB_Command](#), page 277,
- [MB_ReadLintParameter](#), page 281,
- [MB_ReadLintParameter](#), page 283,
- [MB_ReadLrealListParameter](#), page 285,
- [MB_ReadLrealParameter](#), page 288,
- [MB_ReadMaxRealValue](#), page 290,
- [MB_ReadMaxValue](#), page 291,
- [MB_ReadMinRealValue](#), page 293,

- [MB_ReadMinValue, page 294,](#)
- [MB_ReadName, page 296,](#)
- [MB_ReadParameter, page 297,](#)
- [MB_ReadRealParameter, page 299,](#)
- [MB_ReadSercosAttribute, page 300,](#)
- [MB_ReadSercosDataStatus, page 302,](#)
- [MB_ReadStringParameter, page 304,](#)
- [MB_ReadUnit, page 305,](#)
- [MB_WriteLintParameter, page 306,](#)
- [MB_WriteListParameter, page 308,](#)
- [MB_WriteLrealParameter, page 310,](#)
- [MB_WriteParameter, page 312.](#)
- [MB_WriteRealParameter, page 313,](#)
- [MB_WriteStringParameter, page 315,](#)
- [MC_ReadActualPosition, page 316,](#)
- [MC_ReadAxisError, page 317,](#)
- [MC_ReadStatus, page 318,](#)

The function blocks "IL_SercosAttribute", "IL_SercosElementsToParaNo" and "IL_SercosStringToParaNo" are part of the ML_Base_xx.yy.libraries in 12VRS and higher.



A maximum of 100 parameter queries can be triggered per parameter channel.

Parameter channels:

- A-parameters
- C-parameters
- S-/P-parameters
- K-parameters
- ...

- *Touch probe*

- [ML_AbortTrigger, page 321](#)
- [ML_InitTouchProbe, page 322](#)
- [ML_TouchProbe, page 329](#)
- [ML_TouchProbeContinuous, page 331](#)
- [ML_WriteExpectWindow, page 334](#)

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).

ML_PLCopen.library

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.



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

Short 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 172.

Library	Range
ML_PLCopen	Global_Variables

Tab. 6-1: 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

Short 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. 6-2: 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_GearInPos, 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.

ML_PLCOpen.library

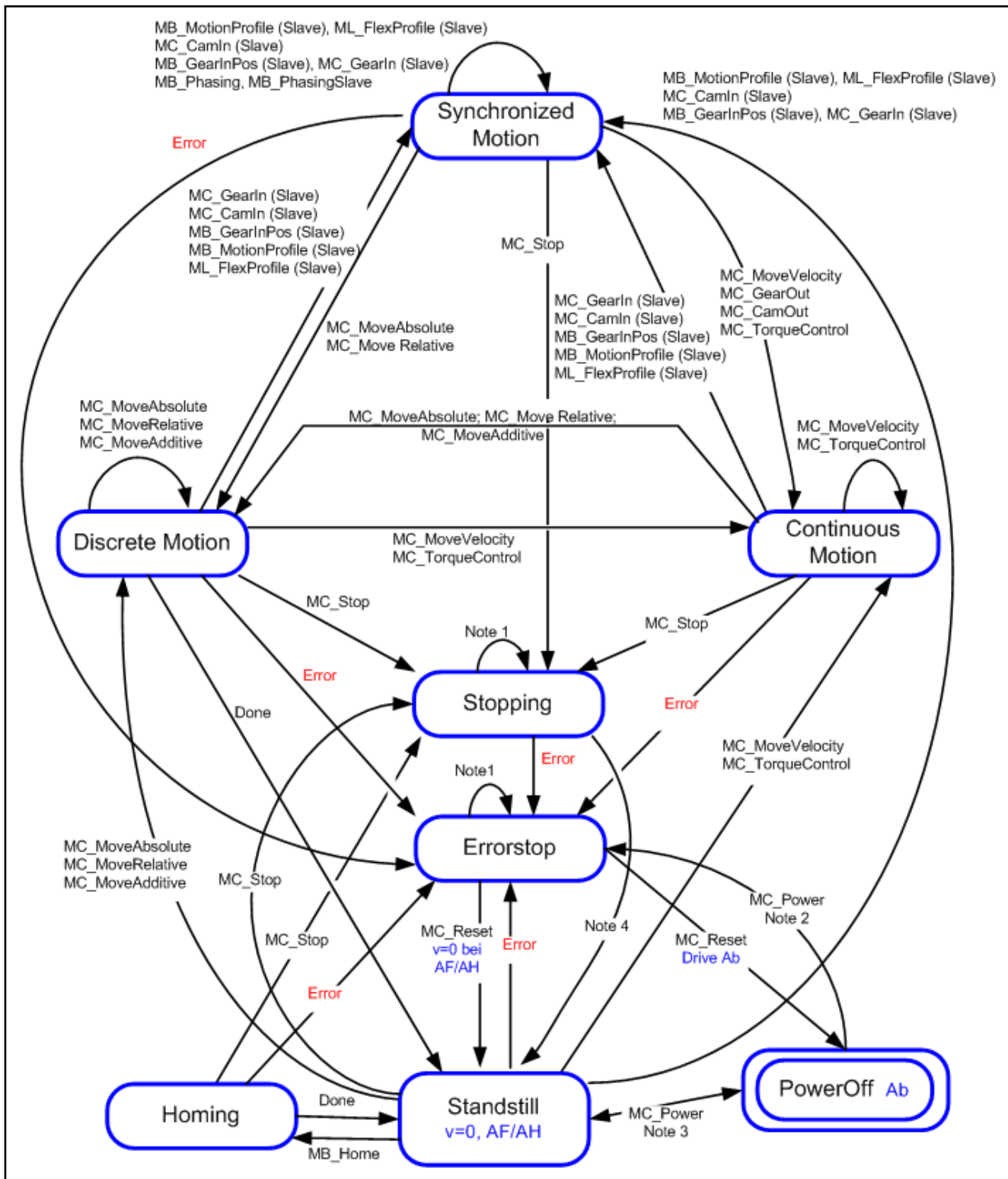


Fig. 6-1: IndraMotion state diagram of a real axis

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 fault-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 status [Controlled Motion](#), [page 262](#), for controlled motions.

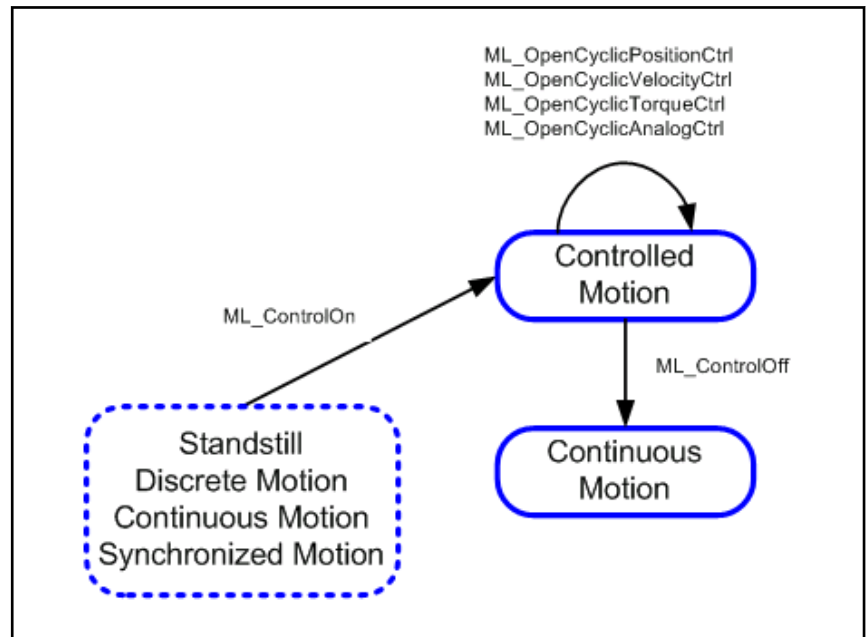


Fig. 6-2: Transition into the "Controlled Motion" status

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).

ML_PLCopen.library

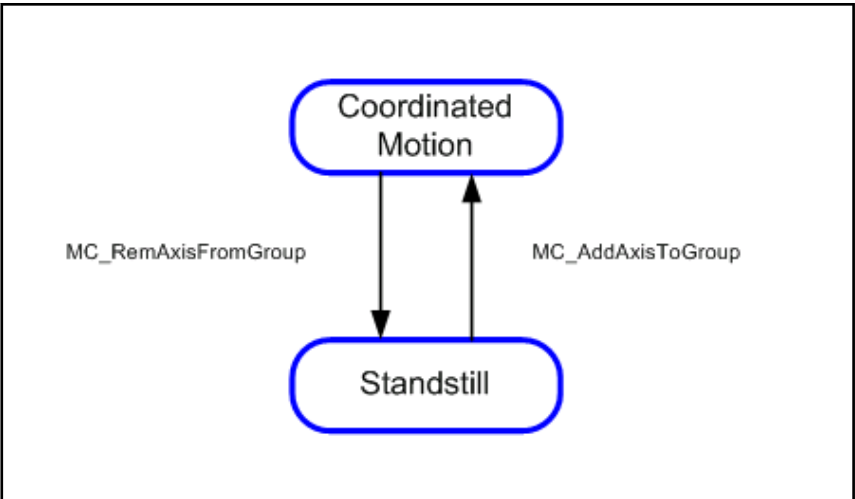


Fig. 6-3: Transition into "Coordinated Motion" state

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 178](#). 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
ML_PLCopen.library	Motion

Tab. 6-3: Library assignment

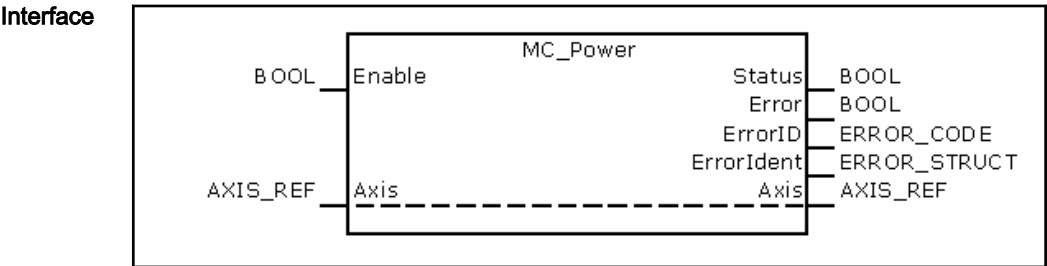


Fig. 6-4: MC_Power function block

	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	Power adding state Refer to "Special features of MC_Power" on page 177
	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. 6-4: 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 standstill. "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-

6.5.2 MC_Power, special features of Sercos-based axes

Troubleshooting:

- 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 343](#) - error when accessing the axis/control.

6.5.3 MC_Reset

Brief Description

This function block implements the transition from the 'ErrorStop' state to the 'StandStill' state by resetting all axis- and control-internal errors. The axis-internal errors are deleted via the corresponding drive number in the AXIS_REF. The control errors can be reset using the ENUM NO_OBJECT or the NoAxis axis reference. The MC_Reset function block uses parameter C-0-1040 or A-0-1040 in the parameterization mode (phase 2). MLC-internal functions are executed in the operating mode (phase 4).



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.

ML_PLCOpen.library

Library	Range
ML_PLCOpen.library	Motion

Tab. 6-5: Library assignment

Interface Description

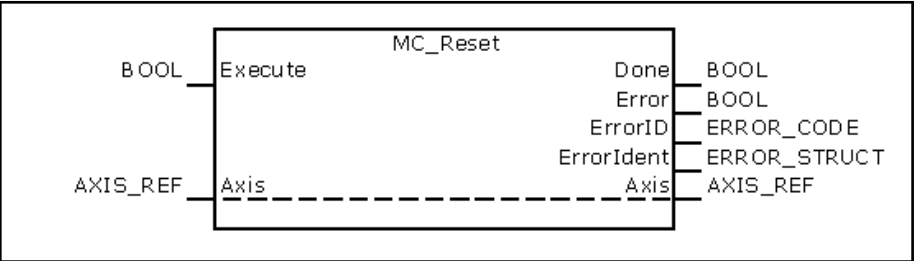


Fig. 6-5: MC_Reset function block

	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
	ErrorID	ERROR_CODE	Brief note on the error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 6-6: Interface of the MC_Reset function block

Signal-time diagram

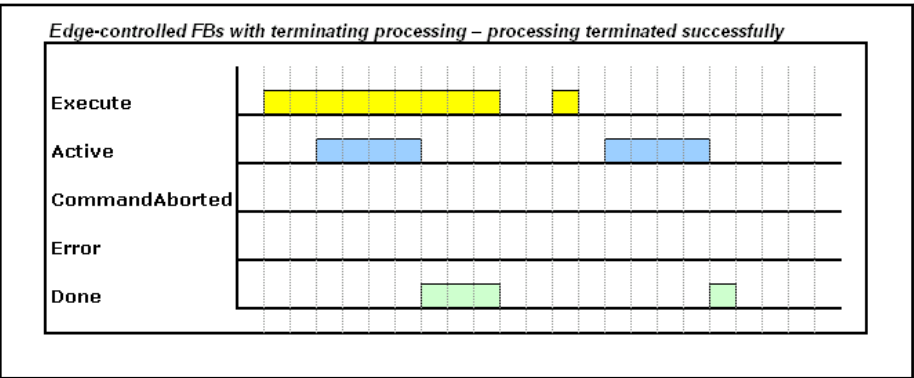


Fig. 6-6: Time diagram of MC_Reset

6.5.4 MC_Reset, Special Features of SERCOS-Based Axes

The function block uses the parameter C-0-1030, Command: Clear all control errors.

- Error handling
- The function block MC_Reset generates the following error messages:
- [MLC_TABLE](#), page 343 - error when accessing the axis/control.

6.5.5 MC_Stop

Brief Description

This function block implements a controlled motion stop and transfers the ax-
is to the "Stopping" state. As a result, any processing function block is stop-
ped. When standstill is reached, the "Done" output is set.
The position is kept active.



As long as "Execute" is TRUE, the axis remains in the "Stopping" state. In this state, the axis cannot be moved by any other command.

In the "Stopping" state, it is not possible to switch off the axis power ([MC_Power](#), page 176,). 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
ML_PLCopen.library	Motion

Tab. 6-7: Library assignment

Interface Description

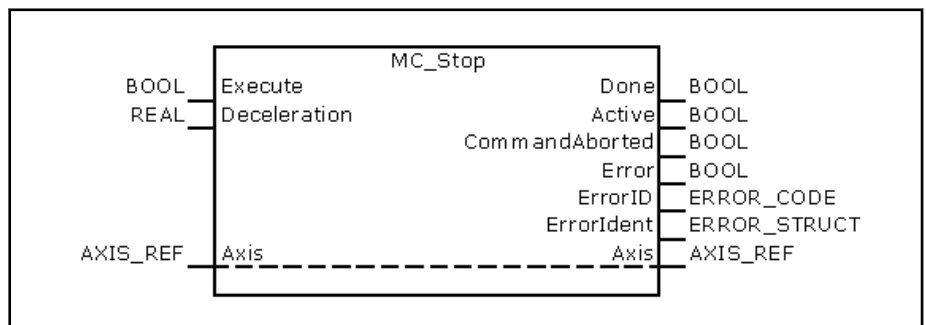


Fig. 6-7: MC_Stop function block

	Name	Type	Comment ¹⁾
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
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 the error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 6-8: Interface of the MC_Stop function block

¹⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

ML_PLCopen.library

Signal-time diagram

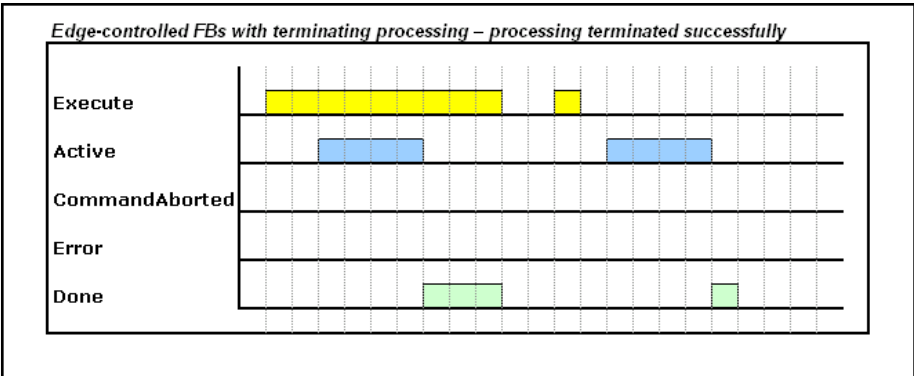


Fig. 6-8: Time diagram of MC_Stop

6.5.6 MC_Stop, Special Features of SERCOS-Based Axes

Error handling The function block MC_Stop generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

6.5.7 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 176](#), (for real axes).

- 
1. In addition to the function block inputs, the jerk limit value set via parameter A-0-0216 influences the function block.
The behavior described in the following occurs if the jerk limit (A-0-0216 > 0) is active
 2. During the acceleration phase, the acceleration input of the function block is always decisive.
If the axis velocity has to be reduced during the acceleration phase (e.g., overshooting of the actual velocity), the acceleration input is the limit value for the required delay.
In the following case, it can occur that the axis is decelerated with a larger delay than specified at the deceleration input:
 - Jerk limit is active (A-0-0216 > 0) AND
 - acceleration input > deceleration input AND
 - Axis is in acceleration phase AND
 - Motion restart with reduced positioning velocity
 3. Without jerk limit (A-0-0216 = 0), the behavior does not occur

Library	Range
ML_PLCopen.library	Motion

Tab. 6-9: Library assignment

Interface description

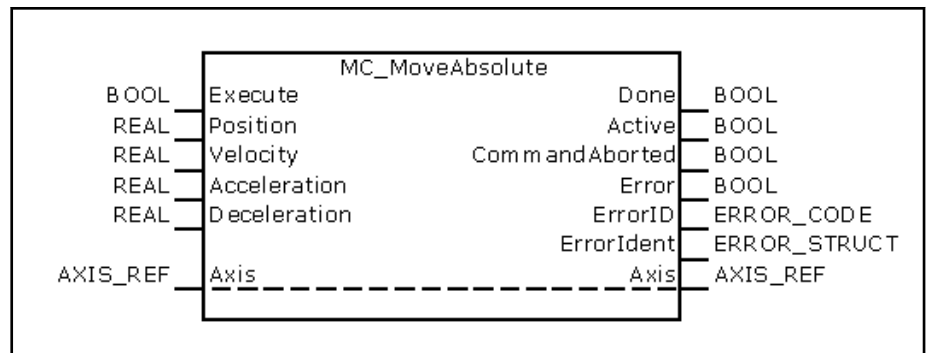


Fig. 6-9: MC_MoveAbsolute function block

	Name	Type	Comment ²⁾
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. 6-10: 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. In this case, 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.

ML_PLCopen.library

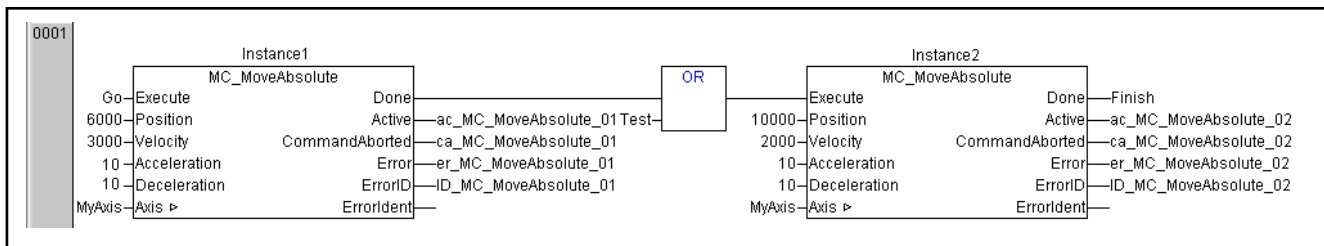


Fig. 6-10: Two instances of MC_MoveAbsolute

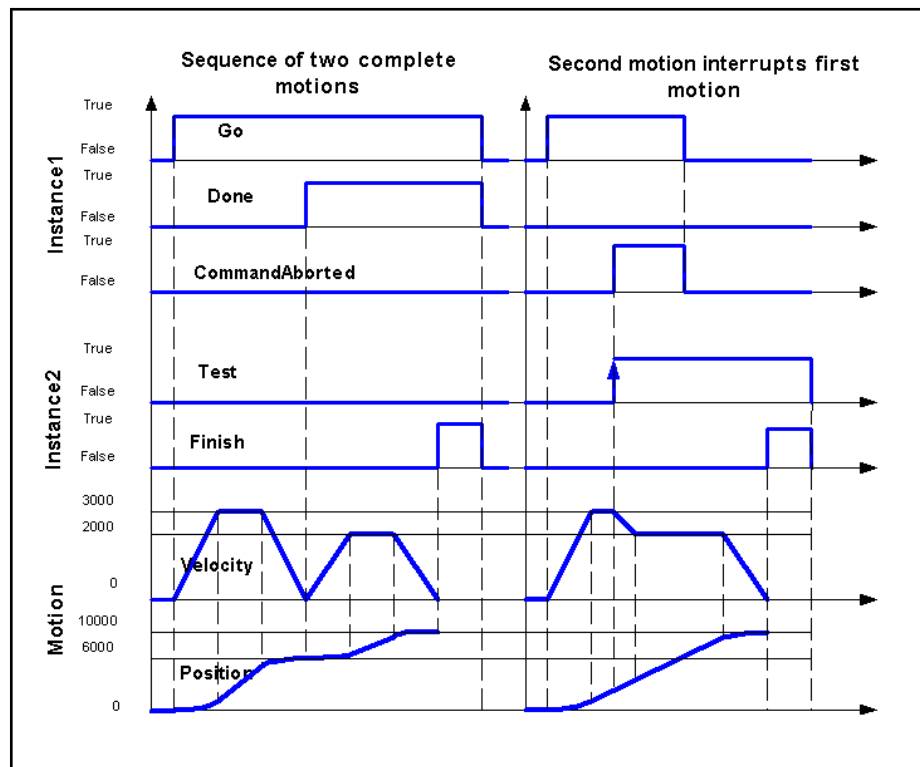


Fig. 6-11: Time diagram: Two instances of MC_MoveAbsolute

6.5.8 MC_MoveAbsolute, Special Features of SERCOS-Based Axes

Parameters used 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 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 traveled. The target position is only overwritten when another Motion function block is called.

Error handling: The function block MC_MoveAbsolute generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

6.5.9 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 176, (for real axes).



1. In addition to the function block inputs, the jerk limit value set via parameter A-0-0216 influences the function block.

The behavior described in the following occurs if the jerk limit (A-0-0216 > 0) is active

2. During the acceleration phase, the acceleration input of the function block is always decisive.

If the axis velocity has to be reduced during the acceleration phase (e.g., overshooting of the actual velocity), the acceleration input is the limit value for the required delay.

In the following case, it can occur that the axis is decelerated with a larger delay than specified at the deceleration input:

- Jerk limit is active (A-0-0216 > 0) AND
- acceleration input > deceleration input AND
- Axis is in acceleration phase AND
- Motion restart with reduced positioning velocity

3. Without jerk limit (A-0-0216 = 0), the behavior does not occur

Library	Range
ML_PLCopen.library	Motion

Tab. 6-11: Library assignment

Interface description

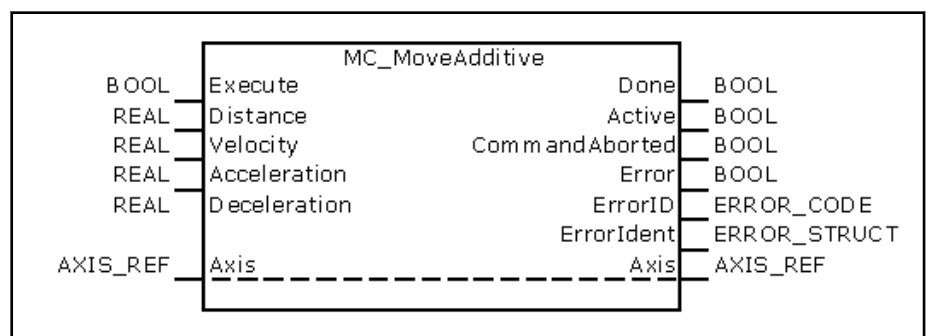


Fig. 6-12: MC_MoveAdditive function block

	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
	Distance	REAL	Relative distance for the motion (+/-).

³⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

ML_PLCOpen.library

	Name	Type	Comment ³⁾
	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. 6-12: 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 desired distance of 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.

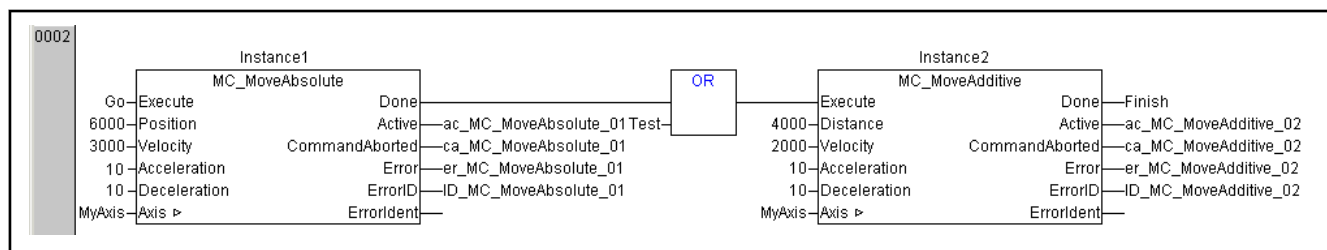


Fig. 6-13: Instances of MC_MoveAbsolute and MC_MoveAdditive

³⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

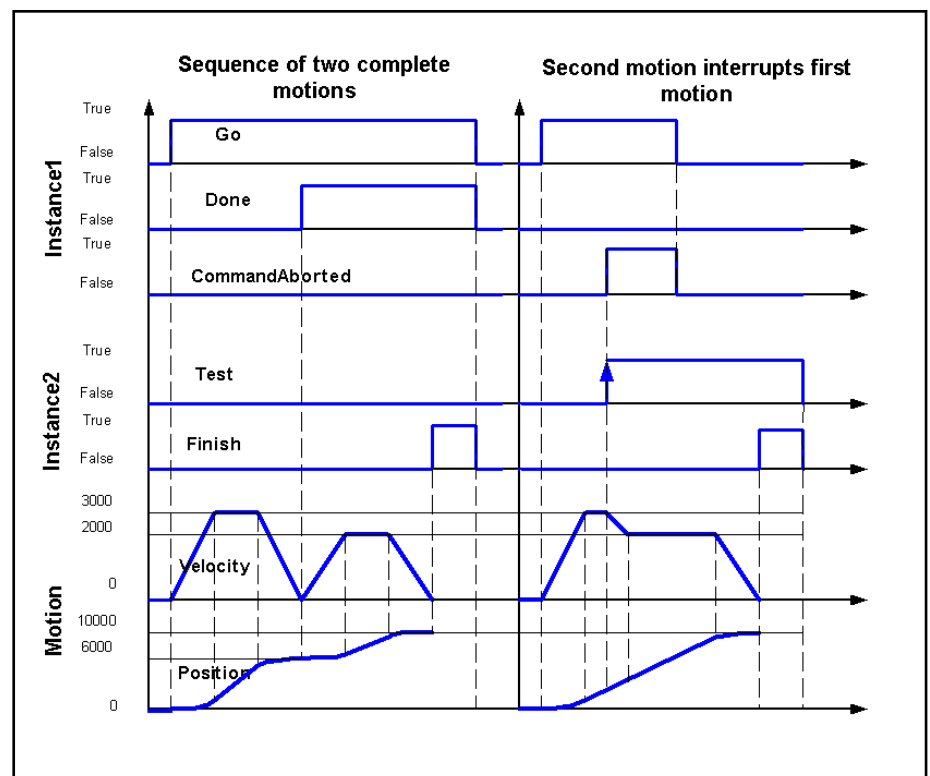


Fig. 6-14: Time diagram: Instances of MC_MoveAbsolute and MC_MoveAdditive

6.5.10 MC_MoveAdditive, Special Features of SERCOS-Based Axes

Parameters used 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 effective 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 traveled. The target position is only overwritten when another Motion function block is called.

Error handling: The function block MC_MoveAdditive generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

6.5.11 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 176](#), (for real axes).

ML_PLCOpen.library



1. In addition to the function block inputs, the jerk limit value set via parameter A-0-0216 influences the function block.

The behavior described in the following occurs if the jerk limit (A-0-0216 > 0) is active

2. During the acceleration phase, the acceleration input of the function block is always decisive.

If the axis velocity has to be reduced during the acceleration phase (e.g., overshooting of the actual velocity), the acceleration input is the limit value for the required delay.

In the following case, it can occur that the axis is decelerated with a larger delay than specified at the deceleration input:

- Jerk limit is active (A-0-0216 > 0) AND
 - acceleration input > deceleration input AND
 - Axis is in acceleration phase AND
 - Motion restart with reduced positioning velocity
3. Without jerk limit (A-0-0216 = 0), the behavior does not occur

Library	Range
ML_PLCOpen.library	Motion

Tab. 6-13: Library assignment

Interface description

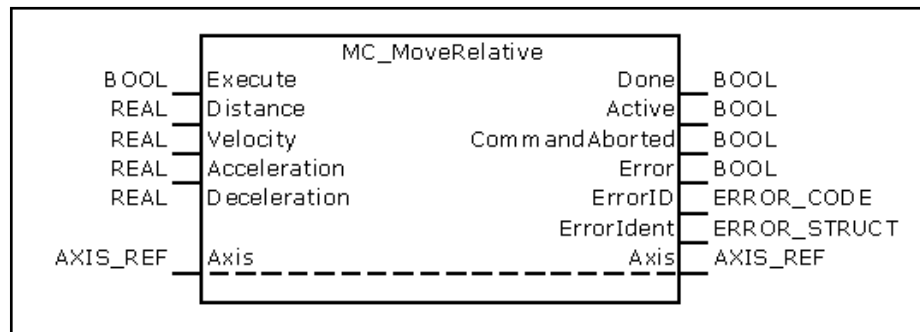


Fig. 6-15: MC_MoveRelative function block

	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

⁴⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

	Name	Type	Comment ⁴⁾
	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. 6-14: Interface of the MC_MoveRelative function block

Signal-time diagram

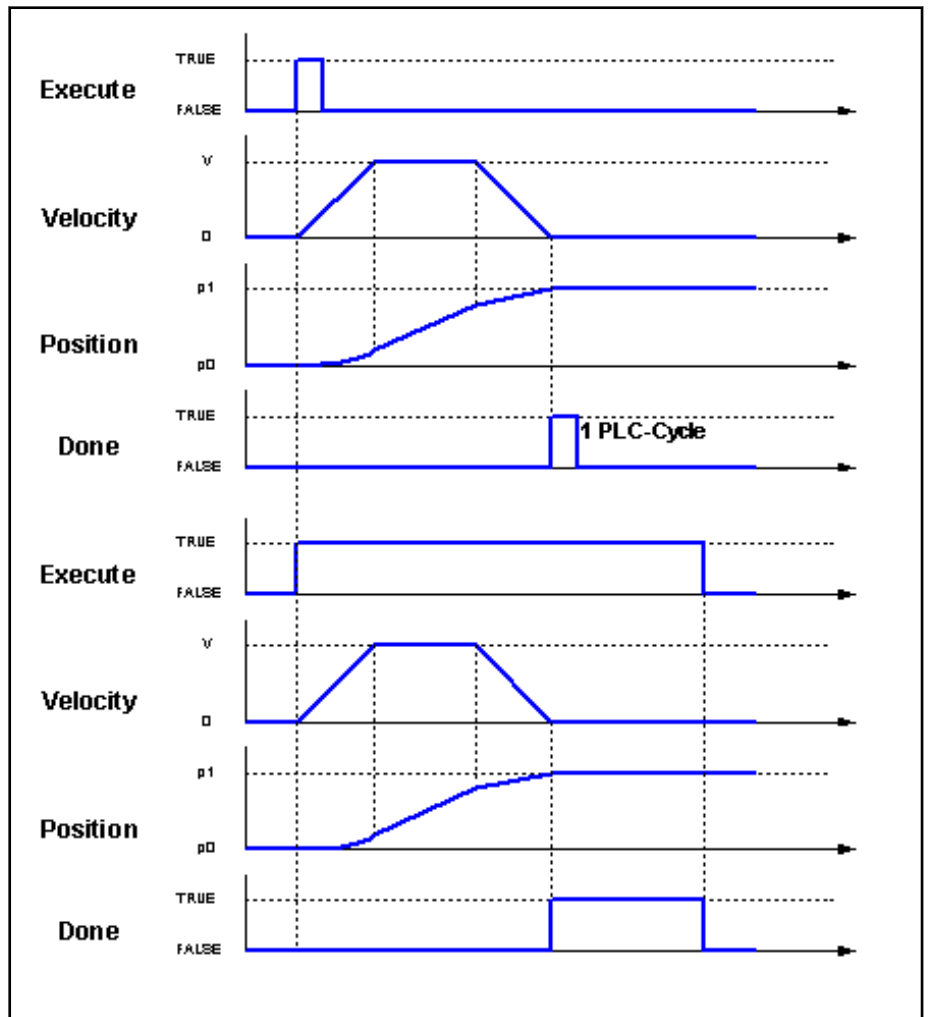


Fig. 6-16: Time diagrams for MC_MoveRelative

Implementation Example

The following example shows the combination of two MC_MoveRelative function block instances.

1. The left part of the time diagram shows a case in which "Instance2" is called after "Instance1" has been 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.

⁴⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

ML_PLCOpen.library

2. The right part shows how "Instance2" is already enabled by the "Test" signal while "Instance1" is still running. In this case, 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.

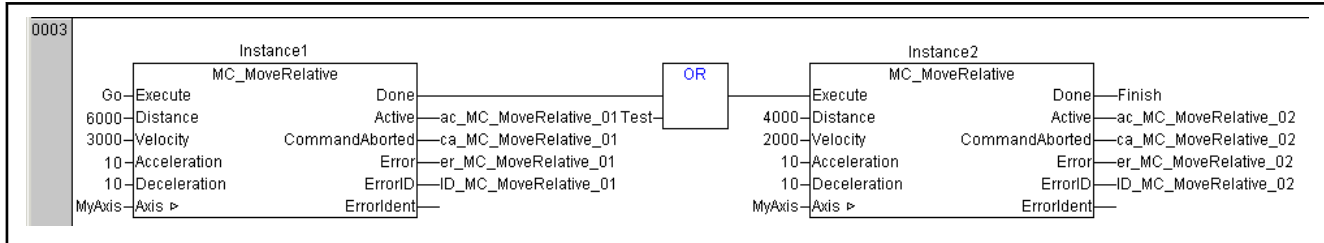


Fig. 6-17: Two instances of MC_MoveRelative

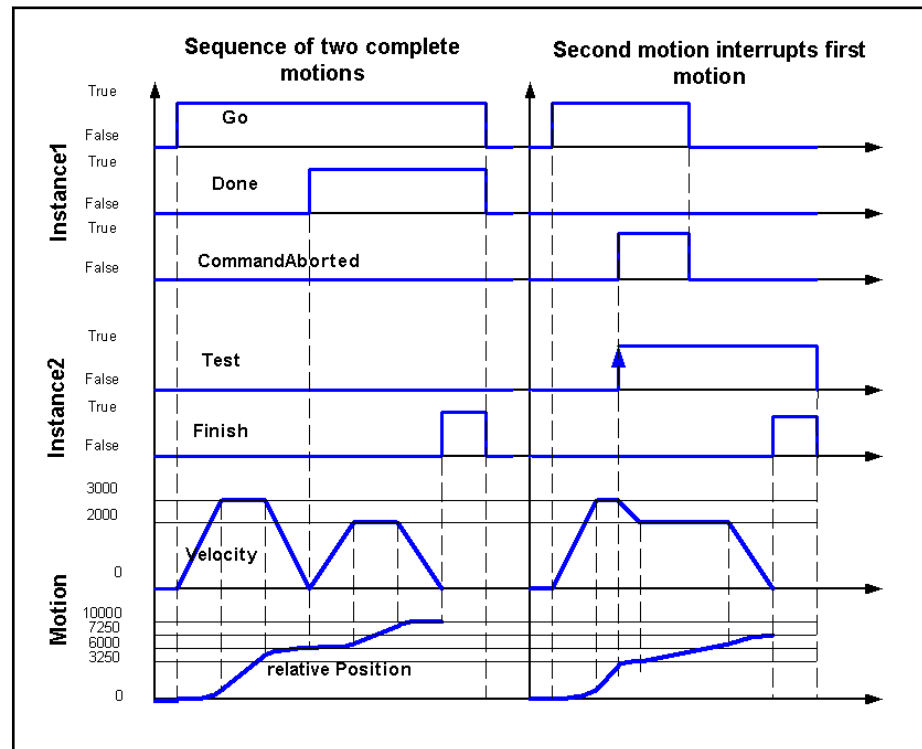


Fig. 6-18: Time diagram: Two instances of MC_MoveRelative

6.5.12 MC_MoveRelative, Special Features of SERCOS-Based Axes

Parameters used 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 (A-0-0100, Actual position value) and approaching it.



The distance is specified relative to the actual position. The parameter A-0-0203, Direction of motion, is not effective 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 $\neq$ 0) is called subsequently, the remaining distance is traveled. The target position is only overwritten when another Motion function block is called.

Error handling: The function block MC_MoveRelative generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

6.5.13 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 176](#), (for real axes).



1. In addition to the function block inputs, the jerk limit value set via parameter A-0-0216 influences the function block.

The behavior described in the following occurs if the jerk limit (A-0-0216 > 0) is active

2. During the acceleration phase, the acceleration input of the function block is always decisive.

If the axis velocity has to be reduced during the acceleration phase (e.g., overshooting of the actual velocity), the acceleration input is the limit value for the required delay.

In the following case, it can occur that the axis is decelerated with a larger delay than specified at the deceleration input:

- Jerk limit is active (A-0-0216 > 0) AND
- acceleration input > deceleration input AND
- Axis is in acceleration phase AND
- Motion restart with reduced positioning velocity

3. Without jerk limit (A-0-0216 = 0), the behavior does not occur

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
ML_PLCopen.library	Motion

Tab. 6-15: Library assignment

ML_PLCOpen.library

Interface description

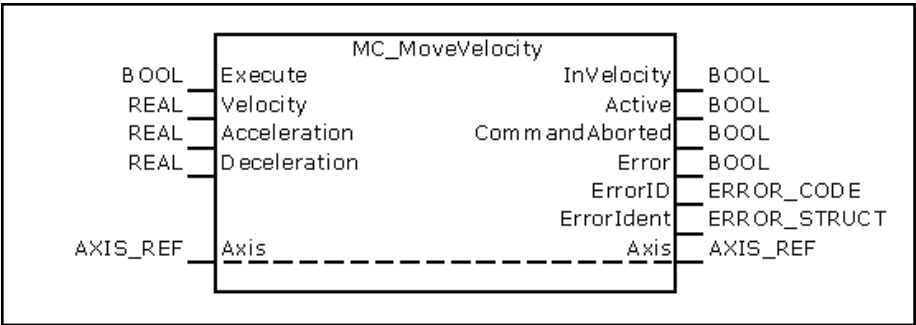


Fig. 6-19: MC_MoveVelocity function block

	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
	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

Tab. 6-16: 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

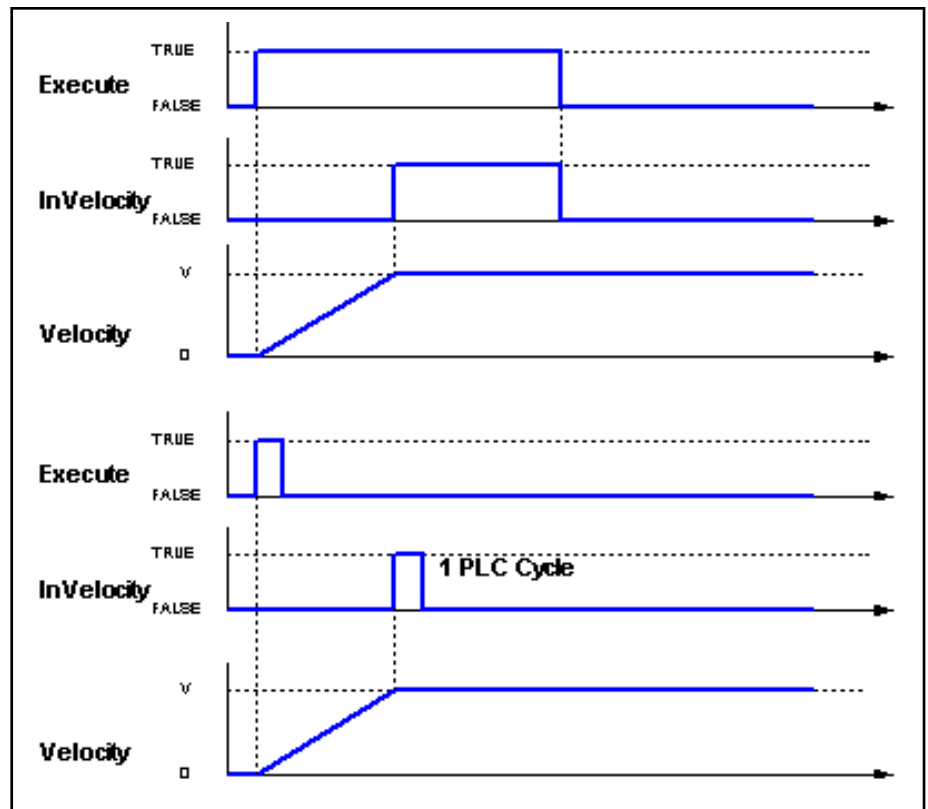


Fig. 6-20: Time diagram of MC_MoveVelocity (without aborted command)

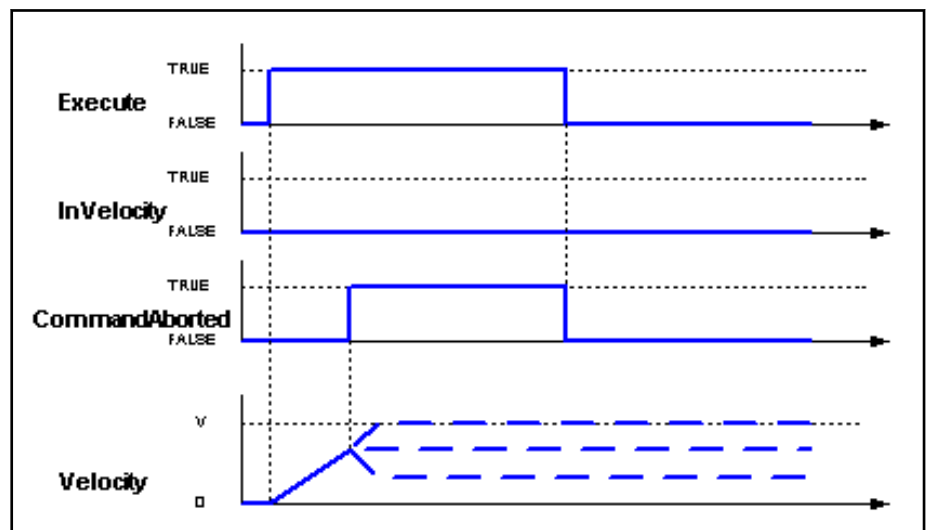


Fig. 6-21: Time diagram of MC_MoveVelocity (with aborted command)

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 has been completed. 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 desired velocity when the second function block is started. The following is shown: The first motion is restarted by

ML_PLCopen.library

"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. Now, the second function block is running and decelerates to a velocity of 2,000.

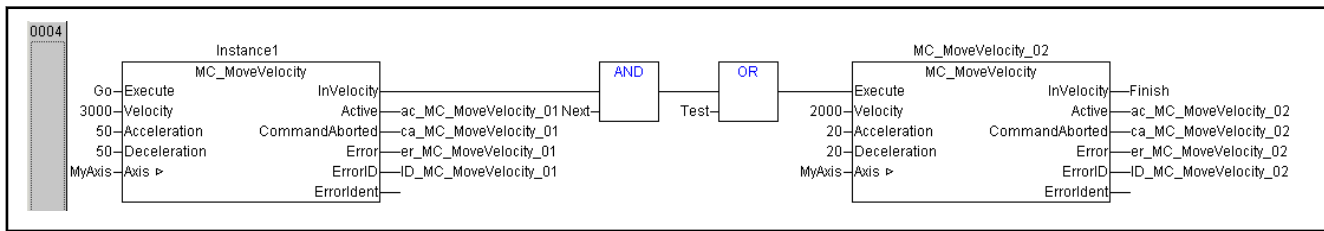


Fig. 6-22: Two instances of MC_MoveVelocity

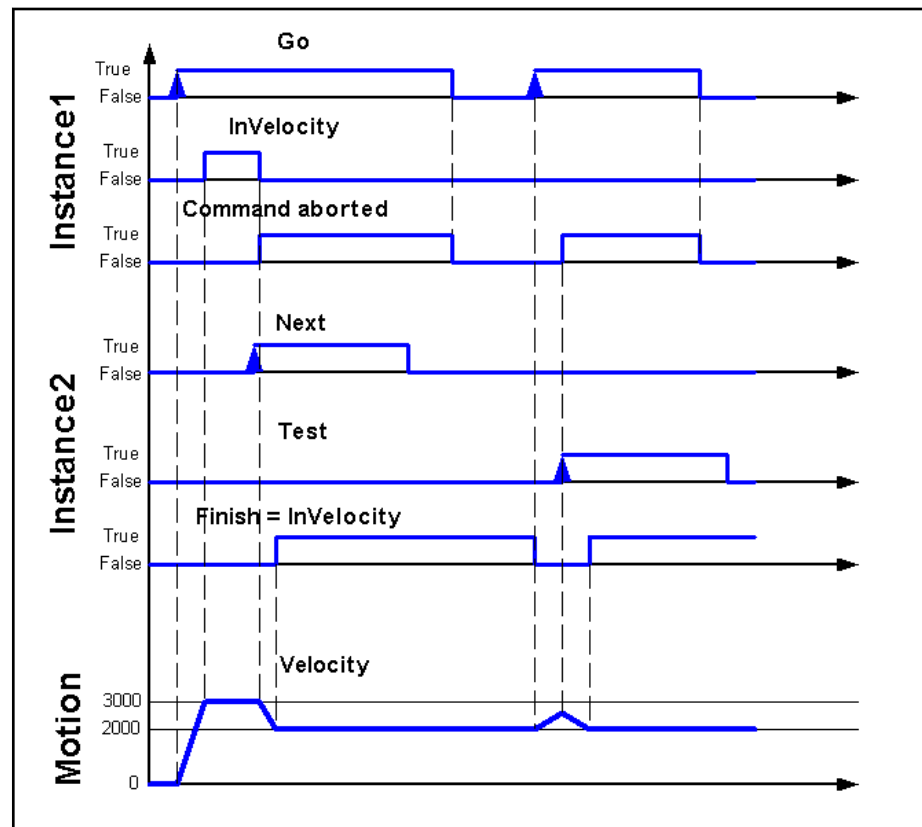


Fig. 6-23: Time diagram: Two instances of MC_MoveVelocity

6.5.14 MC_MoveVelocity, Special Features of SERCOS-Based Axes

Error handling: The function block MC_MoveVelocity generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

6.5.15 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 176](#), (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.

Library	Range
ML_PLCopen.library	Motion

Tab. 6-17: Library assignment

ML_PLCOpen.library

Interface description

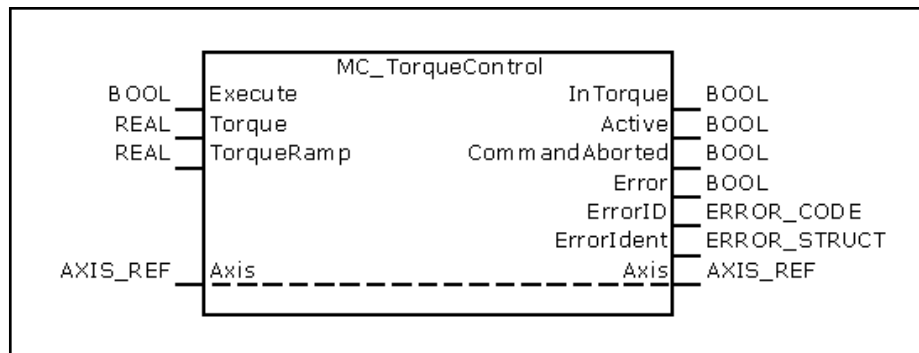
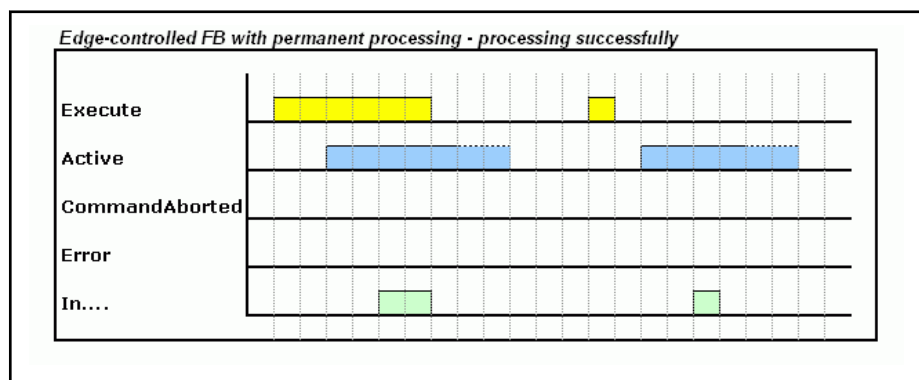


Fig. 6-24: MC_TorqueControl function block

	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
	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

Tab. 6-18: Interface of the MC_TorqueControl function block

Signal-time diagram



In.... InTorque

Fig. 6-25: Signal course of time of the MC_TorqueControl

6.5.16 MC_TorqueControl, special features of Sercos-based axes

Parameters used 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)

⁶⁾ The measuring units of time, position, velocity, acceleration, deceleration and power/torque depend on the effective scaling.

- S-0-0823, Ramp time (to be set before function block call)



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.



The functionality is supported by the following axis types:

- IndraDrive with interpolation in the drive
- IndraDrive with interpolation in the control
- SercosDrive
- Controller axis

Troubleshooting: The function block MC_TorqueControl generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

6.5.17 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.

Library	Range
ML_PLCopen.lib	Motion

Tab. 6-19: Library assignment

Interface

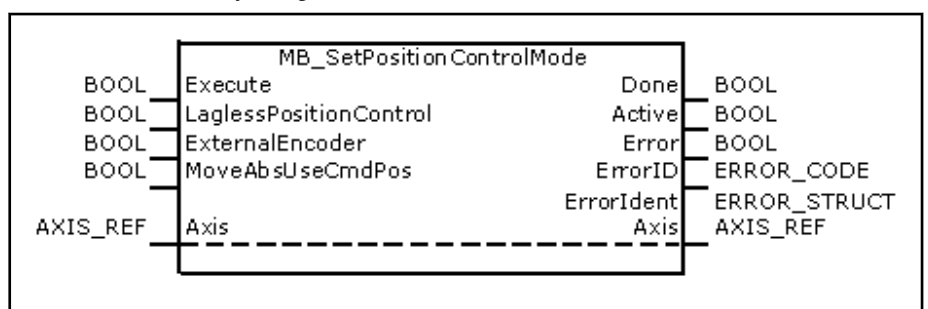


Fig. 6-26: MB_SetPositionControlMode

	Name	Data type	Description
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis.
VAR_INPUT	Execute	BOOL	Processing of the function block enabled (once, edge-controlled)

ML_PLCopen.library

	Name	Data type	Description
	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)
	MoveAbsUseCmd-Pos	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
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

Tab. 6-20: 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.

Error handling

The function block MB_SetPositionControlMode generates the following error messages:

- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 343 - error when accessing the axis/the control.

6.5.18 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.

Library	Range
ML_PLCopen.library	Motion

Tab. 6-21: Library assignment

Interface description

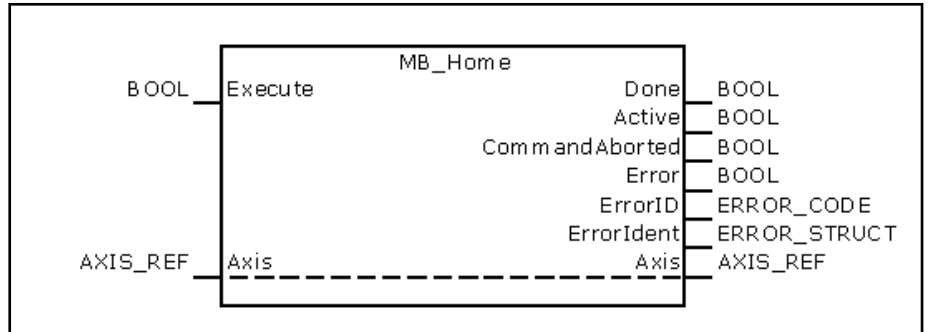


Fig. 6-27: MB_Home function block

	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. 6-22: Interface of the MB_Home function block

Signal-time diagram

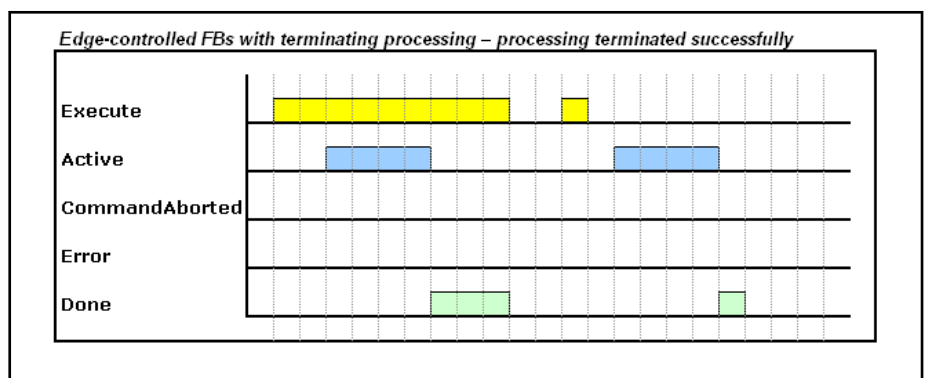


Fig. 6-28: Time diagram of MB_Home

6.5.19 MB_Home, special features of Sercos-based axes



The MB_Home function block may not be used together with virtual axes.

Troubleshooting:

The function block MB_Home generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

ML_PLCopen.library

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_MotionProfile 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 1 ms** can be set.

The following drives can be used:

Support IndraDrive

- MPx03,
- MPx04 (BA03),
- MPD,
- MPx05 (BA03),
- MPx06,
- MPx07,
- MPx08,
- MPx16,
- MPx17,
- MI.

Support sercosDrive (Pack Profile)

- Drives (DIAX02, 03, 04),

Support IndraWorks

- IndraDrive (Pack Profile),
- EcoCS (Pack Profile).

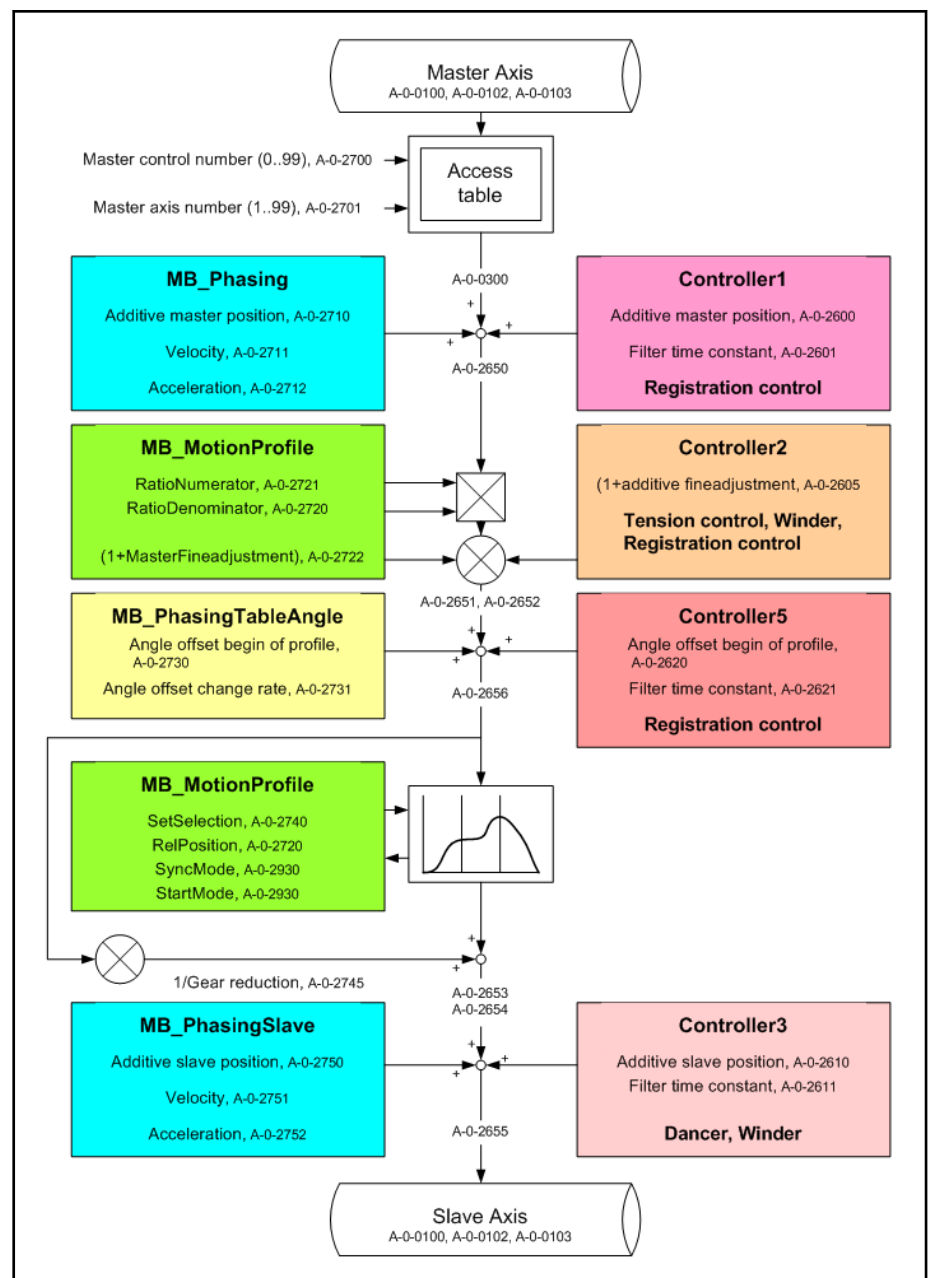


Fig. 6-29: Action chain MotionProfile as example

With an active MB_MotionProfil, it is possible to run the FBs 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 FBs are equipped with individual parameters and an individual interpolator.

For every parallel FB, 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 1st order filter with programmable time constant.

ML_PLCOpen.library

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
```

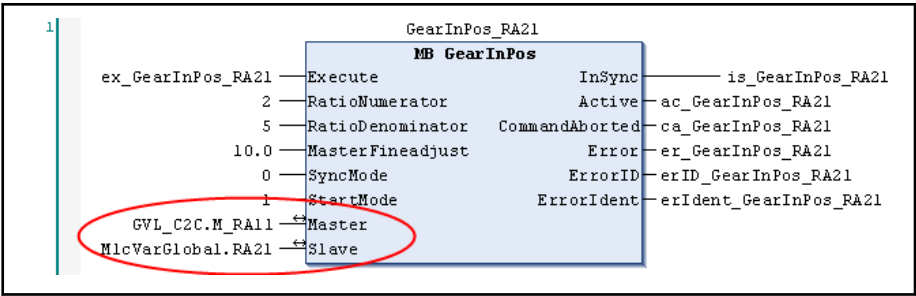


Fig. 6-30: Function blocks with master axis of removed control

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.

Library	Range
ML_PLCopen.library	Motion

Tab. 6-23: Library assignment

Interface Description

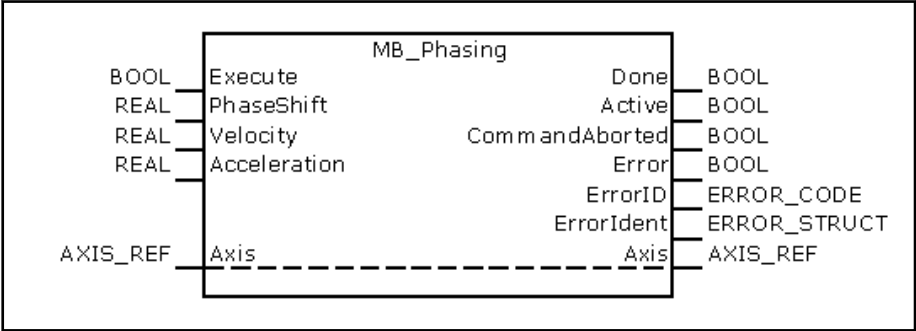


Fig. 6-31: FB MB_Phasing

ML_PLCOpen.library

	Name	Type	Comment ⁷⁾
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the slave axis
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 +).
VAR_OUTPUT	Done	BOOL	The position offset has successfully been traveled
	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. 6-24: Interface of the MB_Phasing FB

Signal-time diagram

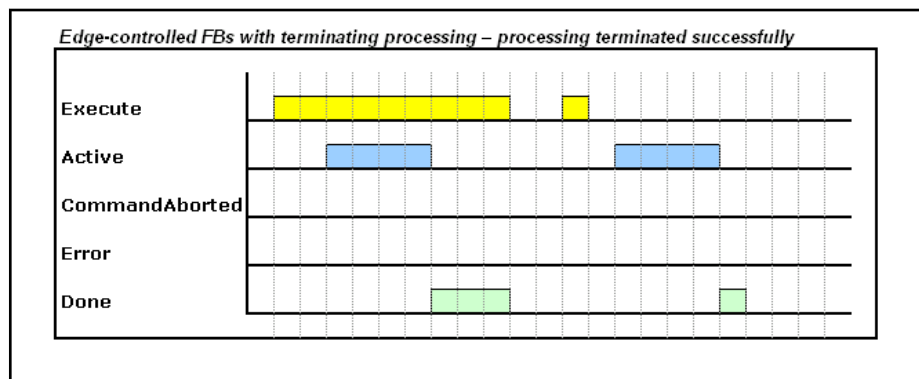


Fig. 6-32: Signal-time diagram

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,

⁷⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling of the master axis.

- A-0-2711, ELS, phasing, additive position command value, velocity, and
- 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.
- 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 that are exchanged via the sercos III control link, a rotary scaling with 360° is used as modulo value for calculation (the scaling information is not transmitted via the control link).

Error Handling

The function block MB_Phasing generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

6.6.3 MB_PhasingSlave

Brief Description

The function block MB_PhasingSlave realizes an adjustment of the slave axis with the specified velocity and acceleration.

Library	Area
ML_PLCopen.library	Motion

Tab. 6-25: Library assignment

Interface

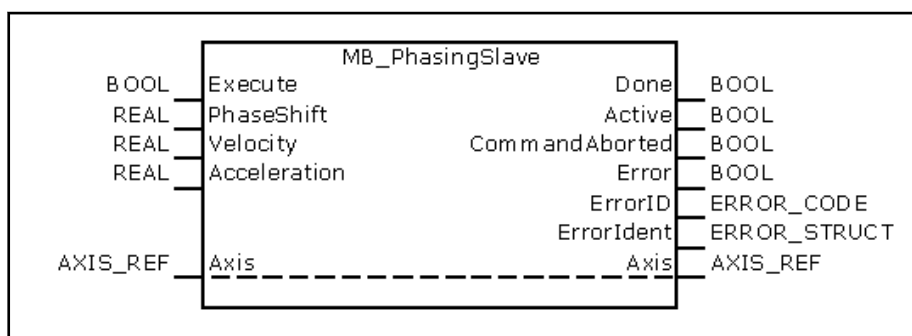


Fig. 6-33: MB_PhasingSlave FB

ML_PLCOpen.library

	Name	Type	Comment ⁸⁾
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	It 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	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command
	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

Tab. 6-26: Interface of the MB_PhasingSlave FB

Signal-time diagram

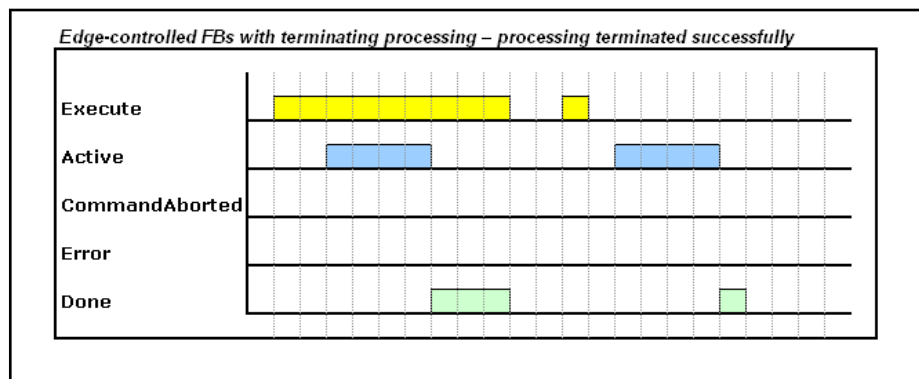


Fig. 6-34: Signal-time diagram

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.

The following control parameters are used:

- A-0-2750, ELS, PhasingSlave, additive position command value,

⁸⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling of the slave axis.

- A-0-2751, ELS, PhasingSlave, additive position command value, velocity, and
- A-0-2752, ELS, PhasingSlave, additive position command value, acceleration.

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.

Error handling

Der Funktionsbaustein MB_PhasingSlave erzeugt folgende Fehlermeldungen:

- [MLC_TABLE, ERROR_TABLE 16#0030 page 343](#) - 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.

Library	Range
ML_PLCOpen.library	Motion

Tab. 6-27: Library assignment

ML_PLCOpen.library

Interface description

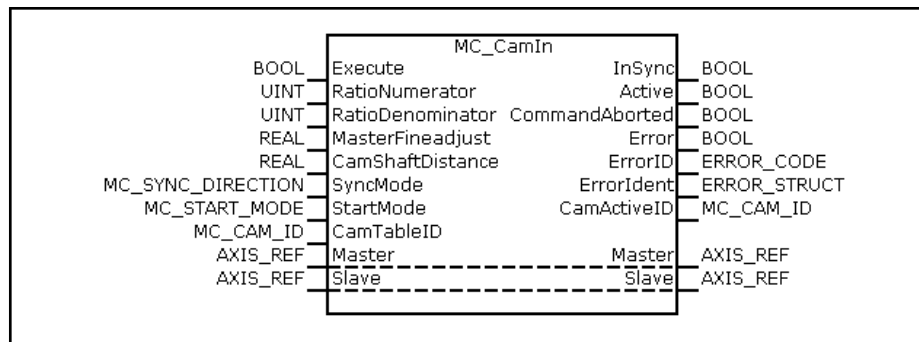
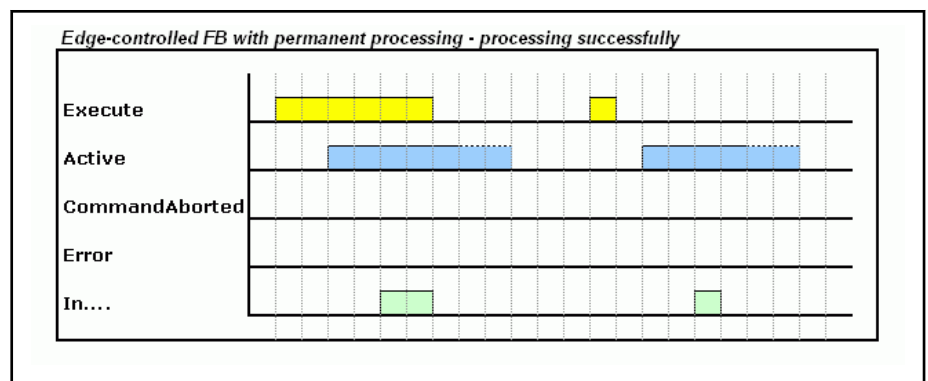


Fig. 6-35: FB MC_CamIn

	Name	Type	Comment
VAR_IN_OUT	Master	AXIS_REF	Contains information on the master axis
	Slave	AXIS_REF	It contains information on the slave axis which follows the master axis
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

Tab. 6-28: Interface of the MC_CamIn function block

Signal-time diagram



in... InSync

Fig. 6-36: Signal-time diagram

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 realize a relationship 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 in the cam table.



Cam table functionality is supported by the following axis types (slave axis):

- IndraDrive with interpolation in the drive

ML_PLCOpen.library

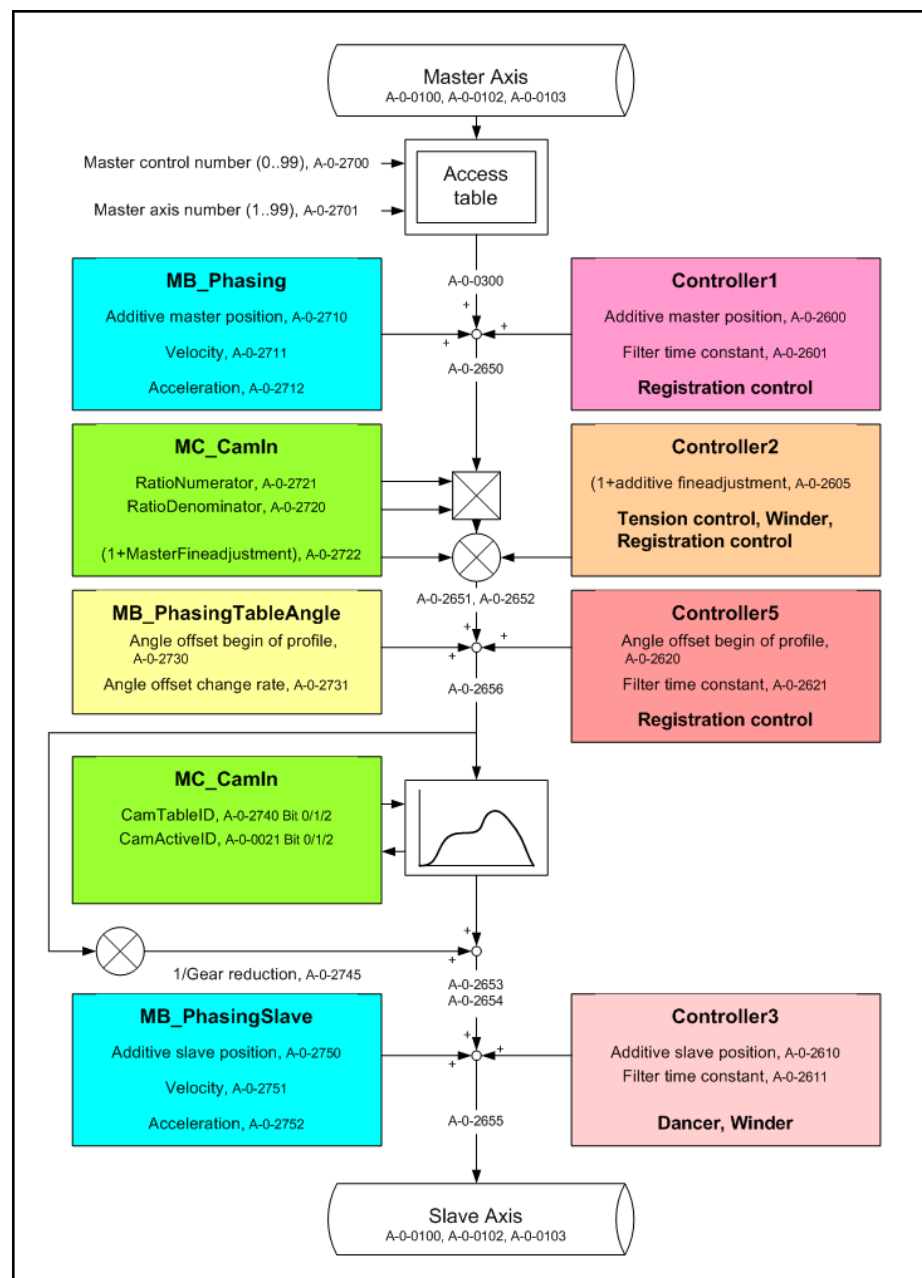


Fig. 6-37: MotionControl action chain MC_CamIn

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,
- A-0-2722, ELS, gear ratio fine adjustment. 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
- Scaling for the slave axis: A-0-2741, ELS, CamIn, cam distance

Based on the incoming position and velocity signal, the function block can implement the synchronization before master and slave axis engage.

Troubleshooting

The function block MC_CamIn generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

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.

Library	Range
ML_PLCLopen.library	Motion

Tab. 6-29: Library assignment

Interface description

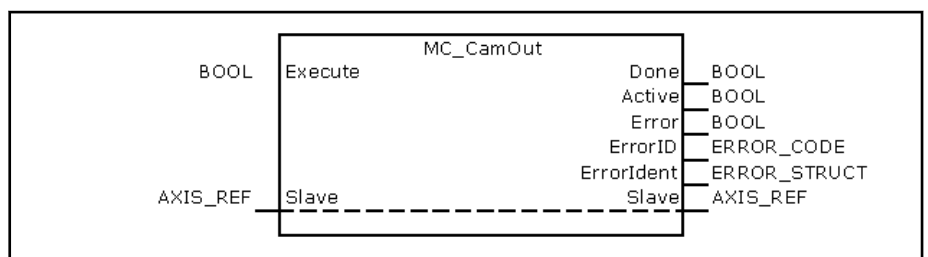


Fig. 6-38: FB MC_CamOut

	Name	Type	Comment
VAR_IN_OUT	Slave	AXIS_REF	Contains information regarding the slave axis which follows the master axis
VAR_INPUT	Execute	BOOL	Starts the action if there is a rising edge
VAR_OUTPUT	Done	BOOL	Decoupling completed successfully
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred in the FB instance.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error

Tab. 6-30: Interface of FB MC_CamOut

Signal-time diagram

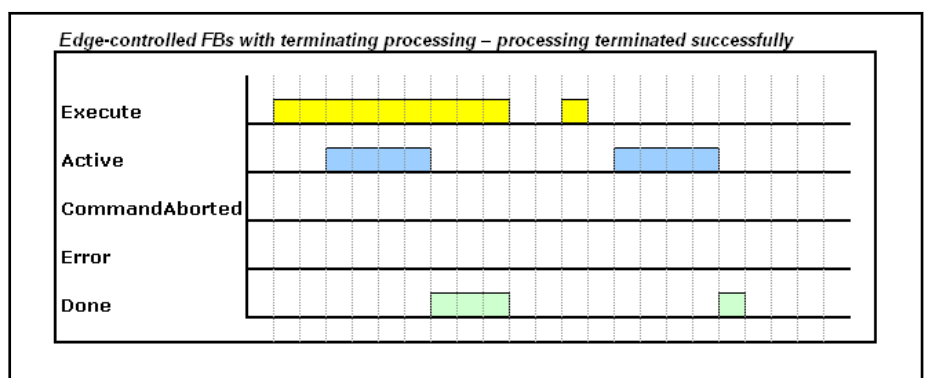


Fig. 6-39: Signal time diagram

6.6.7 MC_CamOut, Special Features of SERCOS-based Axes

Error Handling The function block MC_CamOut generates the following error messages:

ML_PLCopen.library

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

6.6.8 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 176](#), (for real axes).



The function block is deactivated by the block [MC_GearOut, page 243](#),.



MotionProfile functionality is supported by the following axis types (slave axis):

- IndraDrive with interpolation in the control
- IndraDrive with interpolation in the control
- SercosDrive
- Virtual axis
- Controller axis

Library	Range
ML_PLCopen.library	Motion

Tab. 6-31: Library assignment

Interface description

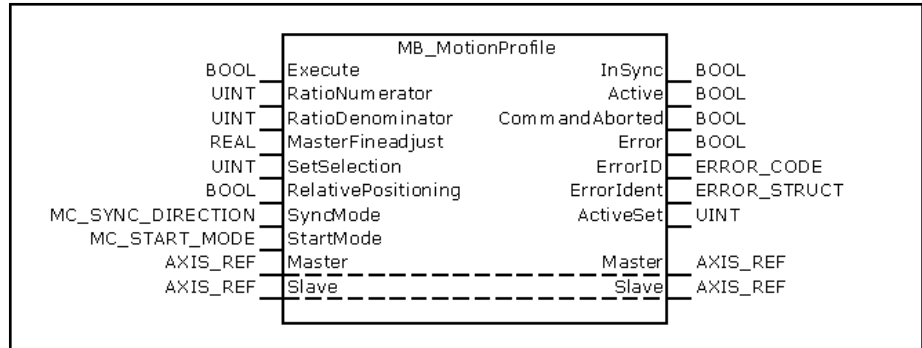


Fig. 6-40: MB_MotionProfile function block

	Name	Type	Comment
VAR_IN_OUT	Master	AXIS_REF	Master axis selection
	Slave	AXIS_REF	Selection of the slave axis following the master axis
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

	Name	Type	Comment
	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: Phase offset 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

Tab. 6-32: 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.

ML_PLCopen.library

The cam tables are stored on the control under the C-parameters C-0-2001...2099. The function block [MB_ChangeCamData](#), page 223, allows for filling these parameters with data from the PLC.

- 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.

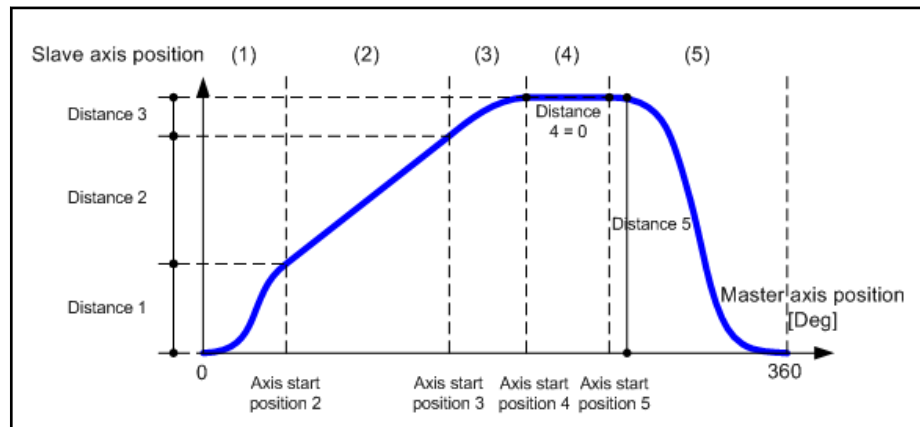


Fig. 6-41: Example of a MotionProfile

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,

6.6.9 MC_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
- IndraDrive 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 realize a relationship 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 in the MotionProfile.

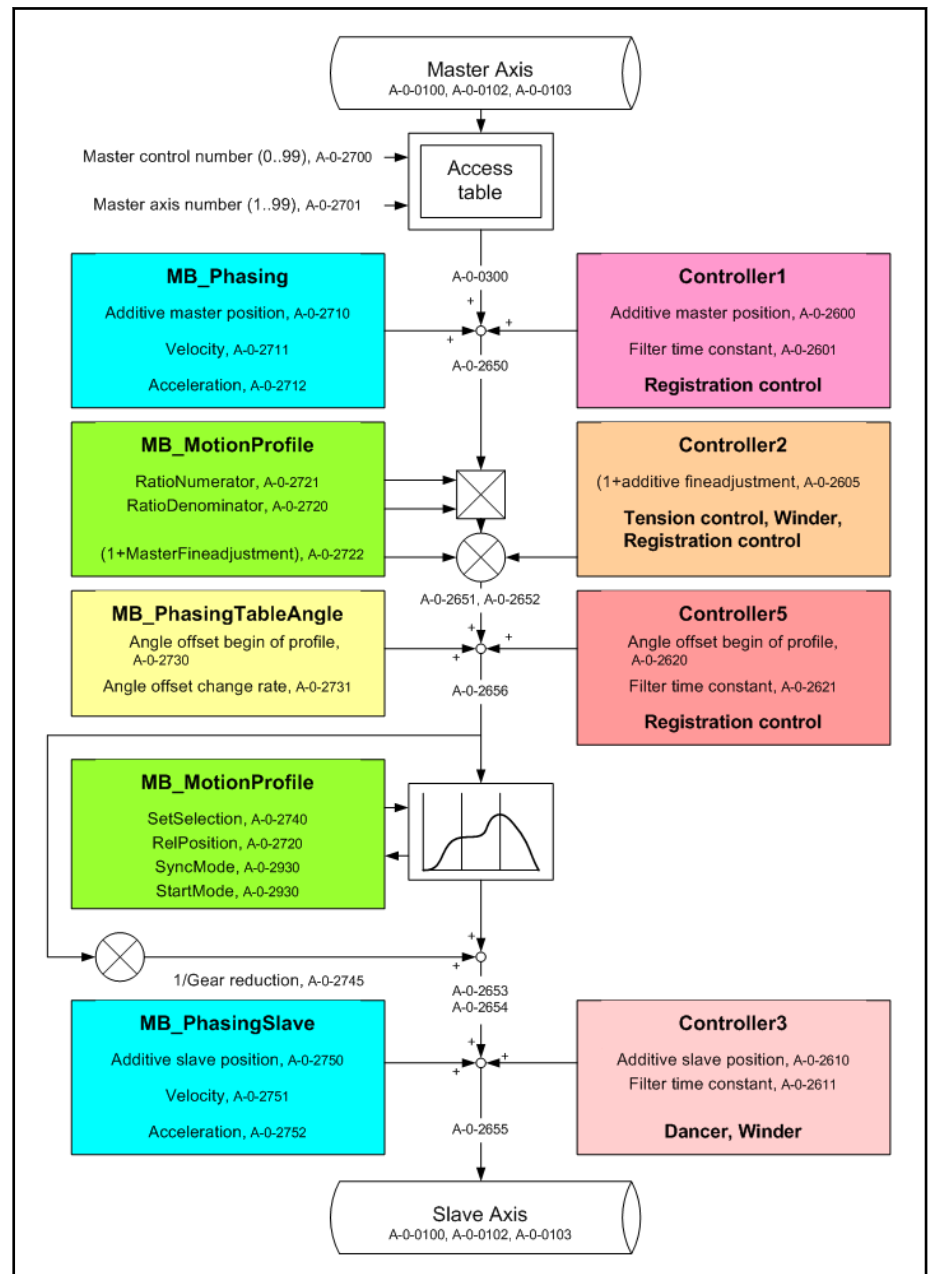


Fig. 6-42: MotionControl action chain MB_MotionProfile

ML_PLCopen.library

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",
- "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 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-0-2902, MotionProfile 0, Diagnostics
- A-0-2903, MotionProfile 1, Diagnostics
- A-0-2904, MotionProfile, basic configuration

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

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.

Troubleshooting The function block MB_MotionProfile generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

6.6.10 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.

The cam tables can be configured either via the PLC function block [MB_ChangeCamData](#), page 223, or via the IndraWorks interface.



MotionProfile functionality is supported by the following axis types (slave axis):

- IndraDrive with interpolation in the control
- IndraDrive with interpolation in the control
- SercosDrive
- Virtual axis
- Controller axis

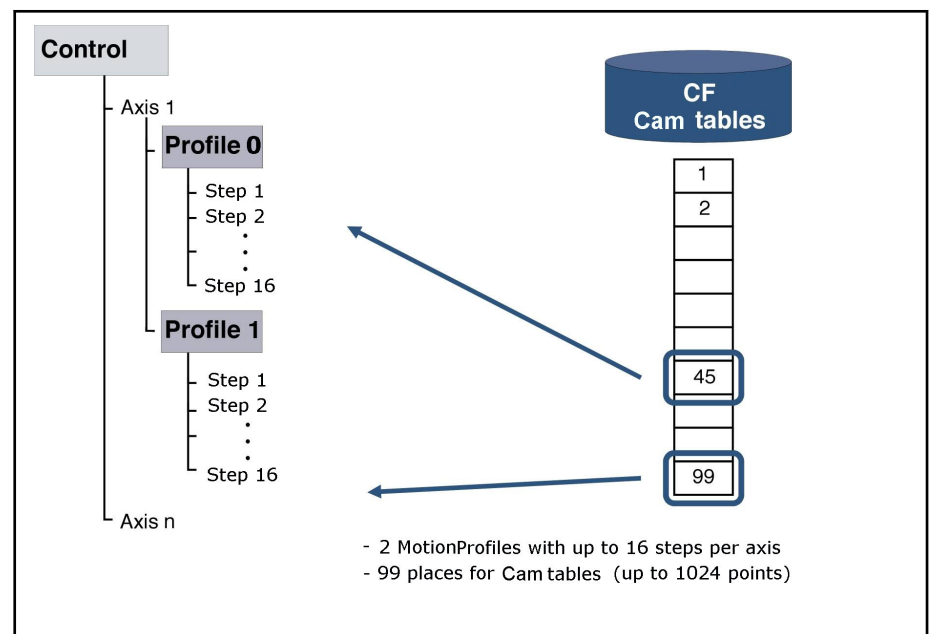


Fig. 6-43: Assigning cam tables/MotionProfile

- 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 216: Writing a complete MotionProfile, from the PLC to the Motion
- [MB_ChangeProfileStep](#), page 219: Replacing a step of the MotionProfile,
- [MB_ChangeCamData](#), page 223,: Transferring a cam table from the PLC memory to the Motion, C-0-2001...2099.

9) depending on the type of

ML_PLCLopen.library

MB_ChangeProfileSet

Brief description The MC_ChangeProfileSet function block allows a complete motion profile with a maximum of 16 steps to be described and an already existing one to be exchanged.

Library	Range
ML_PLCLopen.library	Motion

Tab. 6-33: Library assignment

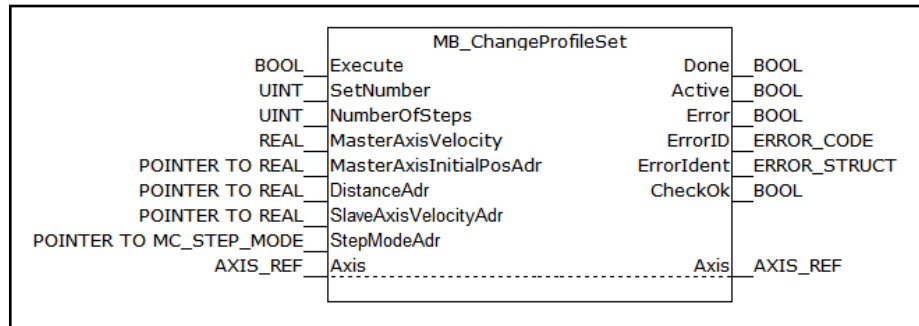
Interface description

Fig. 6-44: FB MB_ChangeProfileSet

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the axis for which the motion profile set is to be exchanged.
VAR_INPUT	Execute	BOOL	Starts the action if there is a rising edge
	SetNumber	UINT	Selection of MotionProfile (0 or 1)
	Number of Steps	UINT	Number of steps of the motion profile (1...16)
	MasterAxis-Velocity	REAL	Master axis velocity for the profile
	MasterAxis-InitialPosAdr	POINTER TO REAL	Pointer to a list of master axis positions at the beginning of the respective step
	DistanceAdr	POINTER TO REAL	Pointer to a list of strokes at the end of the respective step
	SlaveAxis-VelocityAdr	POINTER TO REAL	Pointer to a list of slave axis velocities at the end of the respective 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	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred in the FB instance
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error
	CheckOk	BOOL	Reserved

Tab. 6-34: Interface of the FB MB_ChangeProfileSet

Signal-time diagram

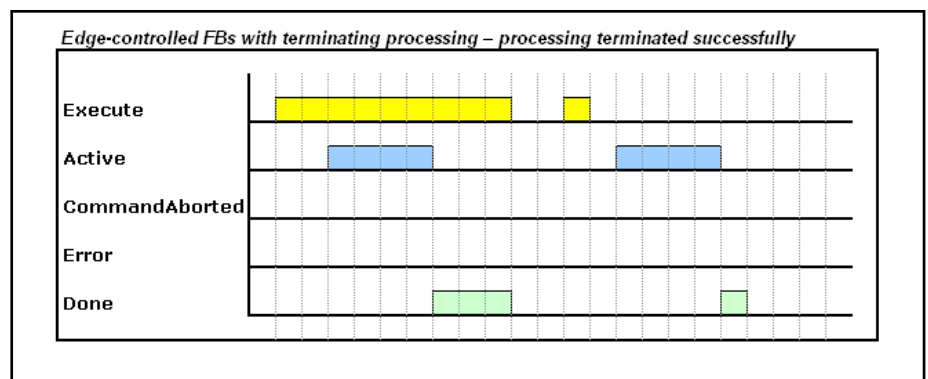


Fig. 6-45: Signal time diagram: MB_ChangeProfileSet

Functional description

The function block allows for transferring a complete motion profile with 1...16 steps 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).

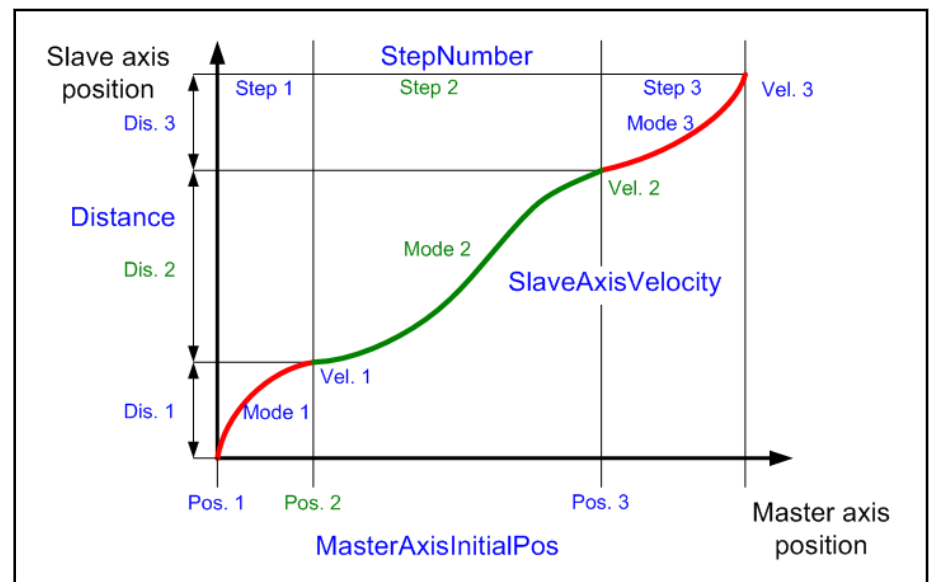


Fig. 6-46: Data sets of the individual steps of a motion profile

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.

ML_PLCopen.library

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" (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" with 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" with MB_WriteParameter (value as DINT),
- Reading parameter "A-0-2931, MotionProfile, status word" with MB_ReadParameter (value as DINT),
- DINT_TO_DWORD,
- **Bit 12/13:**

Bit 12 indicates whether the MotionProfile in set 0 has been checked.

Then, the result of the test has to be taken from bit 13. For more detailed information on a negative test result "F2229235, Error in MotionProfile", please refer to "A-0-2902, MotionProfile 0, diagnostics".

Bit 14/15:

Bit 14 indicates whether the MotionProfile in set 1 has been checked.

Then, the result of the test has to be taken from bit 15. For more detailed information on a negative test result "F2229235, Error in MotionProfile", please refer to "A-0-2903, MotionProfile 1, diagnostics".

**Read status to control whether the
complete profile was tested.****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

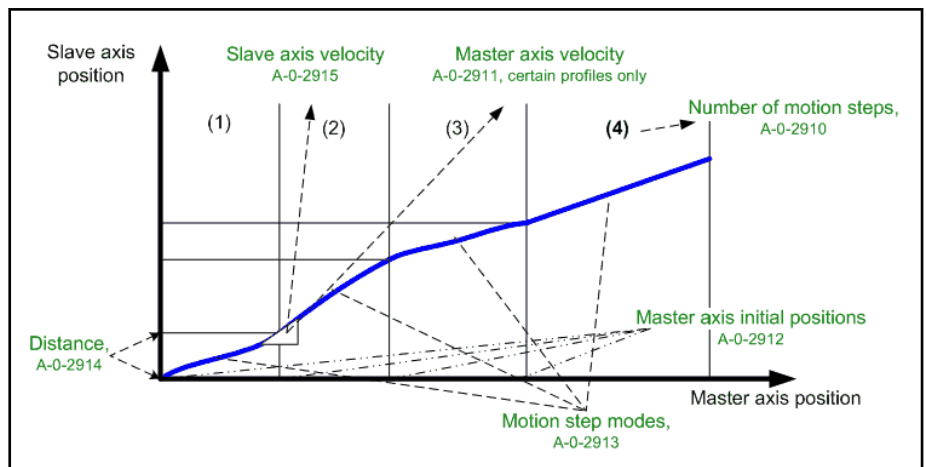


Fig. 6-47: Overview on the parameters of the MotionProfile 0

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

Error handling

The function block MB_ChangeProfileSet generates the following types of error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

MB_ChangeProfileStep

Brief description

The function block MB_ChangeProfileStep changes the four parameters which define a step of a motion profile set.

ML_PLCopen.library

Library	Range
ML_PLCopen.library	Motion

Tab. 6-35: Library assignment

Interface description

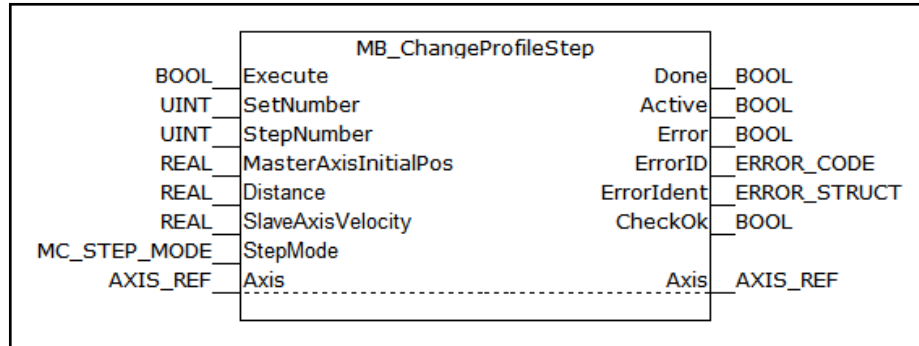


Fig. 6-48: FB MB_ChangeProfileStep

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the axis the motion profile set of which has to be changed.
VAR_INPUT	Execute	BOOL	Starts the action if there is 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	Master axis start position 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	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred in the FB instance
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error
	CheckOk	BOOL	Reserved

Tab. 6-36: Interface of the FB MB_ChangeProfileStep

Signal-time diagram

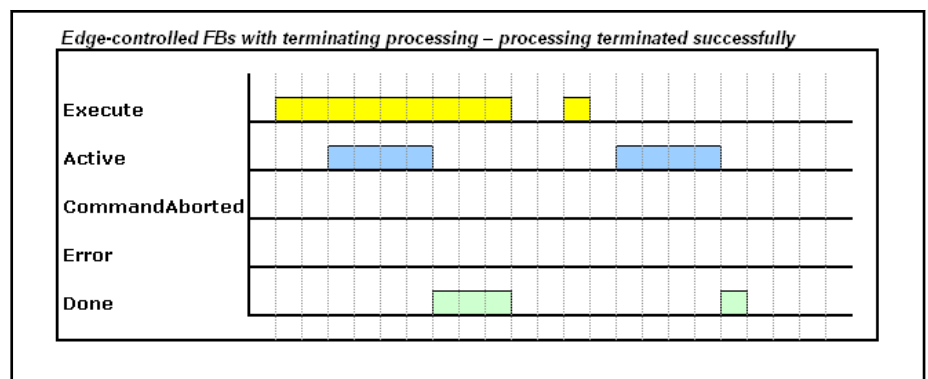


Fig. 6-49: Signal time diagram: MB_ChangeProfileStep

Functional description

The block replaces one step with another.

The input values of the block are the number of the motion profile (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), stroke (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 weighting parameters of the axis.

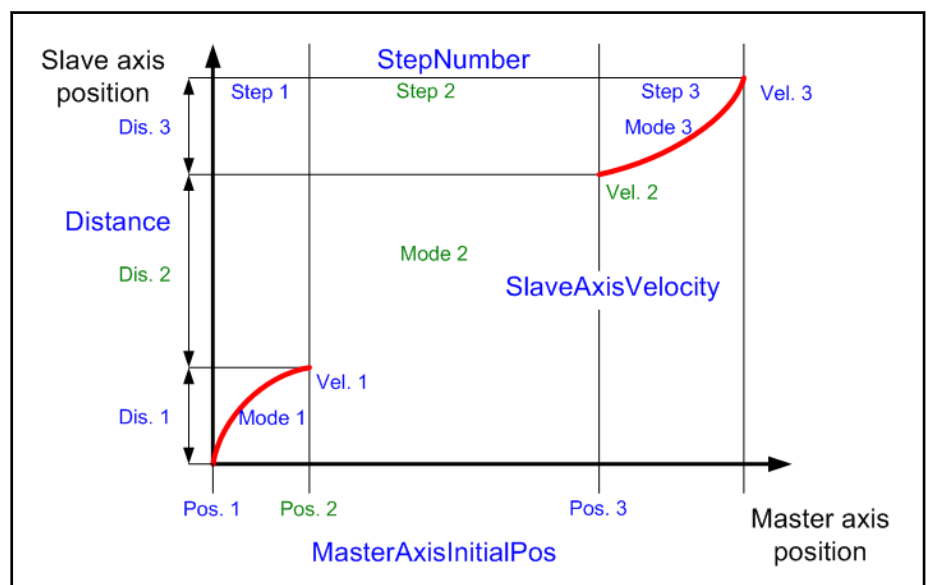


Fig. 6-50: Motion profile with 3 steps, step 2 is modified

ML_PLCopen.library

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).

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.

Example for the usage of the MB_ChangeProfileStep for two (or more) steps in the profile

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" with 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" with MB_WriteParameter (value as DINT),
2. Exchange of the first step
 - MB_ChangeProfileStep for the first data set (carried out without test).

The change of status becomes effective in 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" with 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" with MB_WriteParameter (value as DINT),
5. Read status to control whether the complete profile was tested.
 - Read parameter "A-0-2931, MotionProfile, status word" with MB_ReadParameter (value as DINT),
 - DINT_TO_DWORD,
 - **Bit 12/13:**

Bit 12 indicates whether the MotionProfile in set 0 has been checked.

Then, the result of the test has to be taken from bit 13. For more detailed information on a negative test result "F2229235, Error in

MotionProfile", please refer to "A-0-2902, MotionProfile 0, diagnostics".

Bit 14/15:

Bit 14 indicates whether the MotionProfile in set 1 has been checked.

Then, the result of the test has to be taken from bit 15. For more detailed information on a negative test result "F2229235, Error in MotionProfile", please refer to "A-0-2903, MotionProfile 1, diagnostics".

Error handling The function block MB_ChangeProfileStep generates the following types of error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

MB_ChangeCamData

Brief description

The function block MB_ChangeCamData supports the loading of cam table data in real time conditions which were saved and/or processed on the PLC side to the cam parameters, C-0-2001 ... 2099 (Motion side).

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 make sure that the information to be transferred are of a correct quantity of data and the right type of data (DINT).

Library	Range
ML_PLCopen.library	Motion

Tab. 6-37: Library assignment

Interface description

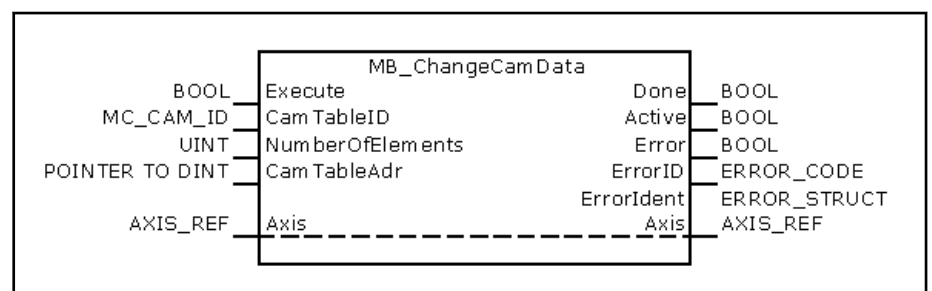


Fig. 6-51: Function block MB_ChangeCamData

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the slave axis
VAR_INPUT	Execute	BOOL	Starts the action if there is a rising edge
	CamTableID	MC_CAM_ID	Number of the cam table to be written 1..99 (C-0-2001...2099).
	NumberO-fElements	UINT	Number of bases 3..1,024 in the desired cam table

ML_PLCOpen.library

	Name	Type	Comment
	CamTableAdr	POINTER TO DINT	Pointer points to the PLC memory range containing the data to be transported.
VAR_OUTPUT	Done	BOOL	Cam table data has been transferred
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred in the FB instance
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error

Tab. 6-38: Interface of the FB MB_ChangeCamData

Signal-time diagram

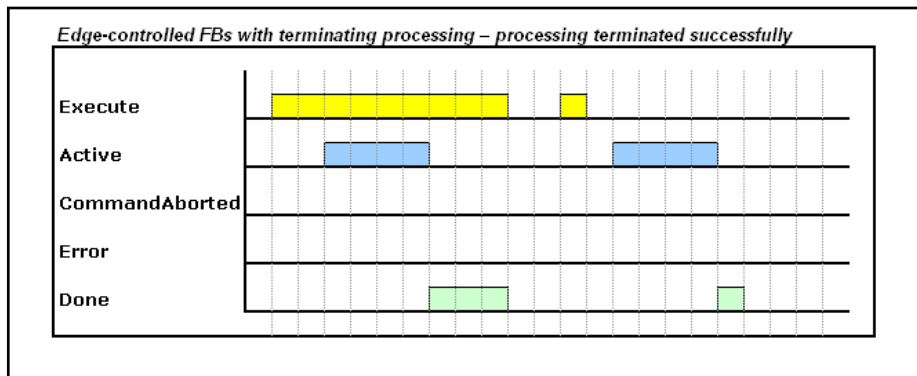


Fig. 6-52: Signal time diagram: MB_ChangeCamData

MB_ChangeCamData, Special Features of SERCOS-based Axes

Error handling The function block MB_ChangeCamData generates the following types of error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

6.6.11 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 176](#), (for real axes).



The function block is deactivated by the block [MC_GearOut, page 243](#).



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

Library	Range
ML_PLCopen.library	Motion

Tab. 6-39: Library assignment

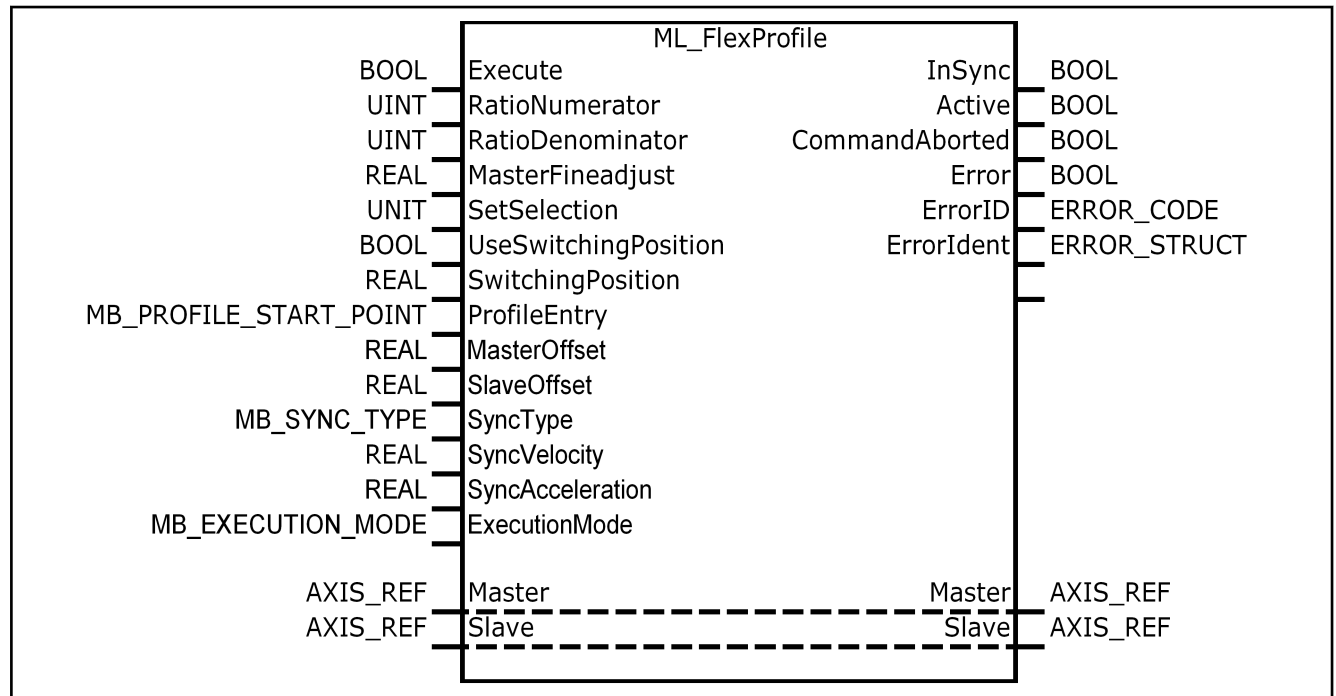


Fig. 6-53: ML_FlexProfile function block

Interface description

	Name	Type	Comment
VAR_IN_OUT	Master	AXIS_REF	Master axis selection
	Slave	AXIS_REF	Selection of the slave axis following the master axis
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.

ML_PLCOpen.library

	Name	Type	Comment
	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
	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

Tab. 6-40: 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](#) function block allows for filling these parameters with data from the PLC.

- 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.

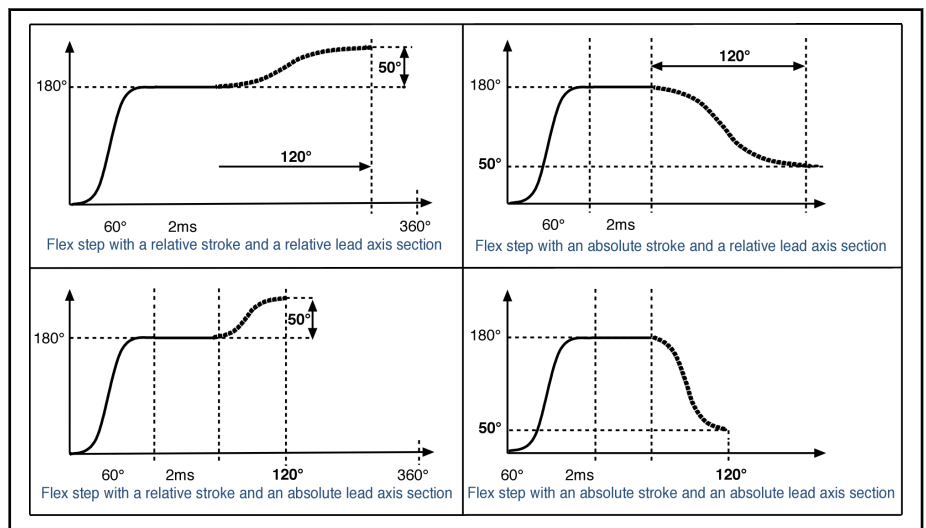


Fig. 6-54: Example of a FlexProfile

6.6.12 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,

ML_PLCOpen.library

- 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 realize a relationship 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 and store this relationship in the FlexProfile.

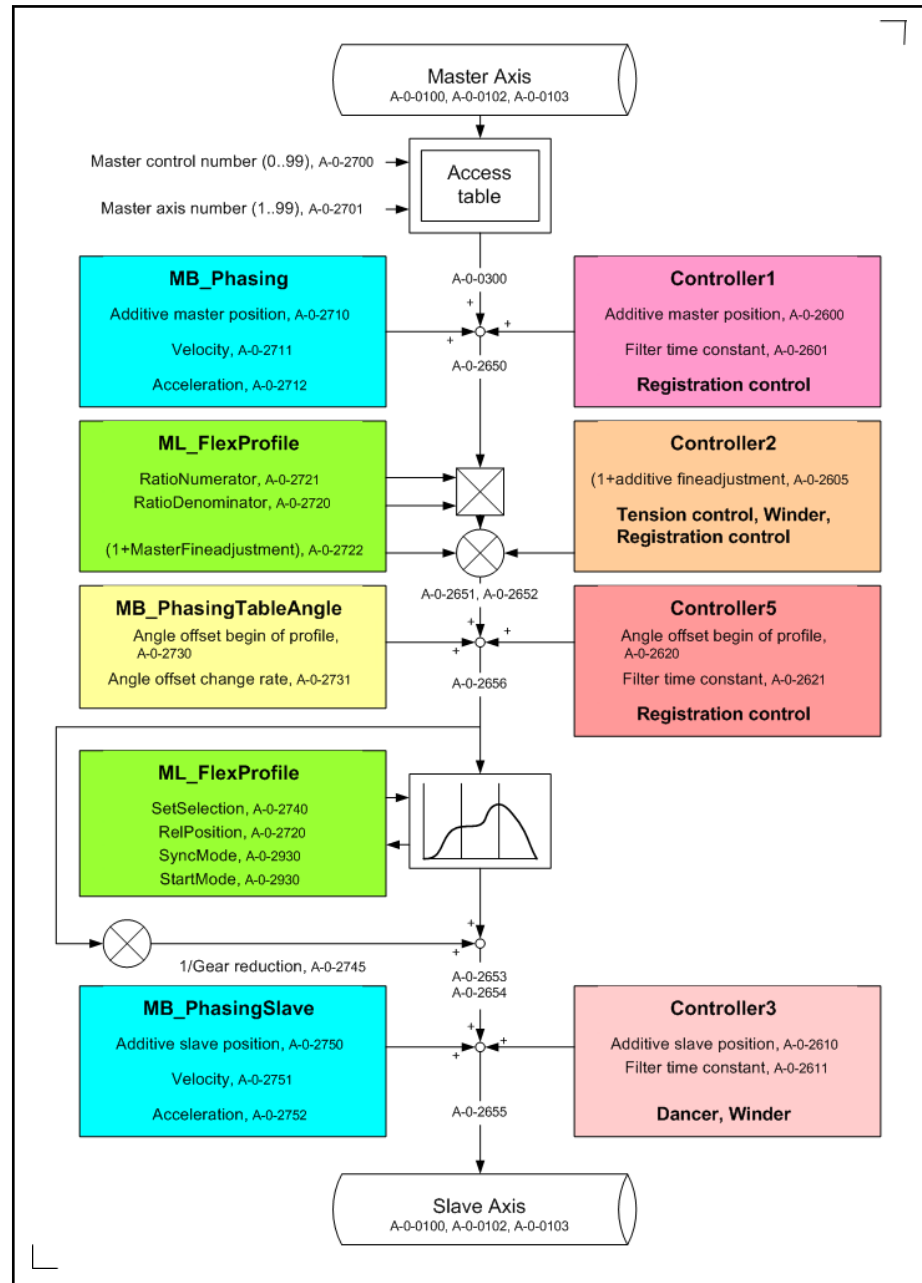


Fig. 6-55: MotionControl action chain ML_FlexProfile

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",
- "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 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 position command 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.

Parameter name	FlexProfile 0	FlexProfile 1	FlexProfile 2	FlexProfile 3
Master axis velocity	A-0-3020	A-0-3050	A-0-3080	A-0-3110
List of master axis sources	A-0-3021	A-0-3051	A-0-3081	A-0-3111
List of slave axis distances	A-0-3022	A-0-3052	A-0-3082	A-0-3112
List of master axis ranges	A-0-3023	A-0-3053	A-0-3083	A-0-3113
List of master axis ranges, units	A-0-3024	A-0-3054	A-0-3084	A-0-3114
List of motion laws	A-0-3025	A-0-3055	A-0-3085	A-0-3115
List of motion step types	A-0-3026	A-0-3056	A-0-3086	A-0-3116
List of slave axis starting velocities	A-0-3027	A-0-3057	A-0-3087	A-0-3117
List of slave axis starting accelerations	A-0-3028	A-0-3058	A-0-3088	A-0-3118
List of slave axis starting jerks	A-0-3029	A-0-3059	A-0-3089	A-0-3119
List of slave axis end velocities	A-0-3030	A-0-3060	A-0-3090	A-0-3120
List of slave axis end accelerations	A-0-3031	A-0-3061	A-0-3091	A-0-3121
List of slave axis end jerks	A-0-3032	A-0-3062	A-0-3092	A-0-3122
List of slave axis travel velocities	A-0-3033	A-0-3063	A-0-3093	A-0-3123
List of slave axis travel accelerations	A-0-3034	A-0-3064	A-0-3094	A-0-3124
List of slave axis travel jerks	A-0-3035	A-0-3065	A-0-3095	A-0-3125
List of turning point displacements	A-0-3036	A-0-3066	A-0-3096	A-0-3126
Events, list of motion step assignments	A-0-3040	A-0-3070	A-0-3100	A-0-3130
Events, list of trigger modes	A-0-3041	A-0-3071	A-0-3101	A-0-3131
Events, list of action modes	A-0-3042	A-0-3072	A-0-3102	A-0-3132
Events, list of trigger values	A-0-3043	A-0-3073	A-0-3103	A-0-3133
Events, list of trigger values, units	A-0-3044	A-0-3074	A-0-3104	A-0-3134
Command: Check and apply event	A-0-3046	A-0-3076	A-0-3106	A-0-3136

ML_PLCopen.library

Parameter name	FlexProfile 0	FlexProfile 1	FlexProfile 2	FlexProfile 3
Diagnostics	A-0-3047	A-0-3077	A-0-3107	A-0-3137
Command: Check and apply profile	A-0-3048	A-0-3078	A-0-3108	A-0-3138
Checksum	A-0-3049	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.

Troubleshooting The function block ML_FlexProfile generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

6.6.13 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.

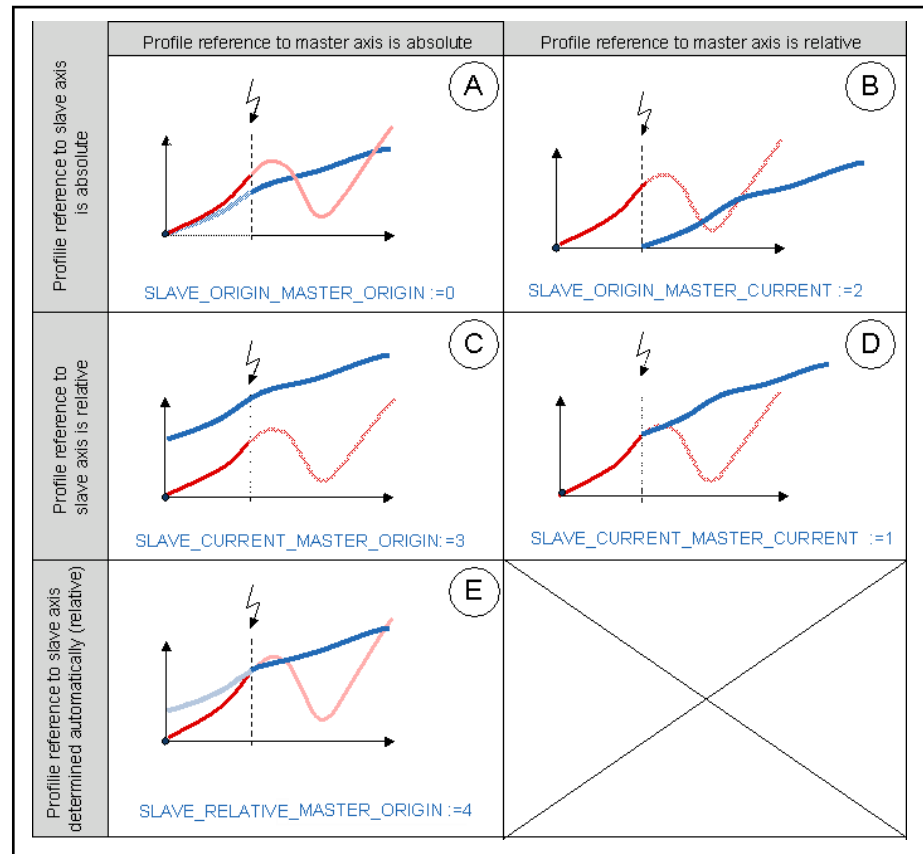


Fig. 6-56: Selecting the coordinate system

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.

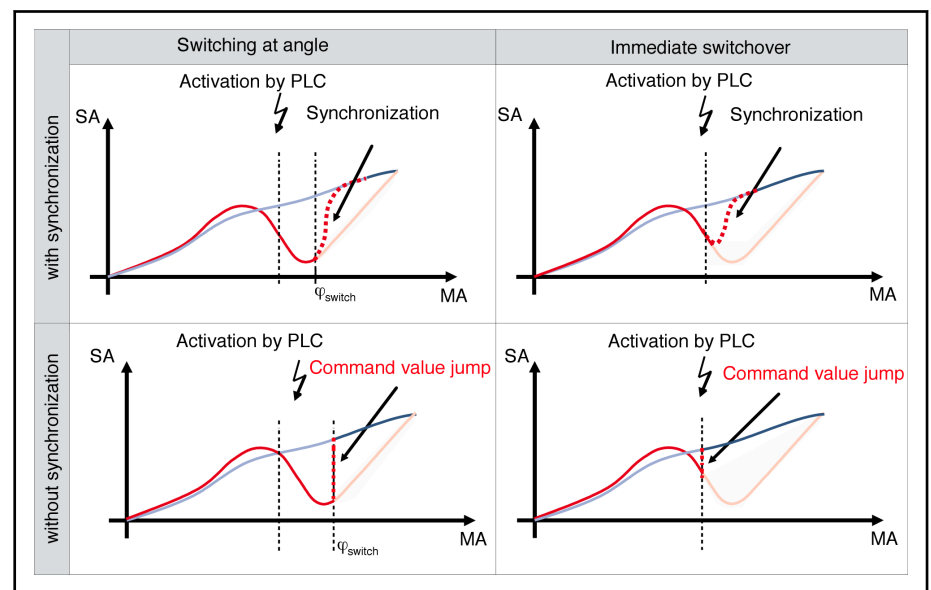


Fig. 6-57: Case A - Switching and synchronizing with absolute master axis position and absolute slave axis position

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).

ML_PLCOpen.library

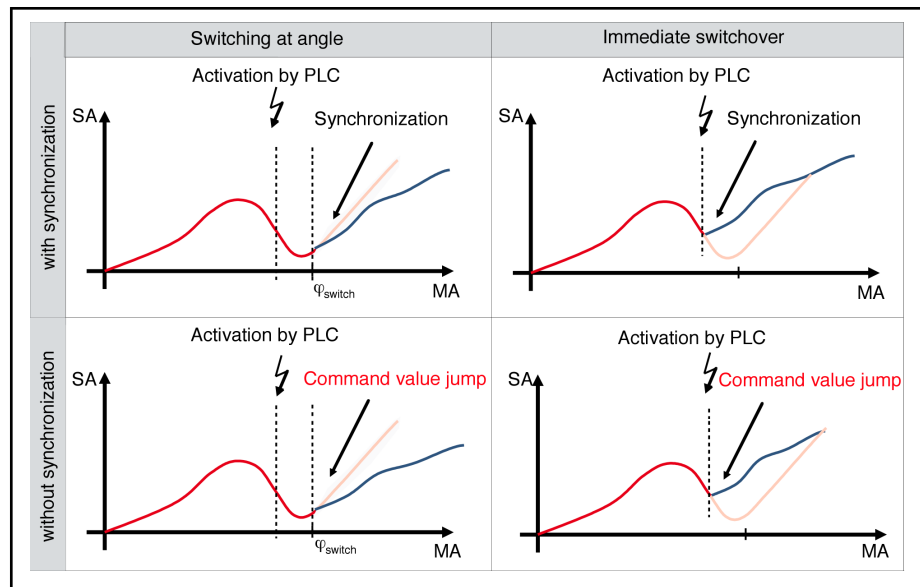


Fig. 6-58: Case D - Switching and synchronizing with relative master axis position and relative slave axis position -

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.

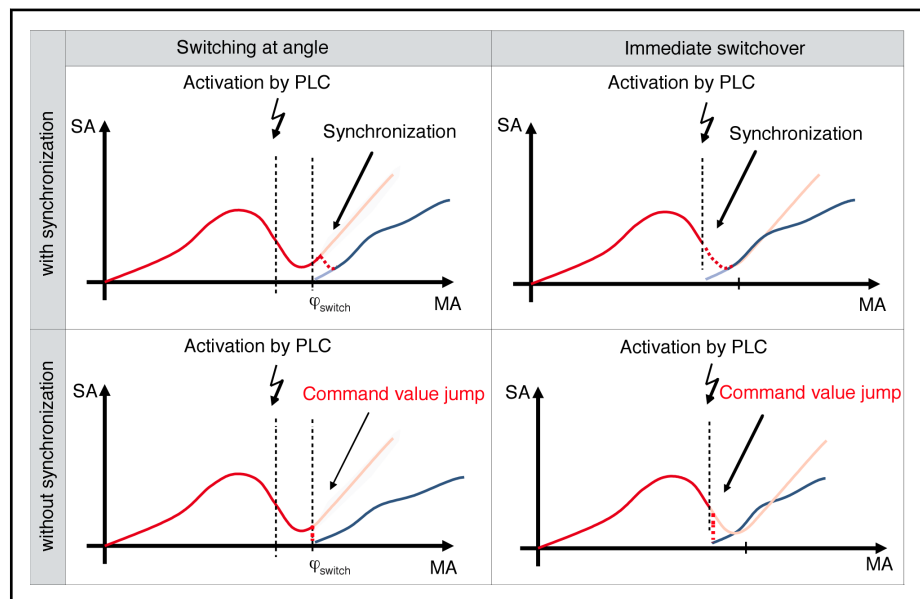


Fig. 6-59: Case B - Switching and synchronizing with relative master axis position and absolute slave axis position

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.

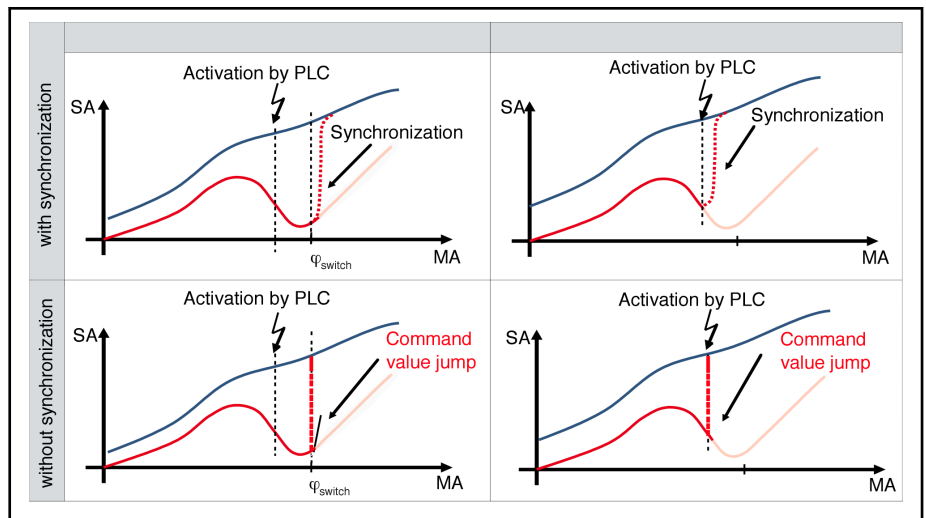


Fig. 6-60: Case C - Switching and synchronizing with absolute master axis position and relative slave axis position

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.

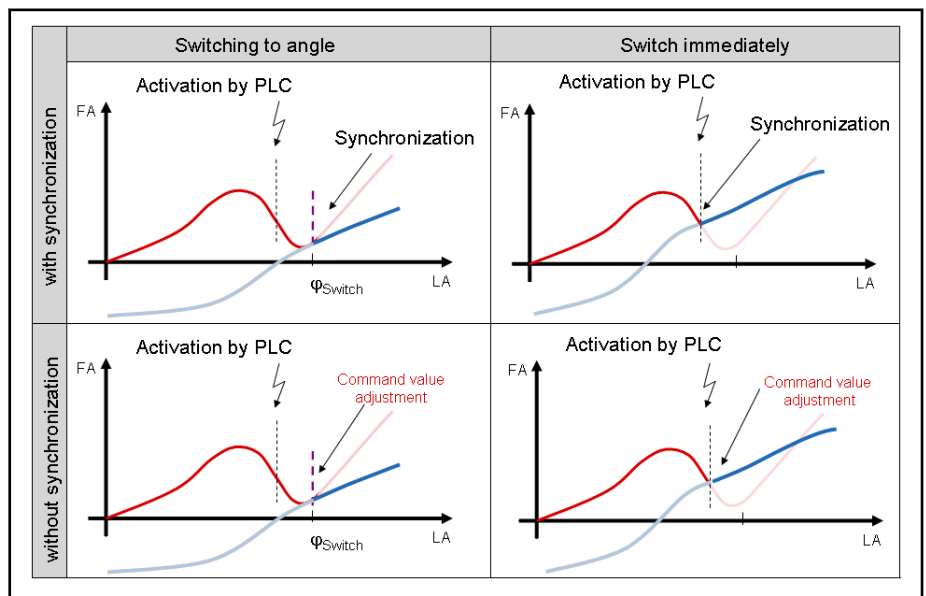


Fig. 6-61: Case E - Profile reference to slave axis is determined automatically (relative)

Processing Types

There are two processing types (see also MB_EXECUTION_MODE, ML_Base.library):

- Continuous processing
- One-time processing (one shot)

ML_PLCopen.library

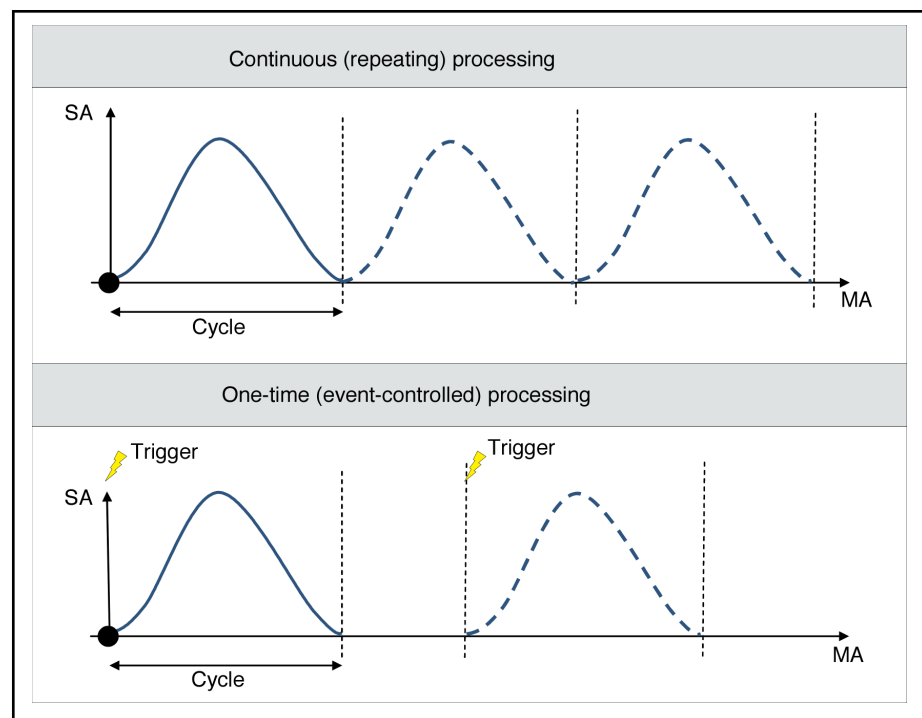


Fig. 6-62: Continuous processing and one-time processing

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.

6.6.14 PLCopen Function Blocks for Supporting FlexProfiles

General information

An XLC/MLC can manage a maximum of 64 axes¹⁰⁾ control.

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 area for a maximum of 99 point tables with a maximum of 1,024 point 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 223](#), or via the IndraWorks interface.

¹⁰⁾ depending on the type of



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

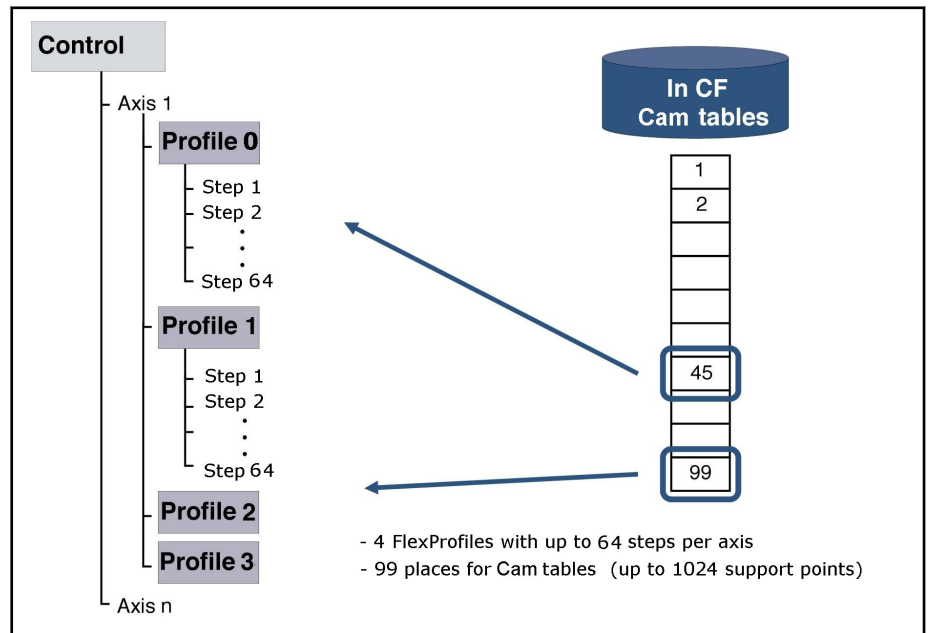


Fig. 6-63: Assigning the cam table/FlexProfile

For processing FlexProfiles and cam tables on the MLC, the following blocks are available:

ML_PLCopen.library

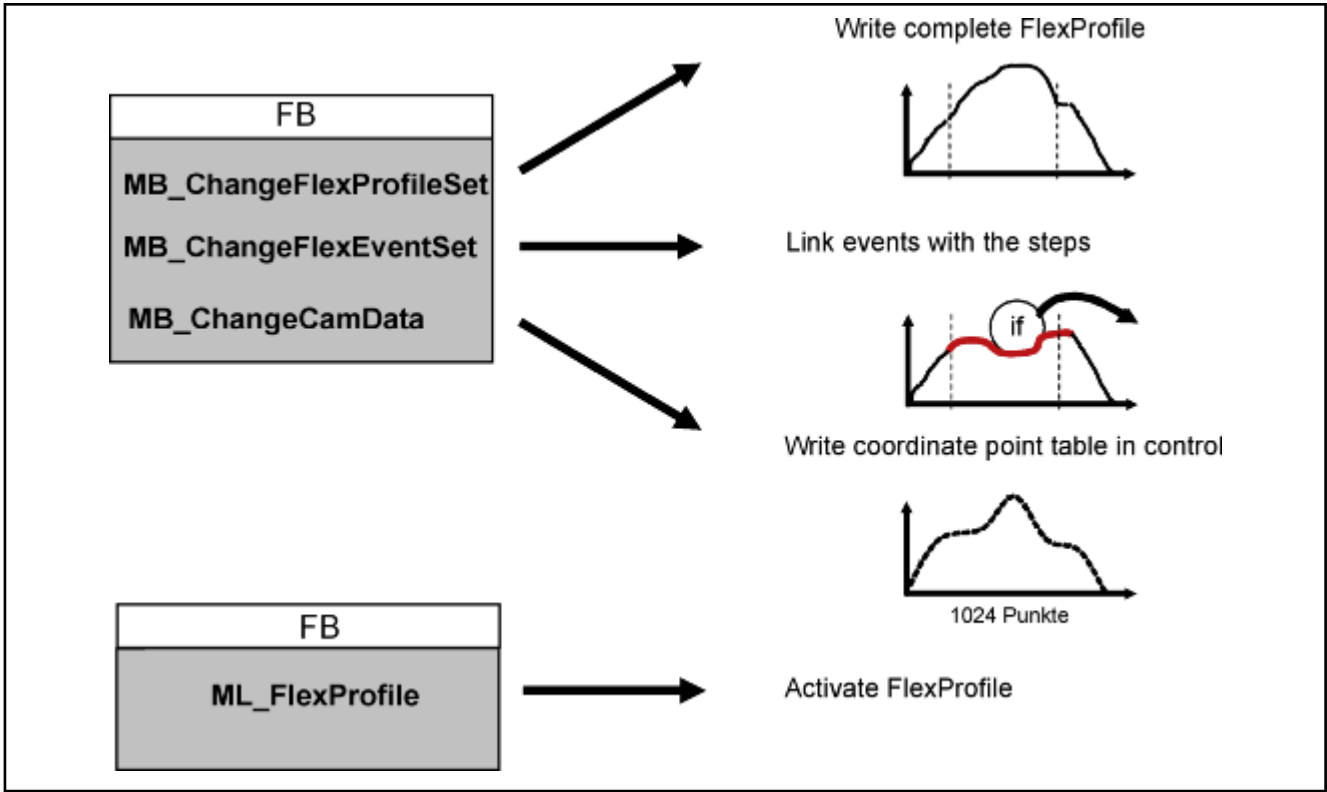


Fig. 6-64: Overview of the PLC blocks

- [MB_ChangeFlexProfileSet, page 236](#): Writing a complete FlexProfile, from the PLC to the Motion
- [MB_ChangeFlexEventSet, page 239](#): Configuration of events.
- [MB_ChangeCamData, page 223](#),: Transferring a cam table from the PLC memory to the Motion, C-0-2001...2099.

MB_ChangeFlexProfileSet

Brief Description The "MB_ChangeFlexProfileSet" function block allows a complete MotionProfile (FlexProfile) with a maximum of 64 steps to be described and/or an already existing one to be exchanged.

Library	Range
ML_PLCopen.library	Motion

Tab. 6-41: Library assignment

Interface description

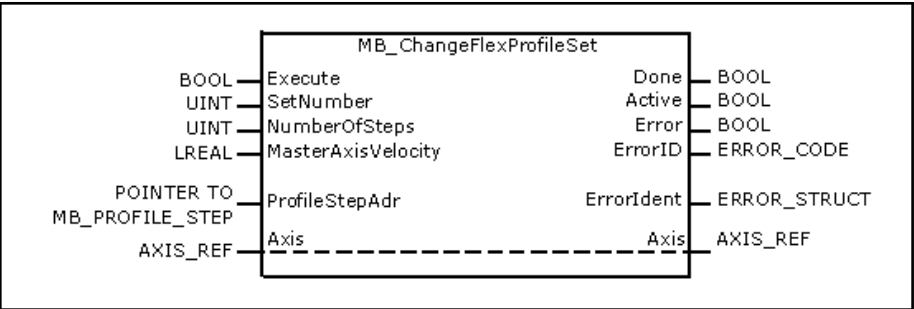


Fig. 6-65: MB_ChangeFlexProfileSet function block

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the axis for which the FlexProfile set is to be exchanged.
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..0.64)
	MasterAxisVelocity	LREAL	Standard master axis velocity to which the specified velocities relate
	ProfileStepAdr	POINTER TO MB_PROFILE_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	It indicates that an error has occurred in the FB instance.
	ErrorID	ERROR_CODE	Brief indication of the error cause.
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error.

Tab. 6-42: Interface of the MB_ChangeFlexProfileSet FB

Signal-time diagram

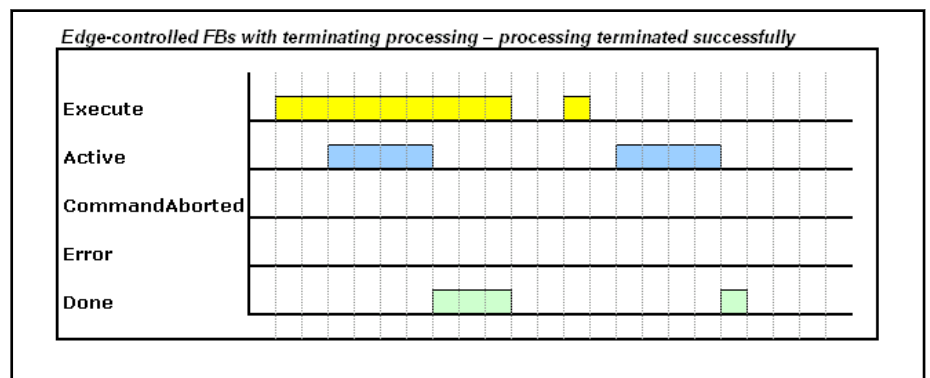


Fig. 6-66: Signal-time diagram: MB_ChangeFlexProfileSet

Functional Description

The function block allows a complete FlexProfile with 1..0.64 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 (of LREAL).

Error handling

The function block MB_ChangeProfileSet generates the following types of error messages:

- [MLC_TABLE](#), [ERROR_TABLE 16#0030](#), [page 343](#) - 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.

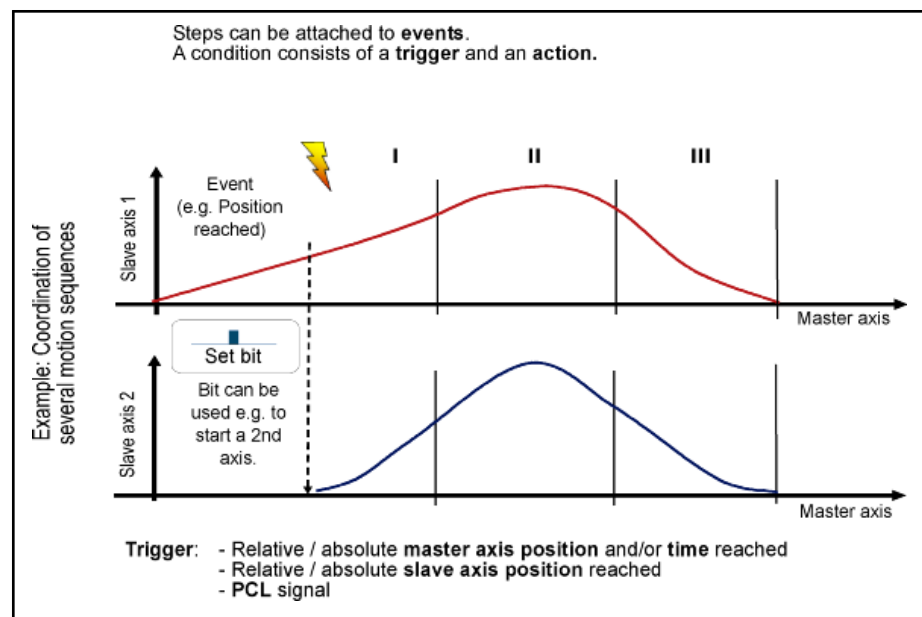


Fig. 6-67: Example of an event control

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.

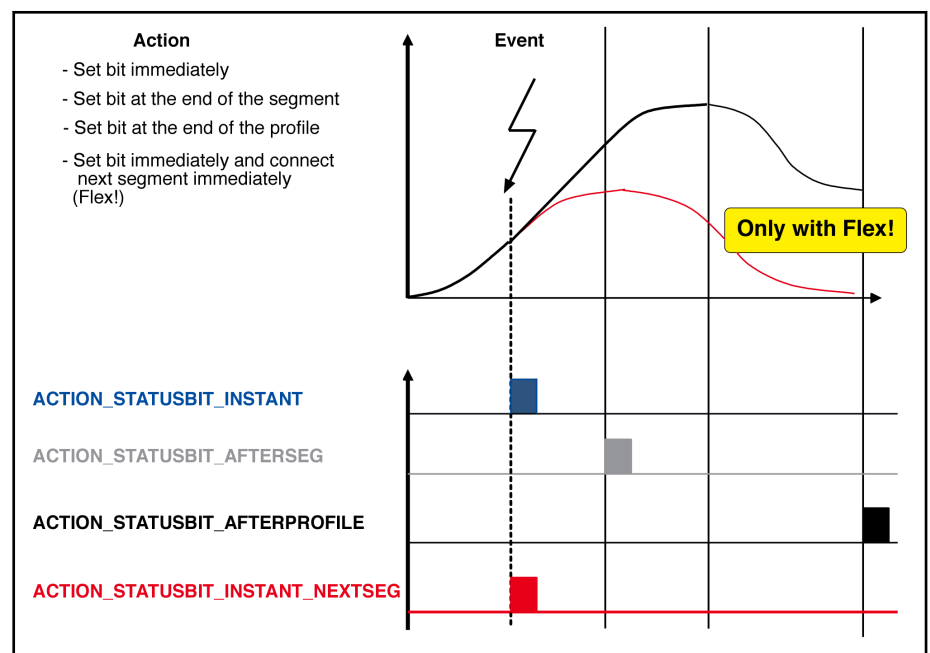


Fig. 6-68: Possible actions

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.

Library	Area
ML_PLCopen.library	Motion

Tab. 6-43: Library assignment

ML_PLCOpen.library

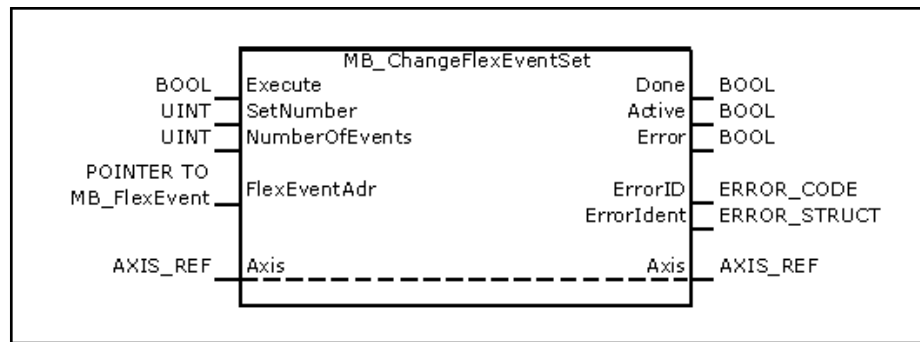


Fig. 6-69: Function block, MB_ChangeFlexEventSet

Interface description

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the axis for which the MotionProfile set is to be exchanged.
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 were accepted
	Active	BOOL	Processing of data runs after preprocessing is completed.
	Error	BOOL	It indicates that an error has occurred in the function blocks instance.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 6-44: 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.

Error handling The function block MB_ChangeFlexEventSet produces the following error messages:

- [MLC_TABLE page 343](#) - error when accessing the axis/control.

6.6.15 MC_GearIn

Brief description The MC_GearIn function block 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 drive and in the control
- SercosDrive
- Virtual axis
- Controller axis

Library	Range
ML_PLCopen.library	Motion

Tab. 6-45: Library assignment

Interface description

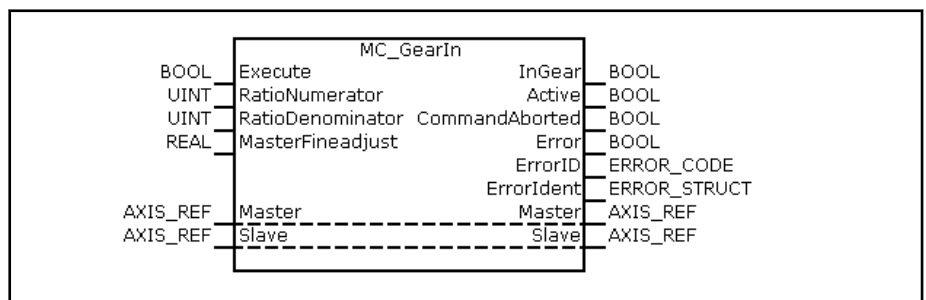


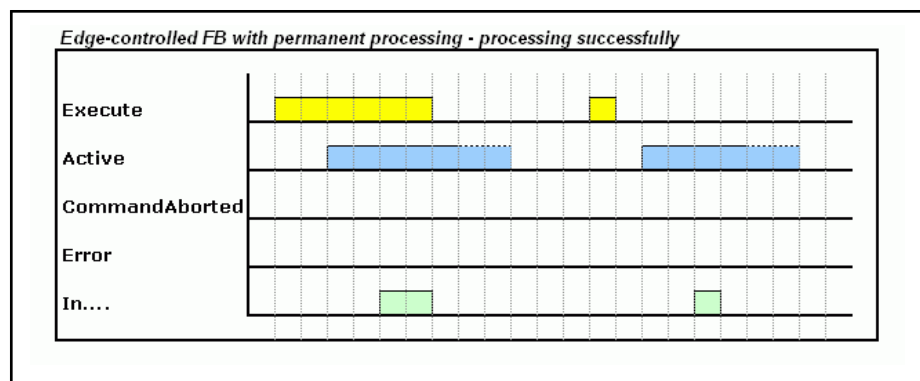
Fig. 6-70: FB MC_GearIn

	Name	Type	Comment
VAR_IN_OUT	Master	AXIS_REF	Contains information on the master axis
	Slave	AXIS_REF	It contains information on the slave axis which follows the master axis
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
VAR_OUTPUT	InGear	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

Tab. 6-46: Interface of the MC_GearIn function block

ML_PLCOpen.library

Signal-time diagram



In... InGear

Fig. 6-71: Signal-time diagram

6.6.16 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 drive and in the control
- SercosDrive
- Virtual axis
- Controller axis

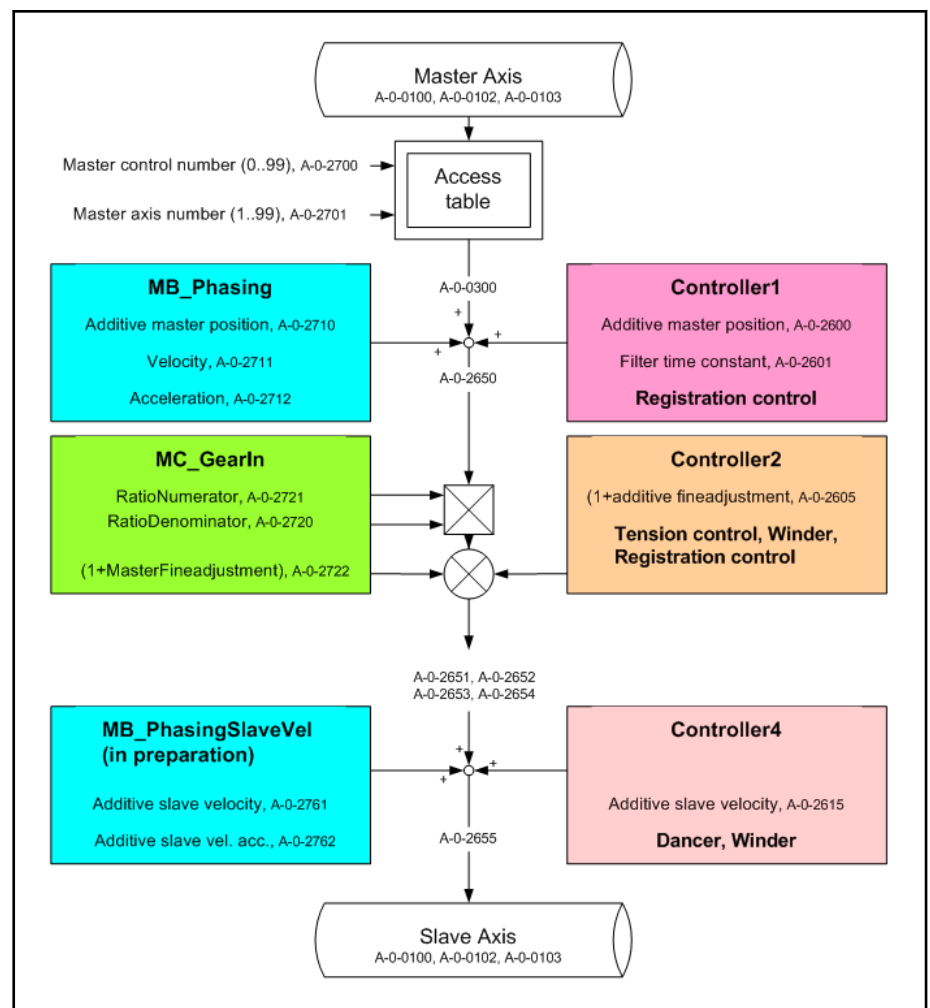


Fig. 6-72: MotionControl functional chain MC_GearIn

The function block contains two 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 adjustment. The fine adjustment value is specified in percent and calculated as $(1 + \text{fine adjustment})$.

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.

Troubleshooting

The function block MC_GearIn generates the following error messages:

- [MLC_TABLE](#), page 343 - error when accessing the axis/control.

6.6.17 MC_GearOut

Brief description

The function block MC_GearOut stops the electronic gear function with fine adjustment activated by function block MC_GearIn or MB_GearInPos or it

ML_PLCOpen.library

stops the profiles activated by function block MB_MotionProfile or ML_Flex-Profile.

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.

Library	Range
ML_PLCOpen.library	Motion

Tab. 6-47: Library assignment

Interface description

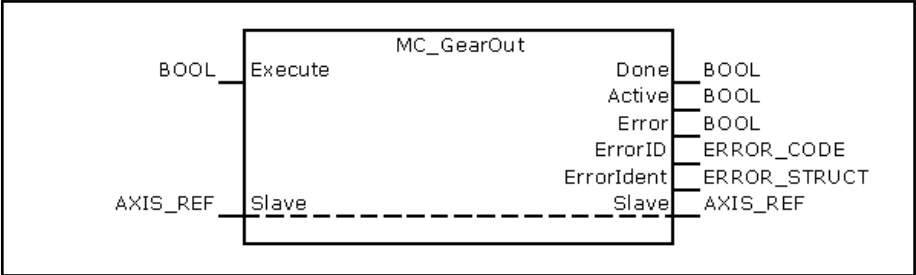


Fig. 6-73: FB MC_GearOut

	Name	Type	Comment
VAR_IN_OUT	Slave	AXIS_REF	Contains information regarding the slave axis which follows the master axis
VAR_INPUT	Execute	BOOL	Starts the action if there is a rising edge
VAR_OUTPUT	Done	BOOL	Decoupling completed successfully
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred in the FB instance
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error

Tab. 6-48: Interface of FB MC_GearOut

Signal-time diagram

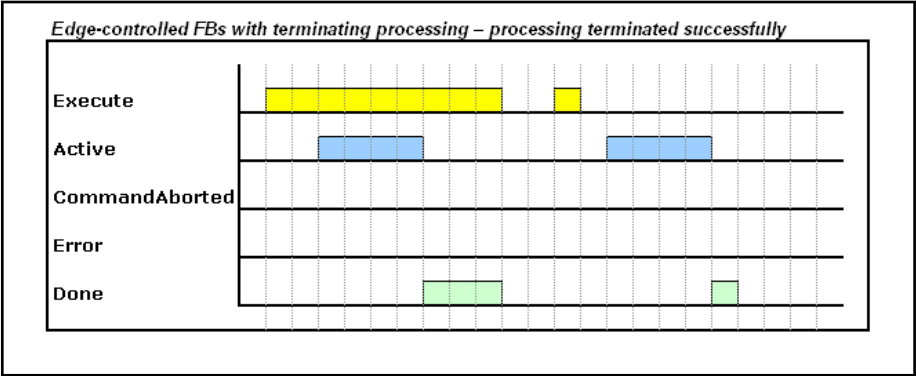


Fig. 6-74: Signal-time diagram

6.6.18 MC_GearOut, Special Features of SERCOS-based Axes

- Error handling
- The function block MC_GearOut generates the following error messages:
- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

6.6.19 MB_GearInPos

Brief description The MC_GearInPos function block 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 drive and in the control
- SercosDrive
- Virtual axis
- Controller axis

Library	Range
ML_PLCopen.library	Motion

Tab. 6-49: Library assignment

Interface description

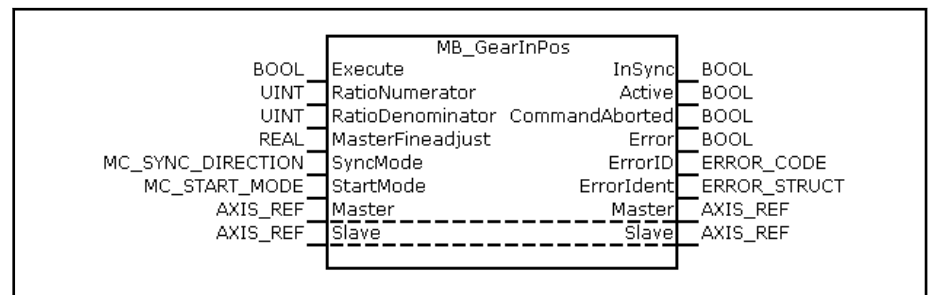


Fig. 6-75: FB MB_GearInPos

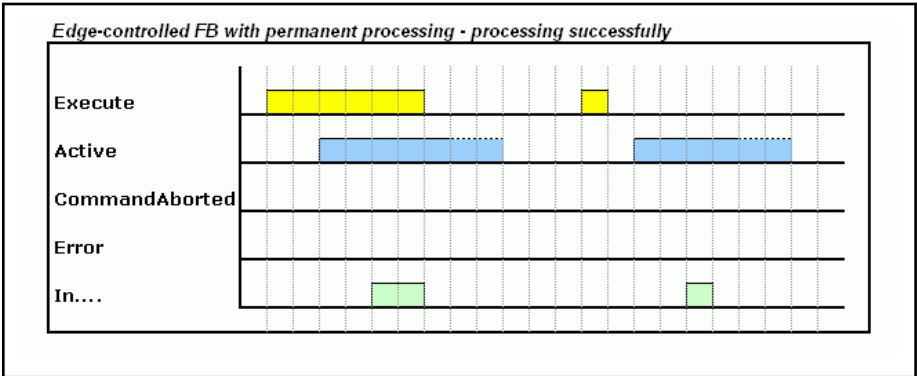
	Name	Type	Comment
VAR_IN_OUT	Master	AXIS_REF	Contains information on the master axis
	Slave	AXIS_REF	It contains information on the slave axis which follows the master axis
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
	RatioNumerator	UINT	Gear ratio - numerator master axis driven output revolutions Min: 1; max: 65535; default: 1
	RatioDenominator	UINT	Gear ratio - denominator "master axis driven input revolutions" Min: 1; max: 65535; default: 1
	MasterFineadjust	REAL	"Fine adjustment gear ratio" in %; default: 0.0, min.: -799.9999%, max.: 799.9999
	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

ML_PLCopen.library

	Name	Type	Comment
VAR_OUTPUT	InSync	BOOL	Synchronous running 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. 6-50: Interface of the MC_GearInPos function block

Signal-time diagram



In... InSync
Fig. 6-76: Signal-time diagram

6.6.20 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 drive and in the control
 - SercosDrive
 - Virtual axis
 - Controller axis

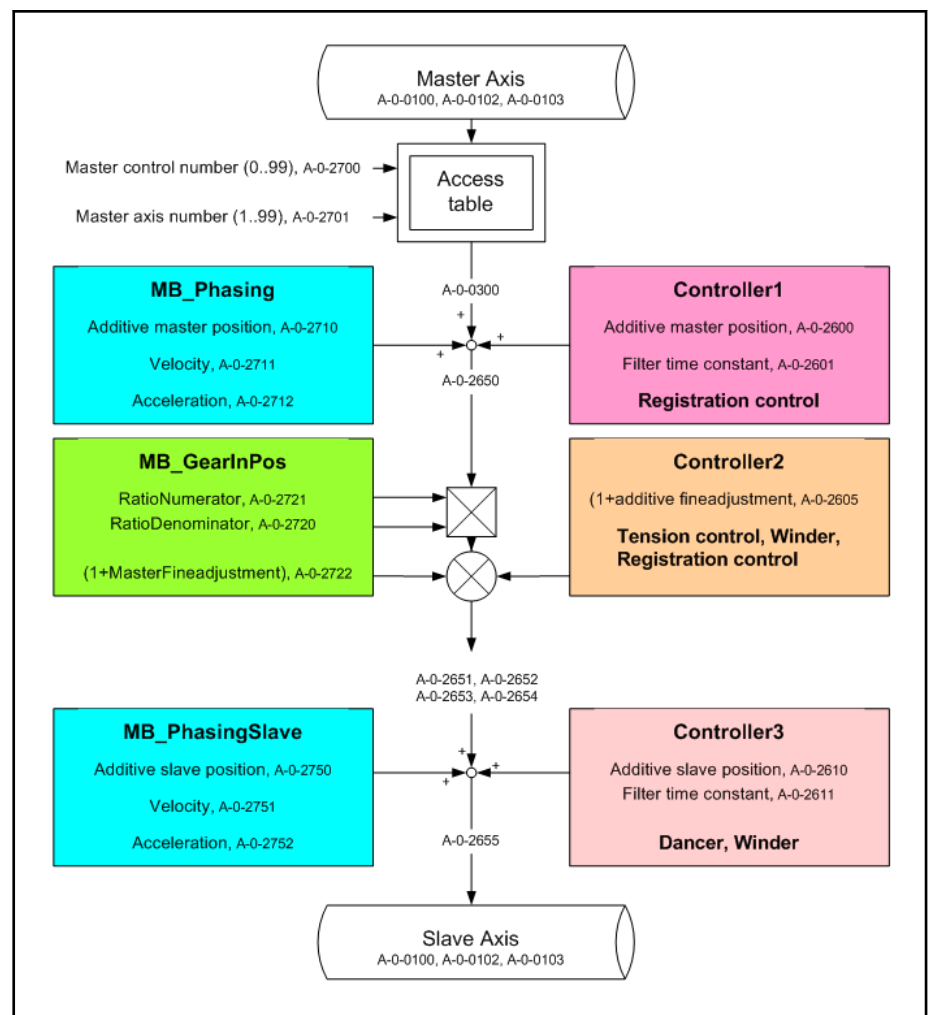


Fig. 6-77: MotionControl functional chain MC_GearInPos

The function block contains two 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 $(1 + \text{fine adjustment})$.

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.

Troubleshooting

The function block MB_GearInPos generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

ML_PLCopen.library

6.7 Function Blocks 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 to ML_PLCopen, 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.

Library	Range
ML_PLCopen	Motion

Tab. 6-51: Library assignment

Interface description

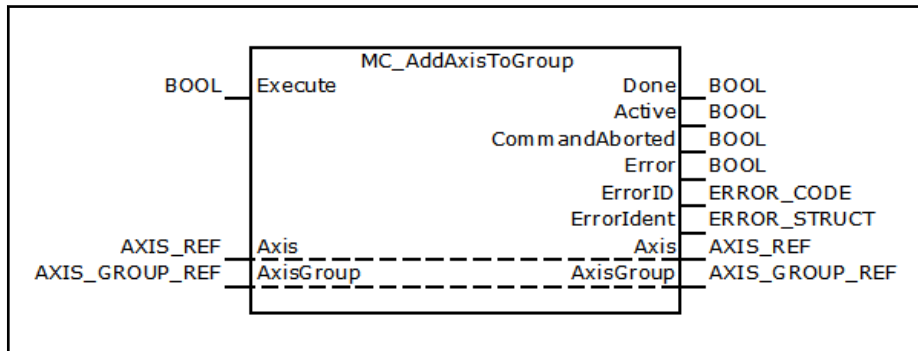


Fig. 6-78: MC_AddAxisToGroup

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Control number and object number of the axis group
	Axis	AXIS_REF	Control number and object number of the axis
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.
	Active	BOOL	Data processing runs when preprocessing is complete
	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

Tab. 6-52: Interface of the MC_AddAxisToGroup FB

Signal-time diagram

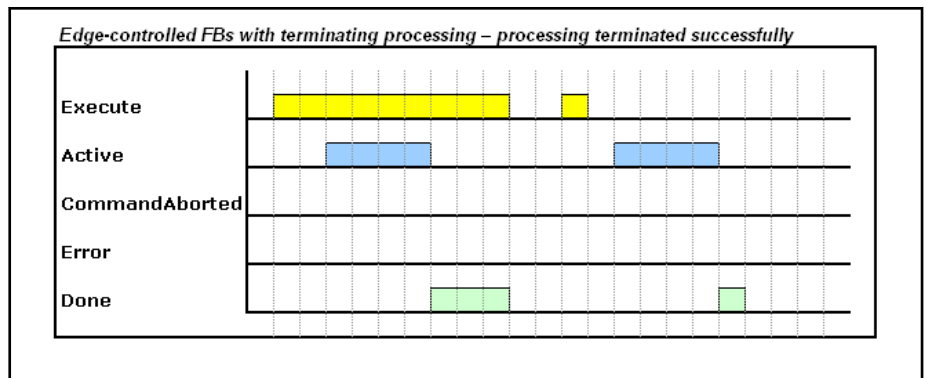


Fig. 6-79: Signal-time diagram: MC_AddAxisToGroup

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 to ML_PLCopen, 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.

Library	Range
ML_PLCopen	Motion

Tab. 6-53: Library assignment

Interface description

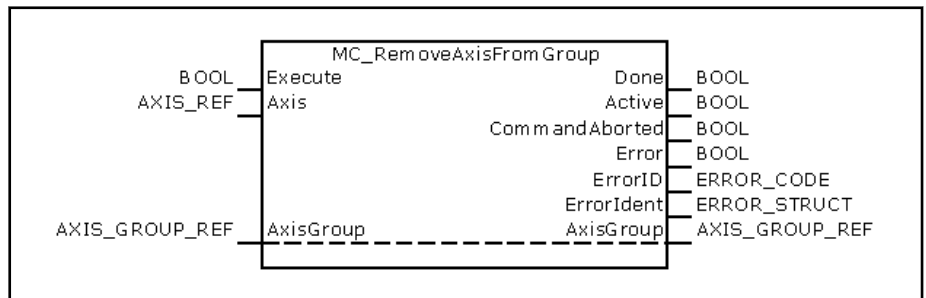


Fig. 6-80: MC_RemoveAxisFromGroup

	Name	Type	Comment
VAR_IN_OUT	AxisGroup	AXIS_GROUP_REF	Control number and object number of the axis group
VAR_INPUT	Execute	BOOL	Starts the action at a rising edge
	Axis	AXIS_REF	Control number and object number of the axis
VAR_OUTPUT	Done	BOOL	The axis has been successfully removed from the axis group.
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by a subsequent axis command
	Error	BOOL	Indicates that an error occurred in the function block instance

ML_PLCopen.library

	Name	Type	Comment
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 6-54: Interface of the MC_RemoveAxisFromGroup function block

Signal-time diagram

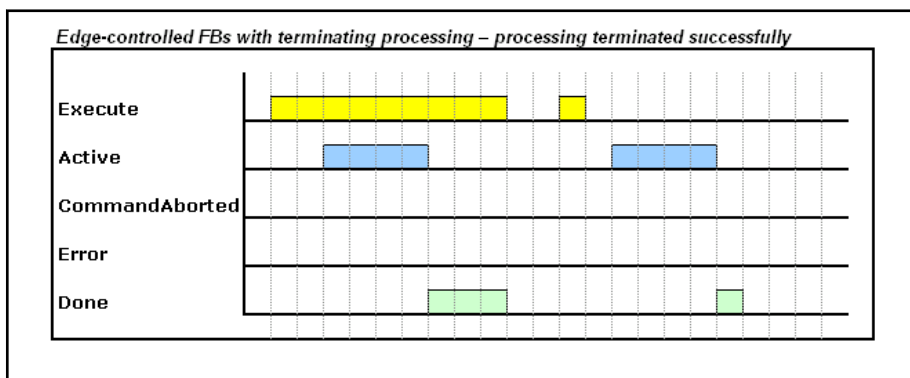


Fig. 6-81: Signal-time diagram: MC_RemoveAxisFromGroup

6.8 Function Blocks for the Use of 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 following blocks are available:

Analog command value channel, operating mode "Cyclic Analog Command"

- [ML_OpenCyclicAnalogChannel](#), page 252, with
- [ML_WriteCyclicAnalog](#), page 253,

Position command value channel, operation mode "Cyclic Position Command"

- [ML_OpenCyclicPositionChannel](#), page 254, with
- [ML_WriteCyclicPosition](#), page 255,

Power command value channel, operation mode "Cyclic Torque Command"

- [ML_OpenCyclicTorqueChannel](#), page 258, with
- [ML_WriteCyclicTorque](#), page 259,

Velocity command value channel, operation mode "Cyclic Velocity Command"

- [ML_OpenCyclicVelocityChannel](#), page 254, with
- [ML_WriteCyclicVelocity](#), page 261,

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:

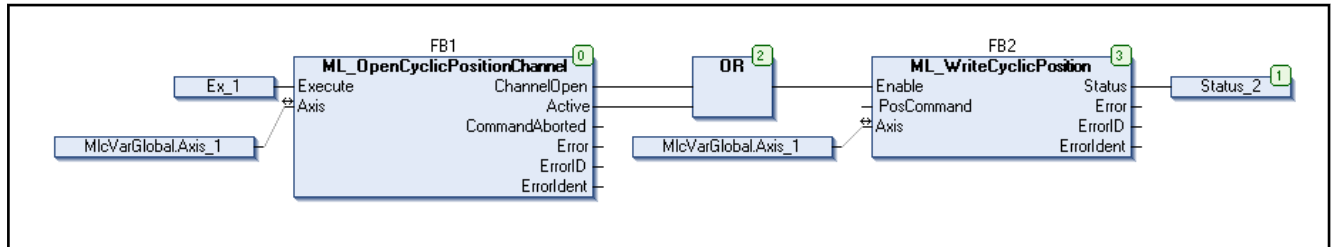


Fig. 6-82: Interconnecting the function blocks ML_OpenCyclicPositionChannel and ML_WriteCyclicPosition

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). As soon as the output CommandAborted is TRUE at the ML_OpenCyclicPositionChannel function block, no cyclic position command value can 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.

ML_PLCOpen.library

Output	Description/action
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. 6-55: 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 analog command 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 with the servos meter). Otherwise, jerky movements of the axis are likely.

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 the real axis of the type HNC100-3X

Library	Range
ML_PLCOpen.library	Motion

Tab. 6-56: Library assignment

Interface description

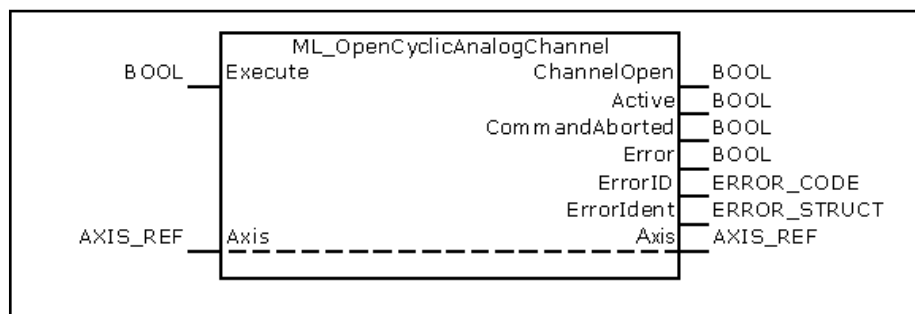


Fig. 6-83: ML_OpenCyclicAnalogChannel function block

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge.

	Name	Type	Comment
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

Tab. 6-57: Interface of the ML_OpenCyclicAnalogChannel function block

Error handling The function block ML_OpenCyclicAnalogChannel generates the following error messages:

- [MLC_TABLE, page 343](#) - 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. Only if the cyclic analog command value channel is open will the analog value become effective in the next cycle on the axis.



In order to open the cyclic analog command value channel, the function block [ML_OpenCyclicAnalogChannel, page 254](#), has to be used.

The function block has to be used in a PLC task which is synchronous with the sercos meter. Otherwise, jerky movements of the axis are likely.

Library	Range
ML_PLCopen.library	Motion

Tab. 6-58: Library assignment

Interface description

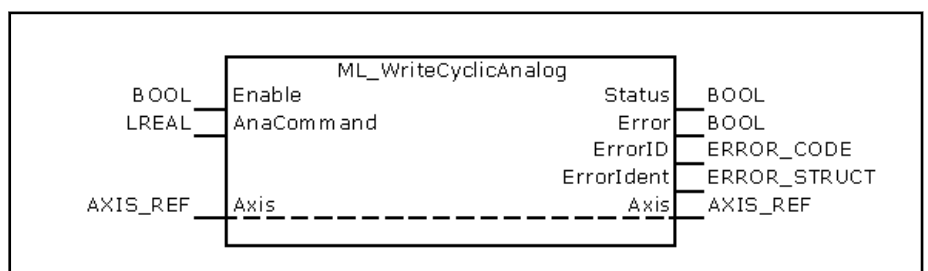


Fig. 6-84: ML_WriteCyclicAnalog

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
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.

ML_PLCOpen.library

	Name	Type	Comment
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

Tab. 6-59: 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 has the scaling of the assigned axis.

Error handling

The function block ML_WriteCyclicAnalog generates the following error messages:

- [MLC_TABLE, page 343](#) - 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 with the sercos meter). Otherwise, jerky movements of the axis are likely.

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).

Library	Range
ML_PLCOpen.library	Motion

Tab. 6-60: Library assignment

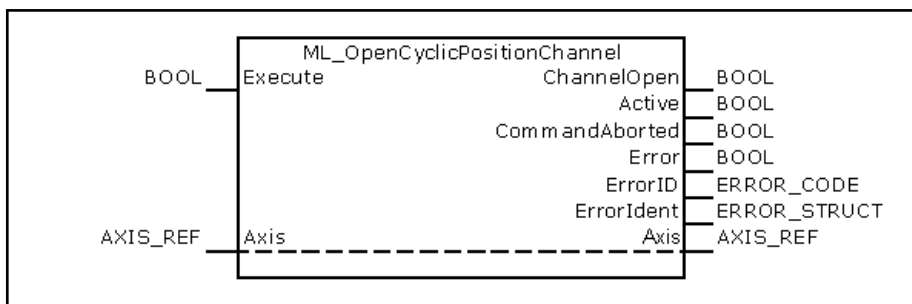
Interface description

Fig. 6-85: ML_OpenCyclicPositionChannel function block

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
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

Tab. 6-61: 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 180.

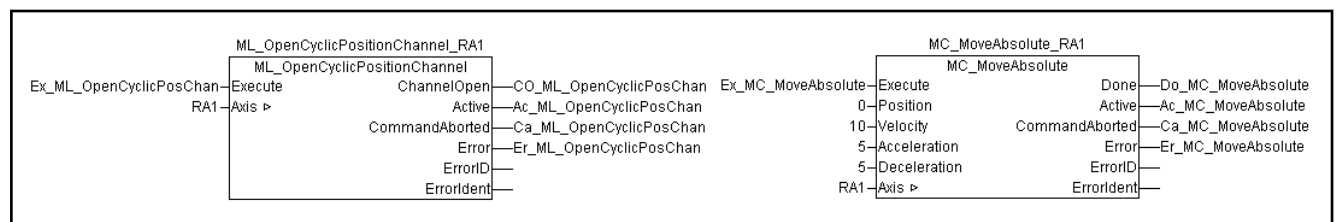


Fig. 6-86: Time behavior ML_OpenCyclicPositionChannel

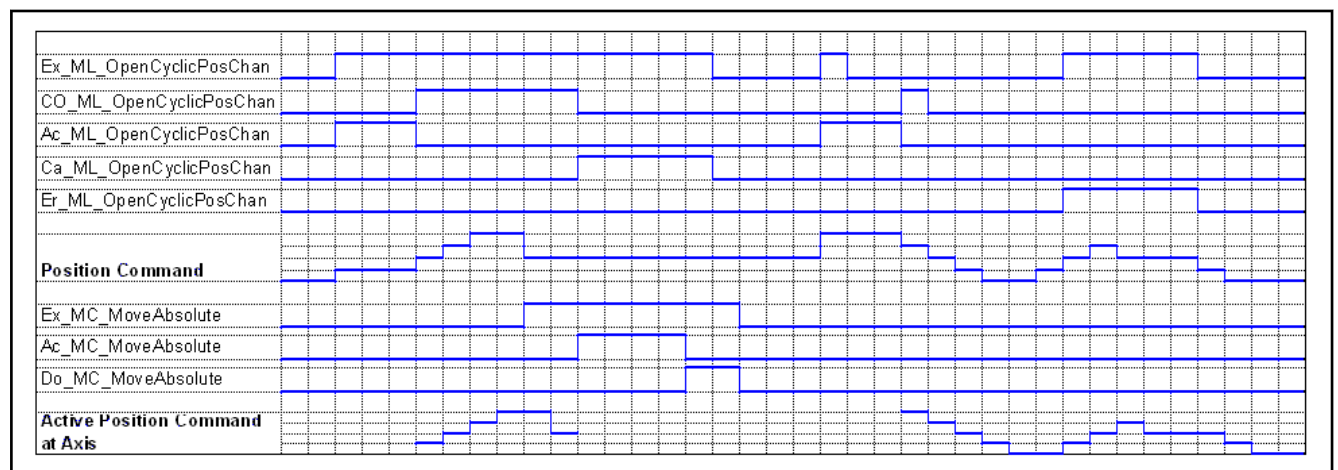


Fig. 6-87: Time behavior ML_OpenCyclicPositionChannel

Error handling

The function block ML_OpenCyclicPositionChannel generates the following error messages:

- [MLC_TABLE](#), page 343 - 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. Only if the cyclic position command value channel is open will the position command become effective in the next cycle on the axis.

ML_PLCOpen.library



In order to open the cyclic position command value channel, the function block **ML_OpenCyclicPositionChannel**, [page 254](#), has to be used.

The function block has to be used in a PLC task which is synchronous with the sercos meter. Otherwise, jerky movements of the axis are likely.

Library	Range
ML_PLCOpen.library	Motion

Tab. 6-62: Library assignment

Interface description

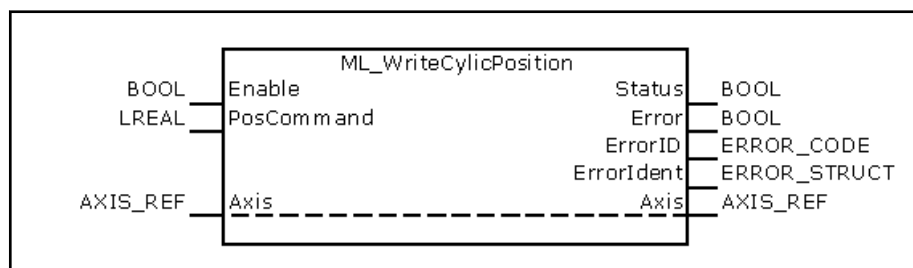


Fig. 6-88: ML_WriteCyclicPosition

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
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

Tab. 6-63: 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).

Error handling

The function block ML_WriteCyclicPosition generates the following error messages:

- [MLC_TABLE](#), [page 343](#) - 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.

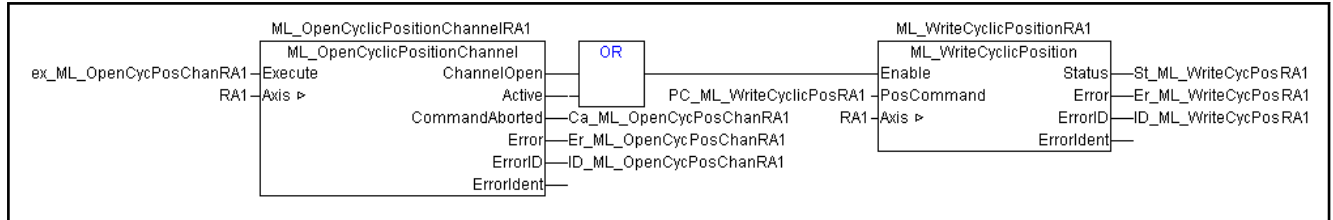


Fig. 6-89: Example: *ML_OpenCyclicPositionChannel* and *ML_WriteCyclicPosition*

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_OpenCyclicPositionCommand* 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.

ML_PLCOpen.library

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. 6-64: 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 with the servos meter). Otherwise, jerky movements of the axis are likely.

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.

Library	Range
ML_PLCOpen.library	Motion

Tab. 6-65: Library assignment

Interface description

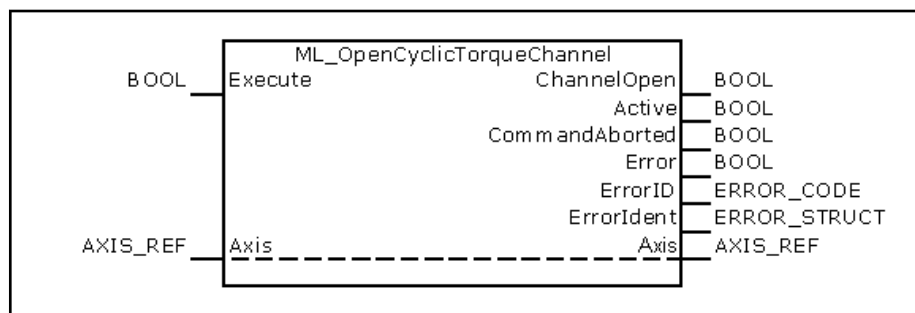


Fig. 6-90: ML_OpenCyclicTorqueChannel function block

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge.

	Name	Type	Comment
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

Tab. 6-66: Interface of the ML_OpenCyclicPoswerChannel function block

Error handling

The function block ML_OpenCyclicTorqueChannel generates the following error messages:

- [MLC_TABLE, page 343](#) - 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. Only if the cyclic force command value channel is open will the force value become effective in the next cycle on the axis.



In order to open the cyclic power command value channel, the function block [ML_OpenCyclicTorqueChannel, page 254](#), has to be used.

The function block has to be used in a PLC task which is synchronous with the sercos meter. Otherwise, jerky movements of the axis are likely.

Library	Range
ML_PLCopen.library	Motion

Tab. 6-67: Library assignment

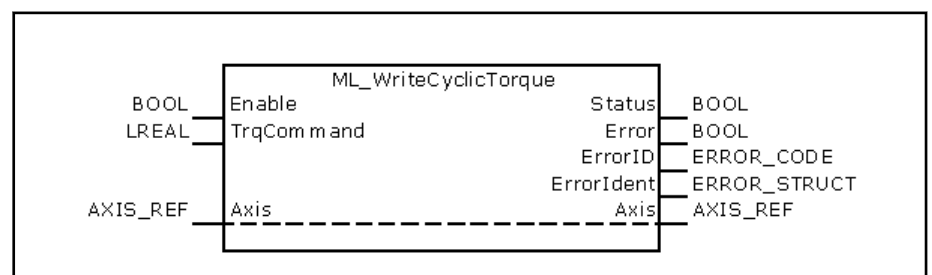
Interface description

Fig. 6-91: ML_WriteCyclicTorque

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
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.

ML_PLCOpen.library

	Name	Type	Comment
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

Tab. 6-68: 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.

Error handling

The function block ML_WriteCyclicTorque generates the following error messages:

- [MLC_TABLE, page 343](#) - 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 with the servos meter). Otherwise, jerky movements of the axis are likely.

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.

Library	Range
ML_PLCOpen.library	Motion

Tab. 6-69: Library assignment

Interface description

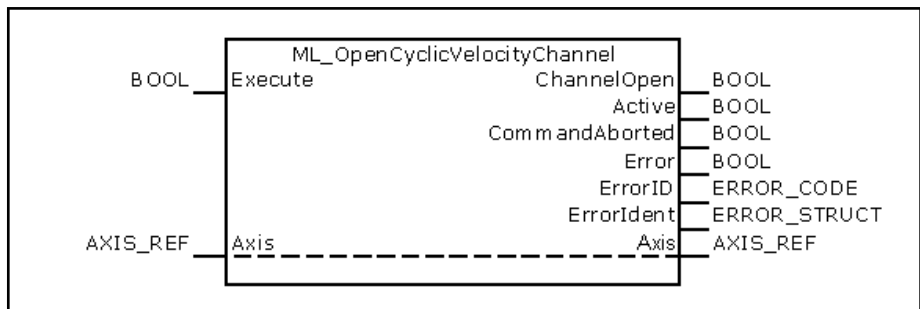


Fig. 6-92: ML_OpenCyclicVelocityChannel function block

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
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

Tab. 6-70: Interface of the ML_OpenCyclicVelocityChannel function block

Error handling The function block ML_OpenCyclicVelocityChannel generates the following error messages:

- [MLC_TABLE, page 343](#) - 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. Only if the cyclic velocity command value channel is open, the velocity value become active in the next cycle on the axis.



In order to open the cyclic velocity command value channel, the function block [ML_OpenCyclicVelocityChannel, page 254](#), has to be used.

The function block has to be used in a PLC task which is synchronous with the sercos meter. Otherwise, jerky movements of the axis are likely.

Library	Range
ML_PLCopen.library	Motion

Tab. 6-71: Library assignment

Interface description

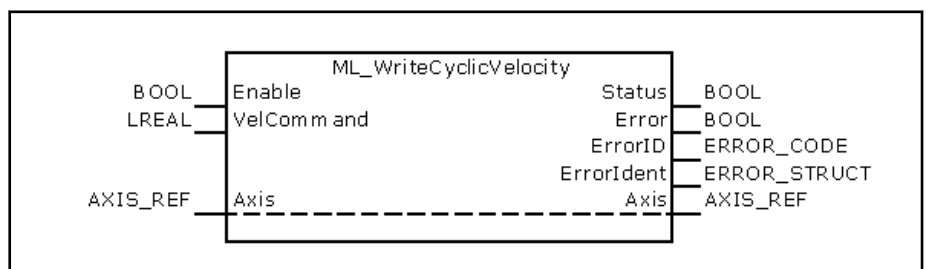


Fig. 6-93: ML_WriteCyclicVelocity

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the concrete axis
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.

ML_PLCOpen.library

	Name	Type	Comment
	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

Tab. 6-72: 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.

Error handling

The function block ML_WriteCyclicVelocity generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

6.9 Function blocks for controlled motion

6.9.1 General Information on Controlled Motion

This section describes the possibility for the PLC to have complete control over the cyclic command value specification of an axis. 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 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.

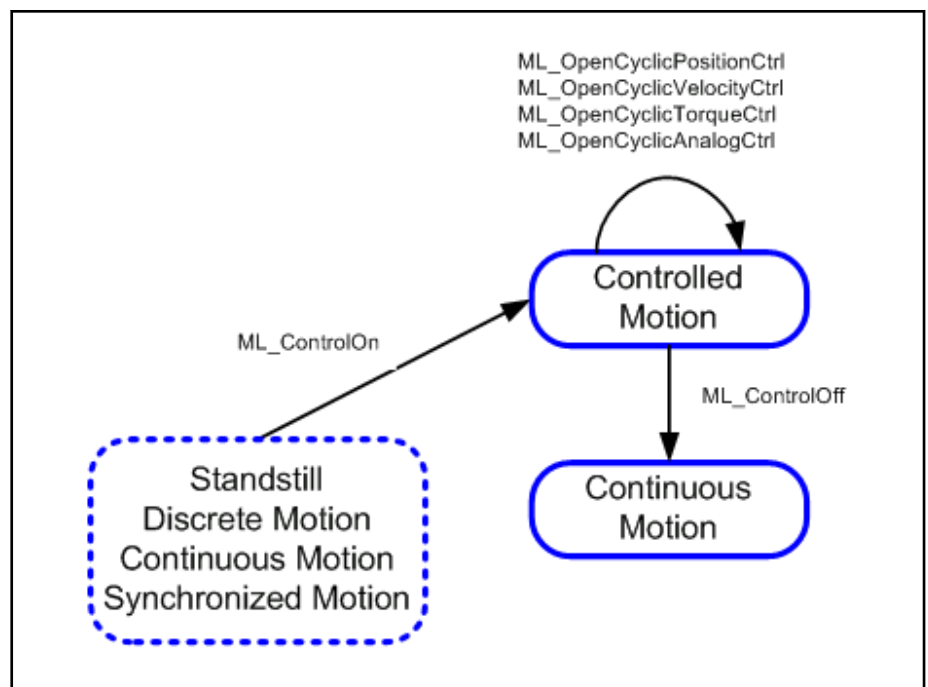


Fig. 6-94: Integrating the "Controlled Motion" state into the PLCopen state machine

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 264
- [ML_ControlOff](#), page 264

ML_ControlWriteActualData function block to write the current values of a PLC controller to the controller axis of the MLC

- [ML_ControlWriteActualData](#), page 265

Analog command value channel, operating mode "Cyclic Analog Command"

- [ML_OpenCyclicAnalogChannelCtrl](#), page 266
- [ML_WriteCyclicAnalogCtrl](#), page 267

Position command value channel, operation mode "Cyclic Position Command"

- [ML_OpenCyclicPositionChannelCtrl](#), page 268
- [ML_WriteCyclicPositionCtrl](#), page 269

Power command value channel, operation mode "Cyclic Torque Command"

- [ML_OpenCyclicTorqueChannelCtrl](#), page 270
- [ML_WriteCyclicTorqueCtrl](#), page 271

Velocity command value channel, operation mode "Cyclic Velocity Command"

- [ML_OpenCyclicVelocityChannelCtrl](#), page 272
- [ML_WriteCyclicVelocityCtrl](#), page 274

The supported function blocks can vary depending on axis type.

ML_PLCOpen.library

6.9.2 ML_ControlOn

Brief description The function block ML_ControlOn is used to give the PLC complete control over the cyclic command value specification of an axis. When this function block is activated, the axis state changes to "Controlled Motion". The "Controlled Motion" state be ended with the function block ML_ControlOff.



If ML_ControlOn is activated, the last motion command remains in effect until a function block of type ML_OpenCyclic*Ctrl is carried out.

Library	Range
ML_PLCOpen.library	Controlled Motion

Tab. 6-73: Library assignment

Interface description

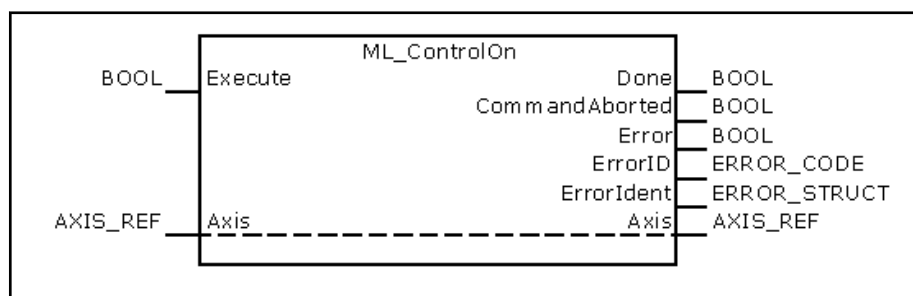


Fig. 6-95: FB ML_ControlOn

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the triggered axis
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 FB instance.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error

Tab. 6-74: Interface of FB MC_ControlOn

Troubleshooting The ML_ControlOn function block generates the following error messages:

- [MLC_TABLE page 343](#) - error when accessing the axis/control.

6.9.3 ML_ControlOff

Brief description The function block ML_ControlOff is used to end the PLC's cyclic control over an axis. 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.

Library	Range
ML_PLCOpen.library	Controlled Motion

Tab. 6-75: Library assignment

Interface description

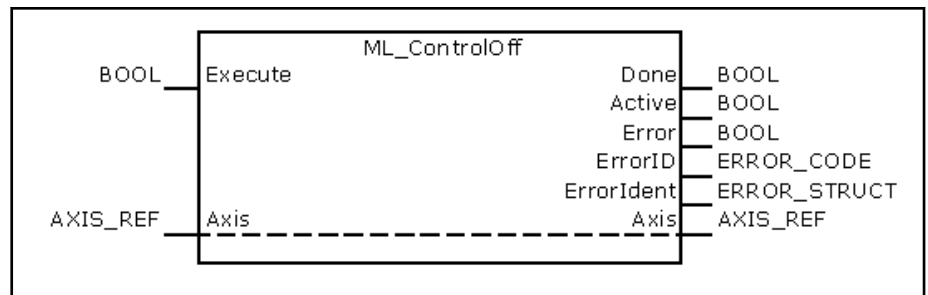


Fig. 6-96: FB ML_ControlOff

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the actual axis.
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	Indicates that the deactivation process is running.
	Error	BOOL	Indicates that an error has occurred in the FB instance.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error

Tab. 6-76: Interface of FB MC_ControlOff

Troubleshooting The ML_ControlOff function block generates the following error messages:

- [MLC_TABLE page 343](#) - 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.

Library	Range
ML_PLCopen.library	Controlled Motion

Tab. 6-77: Library assignment

Interface description

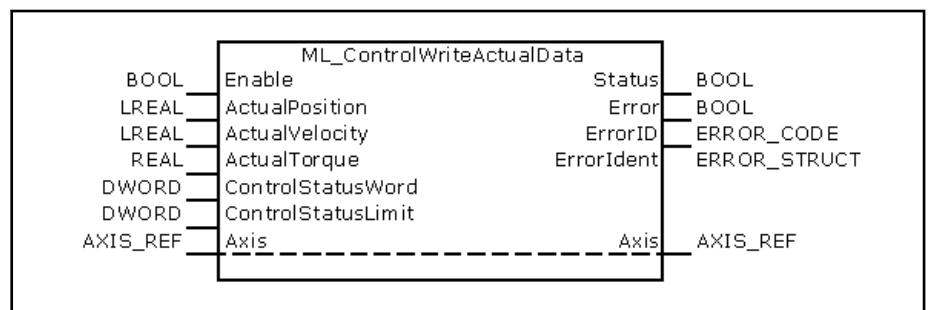


Fig. 6-97: ML_ControlWriteActualData

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the triggered axis.
VAR_INPUT	Enable	BOOL	As long as the "Enable" output is TRUE,

ML_PLCOpen.library

	Name	Type	Comment
	ActualPosition	LREAL	Actual position of the controller
	ActualVelocity	LREAL	Actual velocity of the controller
	ActualTorque	REAL	Actual value torque/force of controller
	ControlStatusWord	DWORD	Status word of controller (see A-0-0019)
	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
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 6-78: Interface of the FB ML_ControlWriteActualData

Troubleshooting

The function block ML_ControlWriteActualData generates the following error messages in:

- [MLC_TABLE, page 343](#) - 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 should be used in a PLC task which is synchronous with the sercos meter. Otherwise, jerky movements of the axis are likely.

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 HNC100-3x.

Library	Range
ML_PLCOpen.library	Controlled Motion

Tab. 6-79: Library assignment

Interface description

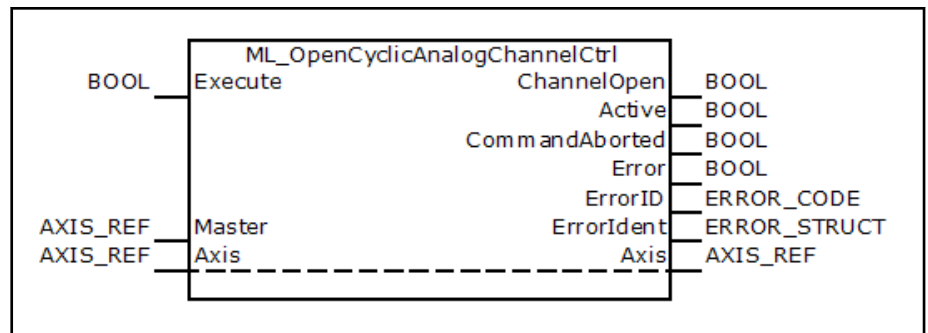


Fig. 6-98: FB ML_OpenCyclicAnalogChannelCtrl

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the triggered axis.
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge.
	Master	AXIS_REF	It contains information on the master axis.
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.
	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

Tab. 6-80: Interface of FB ML_OpenCyclicAnalogChannelCtrl

Error handling The function block ML_OpenCyclicAnalogChannelCtrl generates the following error messages:

- [MLC_TABLE, page 343](#) - 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. Only if the cyclic analog command value channel is open will the analog value become effective in the next cycle at the axis.



In order to open the cyclic analog command value channel, the function block [ML_OpenCyclicAnalogChannelCtrl, page 266](#), has to be used.

The function block should be used in a PLC task which is synchronous with the sercos meter. Otherwise, jerky movements of the axis are likely.

Library	Range
ML_PLCopen.library	Controlled Motion

Tab. 6-81: Library assignment

ML_PLCOpen.library

Interface description

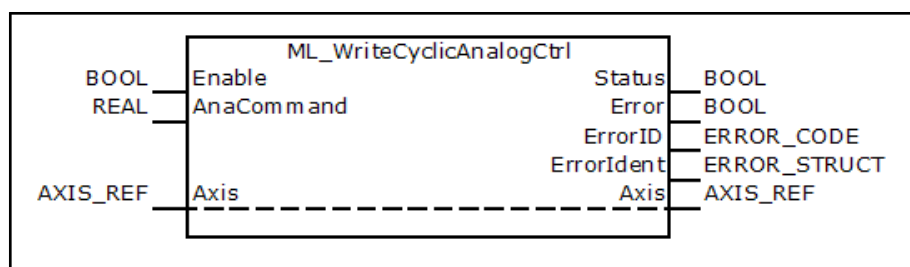


Fig. 6-99: ML_WriteCyclicAnalogCtrl

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the triggered axis.
VAR_INPUT	Enable	BOOL	As long as the "Enable" output is TRUE, the analog value "AnaCommand" is written on the axis. The analog value only becomes effective when the cyclic analog command value channel is open.
	AnaCommand	REAL	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.
	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. 6-82: 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 has the scaling of the assigned axis.

Error handling

The function block ML_WriteCyclicAnalogCtrl generates the following error messages:

- [MLC_TABLE, page 343](#) - 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 should be used in a PLC task which is synchronous with the sercos meter. Otherwise, jerky movements of the axis are likely.

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.

Library	Range
ML_PLCopen.library	Controlled Motion

Tab. 6-83: Library assignment

Interface description

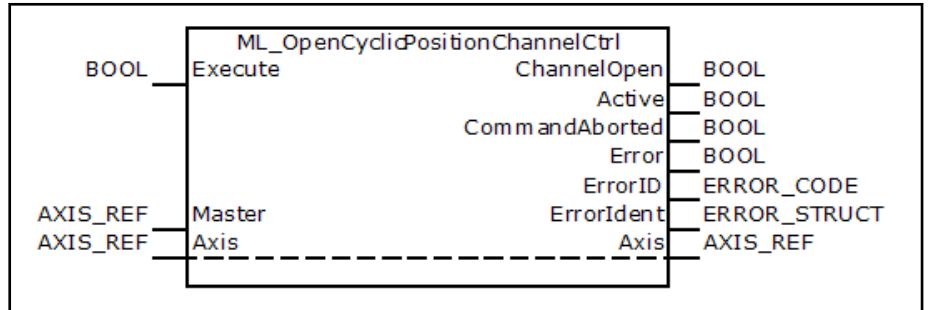


Fig. 6-100: FB ML_OpenCyclicPositionChannelCtrl

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the triggered axis.
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge.
	Master	AXIS_REF	It contains information on the master axis.
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

Tab. 6-84: Interface of FB ML_OpenCyclicPositionChannelCtrl

Error handling

The function block ML_OpenCyclicPositionChannelCtrl generates the following error messages:

- [MLC_TABLE, page 343](#) - 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. Only if the cyclic position command value channel is open will the analog value become effective in the next cycle at the axis.



In order to open the cyclic position value value channel, the function block [ML_OpenCyclicPositionChannelCtrl, page 268](#), has to be used.

The function block has to be used in a PLC task which is synchronous with the sercos meter. Otherwise, jerky movements of the axis are likely.

ML_PLCopen.library

Library	Range
ML_PLCopen.library	Controlled Motion

Tab. 6-85: Library assignment

Interface description

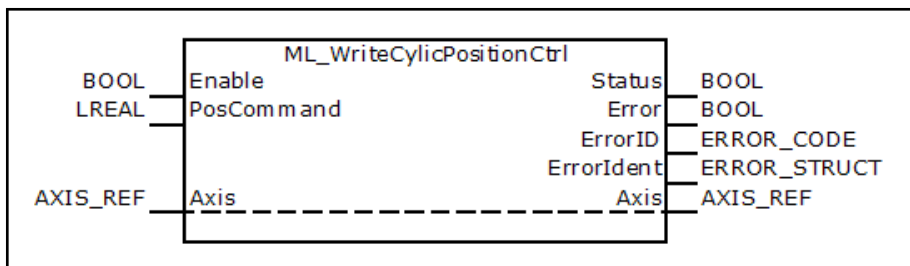


Fig. 6-101: ML_WriteCyclicPositionCtrl

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the triggered axis.
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

Tab. 6-86: 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.

Error handling

The function block ML_WriteCyclicPositionCtrl generates the following error messages:

- [MLC_TABLE, page 343](#) - 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 should be used in a PLC task which is synchronous with the sercos meter. Otherwise, jerky movements of the axis are likely.

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.

Library	Range
ML_PLCOpen.library	Controlled Motion

Tab. 6-87: Library assignment

Interface description

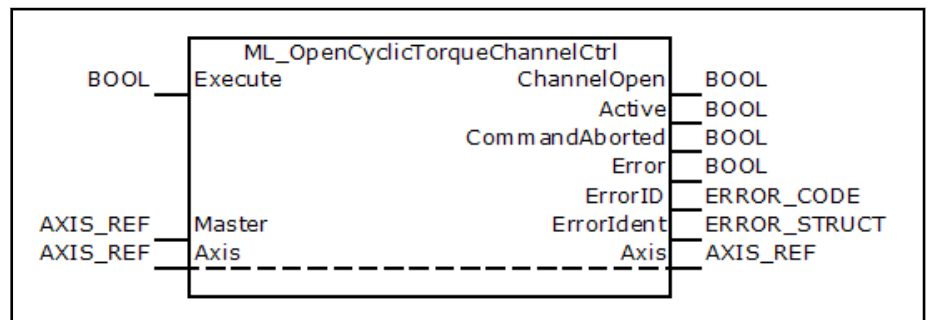


Fig. 6-102: FB ML_OpenCyclicTorqueChannelCtrl

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the triggered axis.
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge.
	Master	AXIS_REF	It contains information on the master axis.
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

Tab. 6-88: Interface of ML_OpenCyclicTorqueChannelCtrl FB

Error handling

The function block ML_OpenCyclicTorqueChannelCtrl generates the following error messages:

- [MLC_TABLE, page 343](#) - 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

ML_PLCOpen.library

force/torque command value channel is open, the force value become effective in the next cycle on the axis.



In order to open the cyclic force/torque command value channel, the function block [ML_OpenCyclicTorqueChannelCtrl](#), page 270, has to be used.

The function block has to be used in a PLC task which is synchronous with the sercos meter. Otherwise, jerky movements of the axis are likely.

Library	Range
ML_PLCOpen.library	Controlled Motion

Tab. 6-89: Library assignment

Interface description

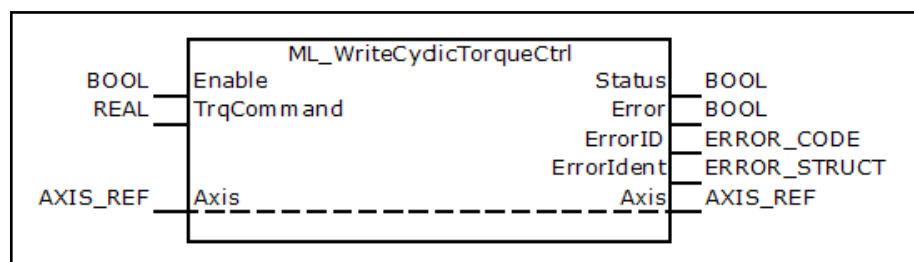


Fig. 6-103: ML_WriteCyclicTorqueCtrl

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the triggered axis.
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	REAL	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

Tab. 6-90: 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.

Error handling

The function block ML_WriteCyclicTorqueCtrl generates the following error messages:

- [MLC_TABLE](#), page 343 - 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 should be used in a PLC task which is synchronous with the sercos meter. Otherwise, jerky movements of the axis are likely.

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.

Library	Range
ML_PLCopen.library	Controlled Motion

Tab. 6-91: Library assignment

Interface description

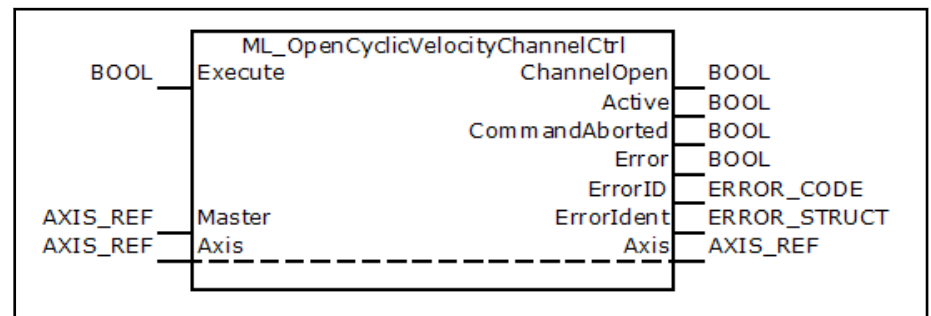


Fig. 6-104: FB ML_OpenCyclicVelocityChannelCtrl

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the triggered axis.
VAR_INPUT	Execute	BOOL	Starts the function block at a rising edge.
	Master	AXIS_REF	It contains information on the master axis.
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

Tab. 6-92: Interface of the ML_OpenCyclicVelocityChannelCtrl FB

Error handling

The function block ML_OpenCyclicVelocityChannelCtrl generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

ML_PLCOpen.library

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. Only if the cyclic velocity command value channel is open will the velocity value become effective in the next cycle on the axis.



In order to open the cyclic velocity command value channel, the function block [ML_OpenCyclicVelocityChannelCtrl](#), page 272, has to be used.

The function block has to be used in a PLC task which is synchronous with the sercos meter. Otherwise, jerky movements of the axis are likely.

Library	Range
ML_PLCOpen.library	Controlled Motion

Tab. 6-93: Library assignment

Interface description

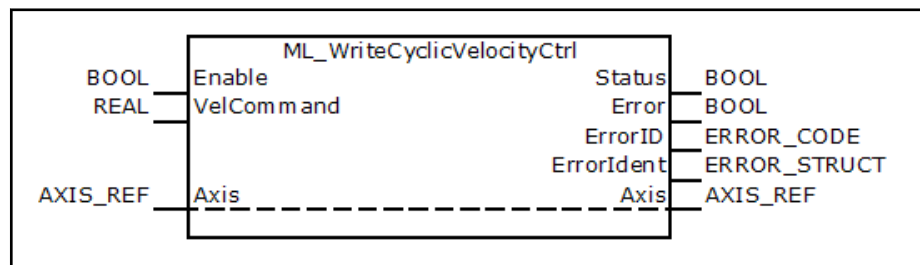


Fig. 6-105: ML_WriteCyclicVelocityCtrl

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the triggered axis.
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	REAL	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

Tab. 6-94: 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.

Error handling

The function block ML_WriteCyclicVelocityCtrl generates the following error messages:

- [MLC_TABLE](#), page 343 - 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

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 135,](#)
- [GVL_C_Param_ID, page 135,](#)
- [GVL_M_Param_ID, page 136,](#)
- [GVL_N_Param_ID, page 136,](#)
- [GVL_O_Param_ID, page 137,](#)
- [GVL_P_Param_ID, page 137,](#)
- [GVL_S_Param_ID, page 138](#)

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.

ML_PLCOpen.library

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. 6-95: Parameter offsets

Several axes can be actuated with the control.

That is why the blocks have 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 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 Write Parameter Function Blocks

The MB_Read***Parameter and the MB_Write***Parameter function blocks as well as the MB_GroupRead***Parameter and the MB_GroupWrite***Parameter function blocks use the required data channel of the control and drives for the data transmission.

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)).

Parameters	Number of bytes	Function block	average access time	min	max
S-0-0256	UDINT, 4	MB_ReadParameter	9,8 ms	7ms	23ms
P-0-0072	Liste, 1024	MB_WriteListParameter	232ms	219ms	245ms
P-0-0072	Liste, 1024	MB_ReadListParameter	139ms	135ms	157ms
A-0-0224	REAL, 4	MB_ReadRealParameter	2ms	2ms	2ms
A-0-0101	LREAL, 8	MB_ReadLrealParameter	2ms	2ms	2ms
A-0-0023	UDINT, 4	MB_ReadParameter	2ms	2ms	2ms
C-0-0852	LREAL, 8	MB_WriteLrealParameter	2 ms	2ms	2ms
A-0-3020	LREAL, 8	MB_WriteLrealParameter	32ms	12ms	79ms
A-0-0030	REAL, 4	MB_WriteRealParameter	2ms	2ms	2ms
A-0-0002	STRING	MB_ReadStringParameter	2ms	2ms	2ms

6.10.3 MB_Command

Brief description The function block MB_Command allows a command to be carried out. The command can be started, controlled, observed or aborted.

Library	Range
ML_PLCopen.library	Parameters

Tab. 6-96: Library assignment

Interface description

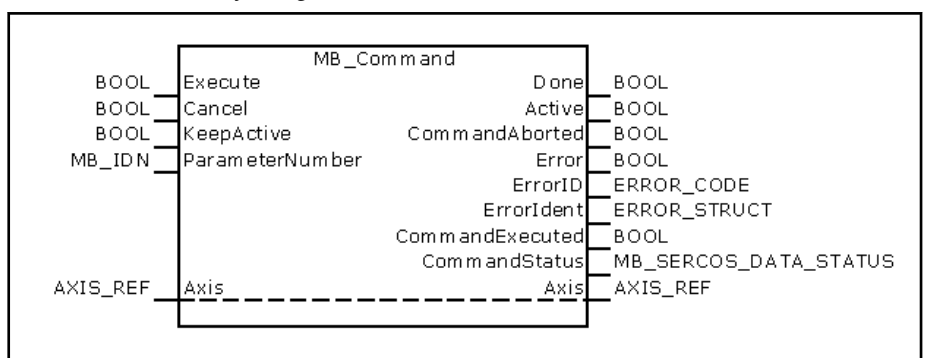


Fig. 6-106: FB MB_Command

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the actual axis
VAR_INPUT	Execute	BOOL	Starts the motion if there is a rising edge

ML_PLCOpen.library

	Name	Type	Comment
	Cancel	BOOL	Aborts the command that is running if there is a rising edge
	KeepActive	BOOL	Keeps the command active after processing the status graph
	Parameter-Number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data runs after preprocessing is completed
	CommandAborted	BOOL	Command successfully aborted and reset
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error
	CommandExecuted	BOOL	Command successfully executed
	Command-Status	MB_SERCOS_DATA_STATUS	Status of the selected parameter

Tab. 6-97: Interface of FB MB_Command

Brief description The function block "MB_Command" starts the execution of the command defined over the input "ParameterNumber" after having enabled processing with "Execute". In general, the command will be automatically reset after processing is completed.

Depending on the respective command, its execution can be canceled any time with the "Cancel" input.

The "KeepActive" input allows the command to stay active after it has been executed. If the input "KeepActive" is later deactivated, the command will be reset automatically.

The successful completion or successful cancellation of a command will be displayed over the outputs "Done" or "CommandAborted" for a period of at least one call.

The active processing of the command will be displayed at the "Active" output and the respective datum at the "CommandStatus" output.

The current status of the command execution will be permanently displayed at the "CommandStatus" output and the completion of the command execution will be signaled at the "CommandExecuted" output.

If an error occurs during the processing of a command, this will be signaled by the "Error" output and by "ErrorID", "ErrorIdent" for a period of 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 minimum/maximum range

Signal-time diagram

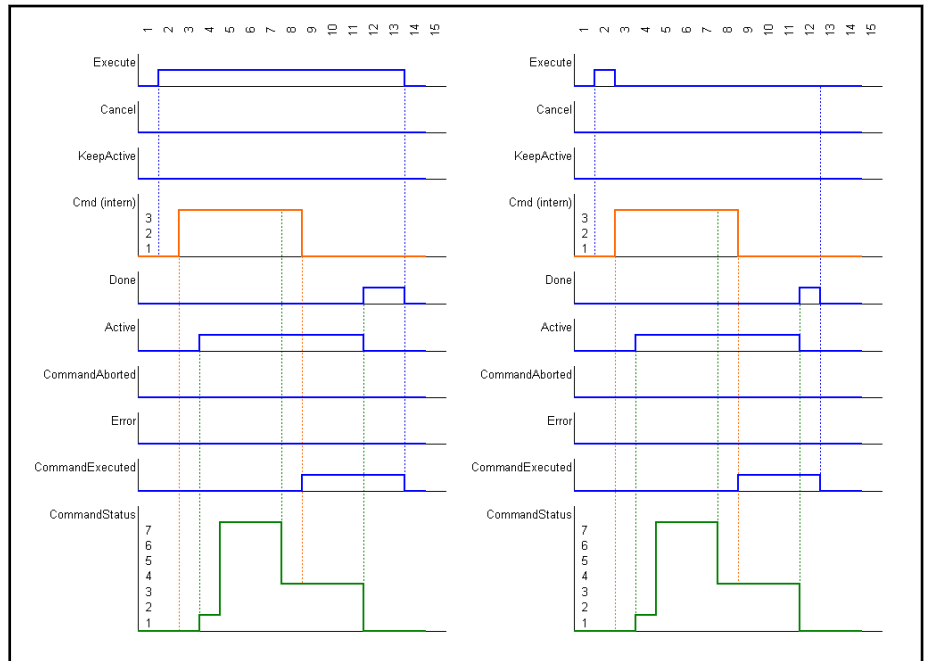


Fig. 6-107: Signal/time diagram for signal flow "Execute-Done"

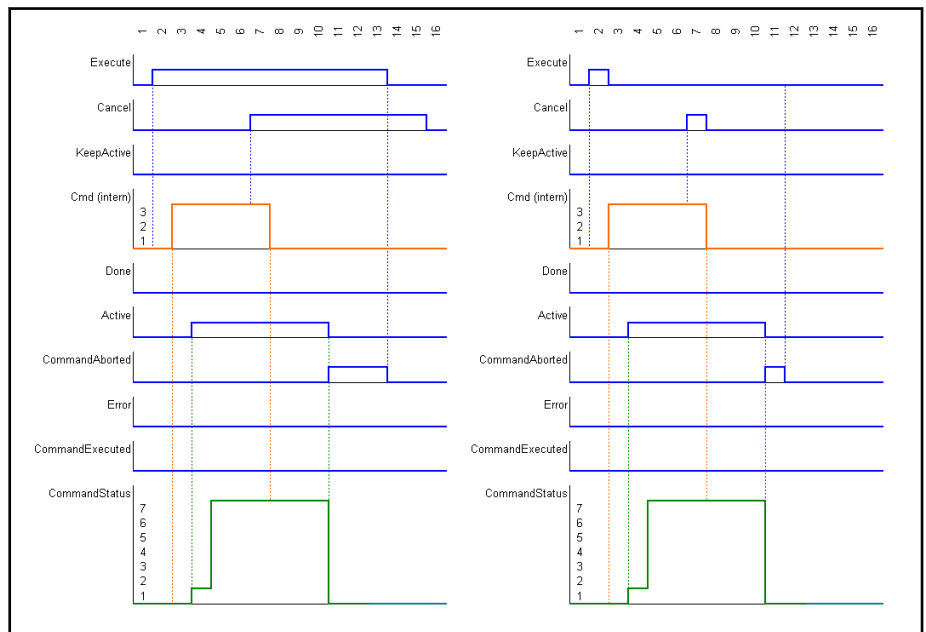


Fig. 6-108: Signal/time diagram for signal flow "Execute-Cancel-CommandAborted"

ML_PLCopen.library

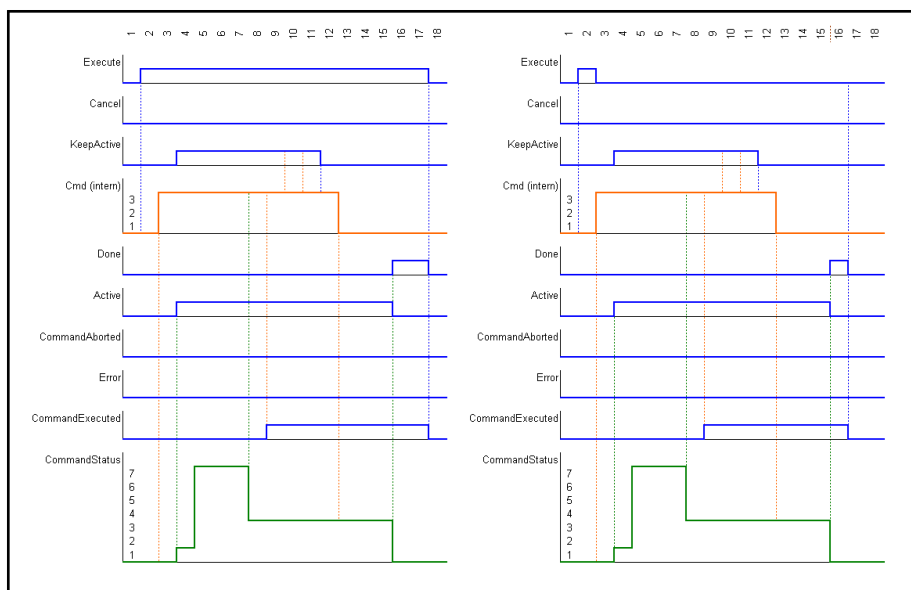


Fig. 6-109: Signal/time diagram for signal flow "Execute-KeepActive-Done"

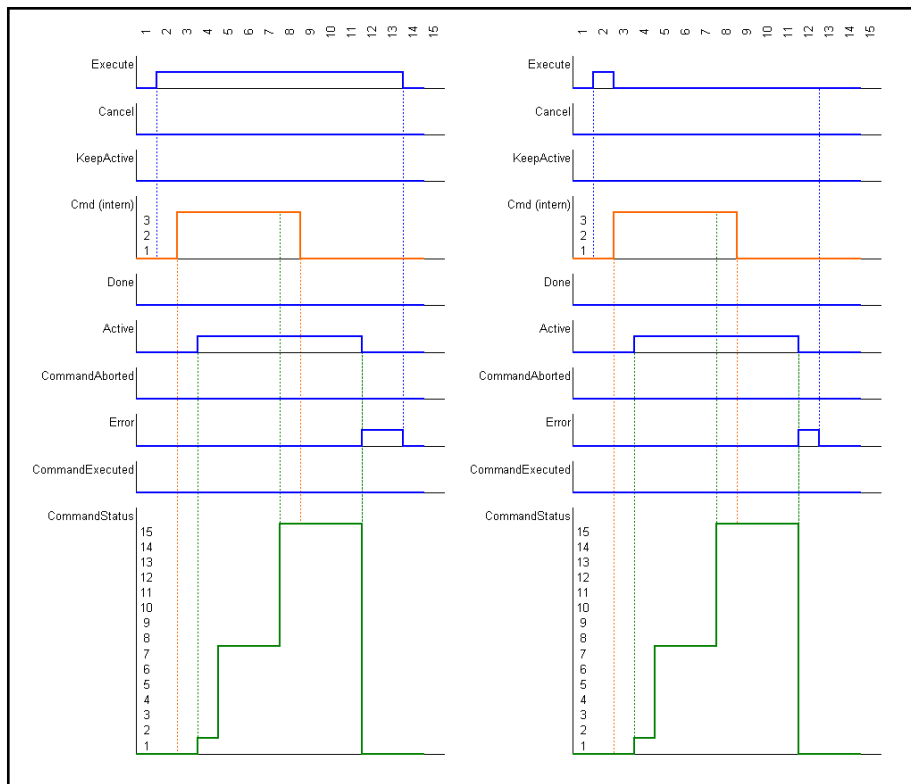


Fig. 6-110: Signal/time diagram for signal flow "Execute-Error"

6.10.4 MB_Command, Special Features of SERCOS-based Axes



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

A command supported by parameters can be triggered in three different ways

(Example A-0-1030 command: clear all axis errors):

- **Starting from IndraWorks:** "Delete error" button in pop-up menu for the axis ⇒ Diagnostics ⇒ "Delete error".
- **With the parameter editor:** Enter the binary value "0000000000000011" (=0x0003) in the parameter. This triggers the error deletion process.
After the error is deleted, the PLC automatically resets the parameter value back to "0000000000000000".
- **From PLC:** via the "MB_Command" block.

Error handling The function block MB_Command generates the following types of error messages:

ERROR_TABLEs

- MLC_TABLE, table number 16#0030,
- 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	16x0006	State	Invalid status of the function block
DEVICE_ERROR	F_RELATED_TABLE	16x1701	16#0000	Error during command processing (control)
DEVICE_ERROR	INDRV_TABLE	Value	16#0000	Error during command processing (drive) Value in S-0-0390.

Tab. 6-98: Error numbers caused by MB_Command

Additionally, error codes of the MLC_TABLE can occur as for

- MB_WriteParameter,
- MB_ReadSercosDataStatus,
- MB_ReadParameter,
- MB_ReadSercosAttribute.

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 the "Enable" signal.

Library	Range
ML_PLCopen.library	Parameters

Tab. 6-99: Library assignment

ML_PLCOpen.library

Interface description

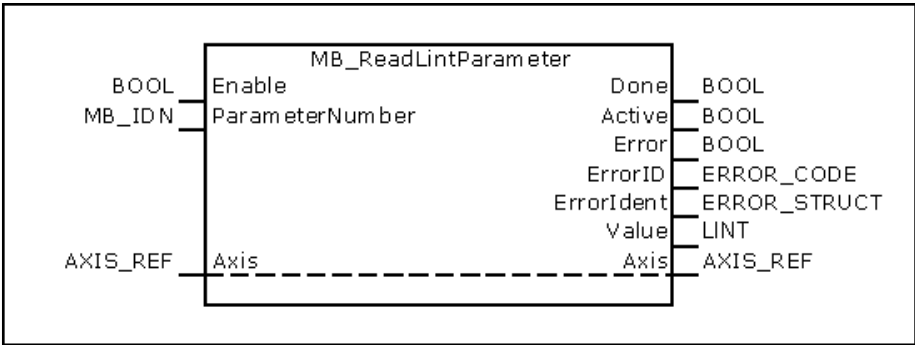


Fig. 6-111: MB_ReadLintParameter FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding 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 desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error
	Value	LINT	8 byte value, represented as LINT

Tab. 6-100: Interface of FB MB_ReadLintParameter

Signal-time diagram

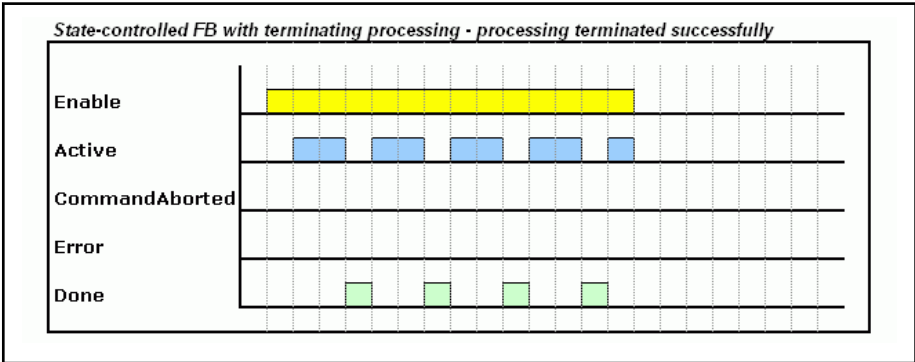


Fig. 6-112: Default signal time diagram

6.10.6 MB_ReadLintParameter, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks](#), page 277.



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object, page 275](#).

Error handling

The function block MB_ReadListParameter generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 343 - error when accessing the axis/the control.

6.10.7 MB_ReadListParameter

Brief description

The function block returns the value of a user-specific parameter. A pointer addressing a memory space of no more than 64 kbytes as defined **by the user** is transferred to the block. In addition, the desired useful data length is transferred (maximum number of readable bytes).

The return value is the number of bytes that has actually been 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 be the same.

Library	Range
ML_PLCopen.library	Parameters

Tab. 6-101: Library assignment

Interface description

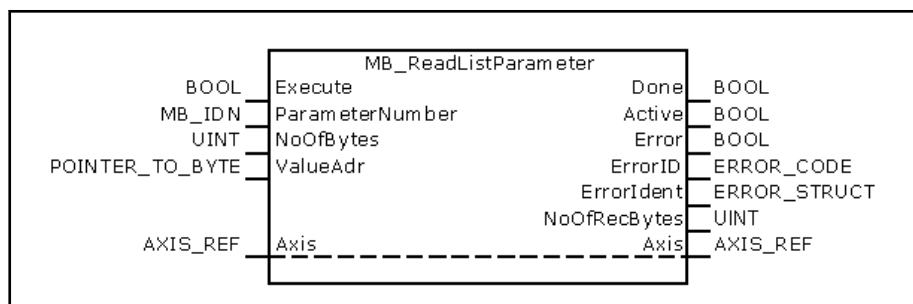


Fig. 6-113: FB MB_ReadListParameter

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the desired control and the object controlled by this control.
VAR_INPUT	Execute	BOOL	The parameter is read, triggered by the positive edge of "Execute".
	Parameter-Number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
	NoOfBytes	UINT	Maximum number of data to be read (in bytes)
	ValueAdr	POINTER TO BYTE	Points to the first address of the area where data read is to be stored (maximum NoOfBytes).

ML_PLCOpen.library

	Name	Type	Comment
VAR_OUTPUT	Done	BOOL	Valid data are available.
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error
	NoOf RecBytes	UINT	Number of bytes actually read

Tab. 6-102: Interface of FB MB_ReadListParameter

Signal-time diagram

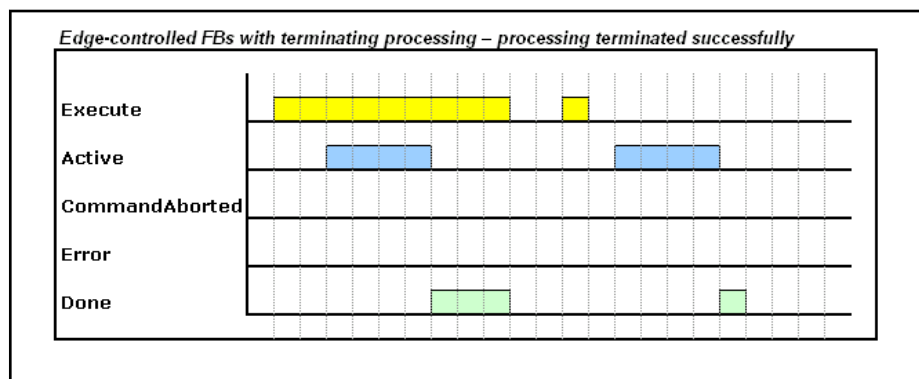


Fig. 6-114: Default signal time diagram

Program example MB_ReadList-Parameter

The program example shows the reading of a list parameter (C-0-2001) of the MLC with 1,024 values (DINT) and the saving to the ReadListArray.

Program:

```

PROGRAM ReadListParameter
VAR IN_OUT
    Axis:          AXIS_REF;          (* drive1 is a AXIS_REF variable*)
END_VAR

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; (* 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 occur *)
    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 plan

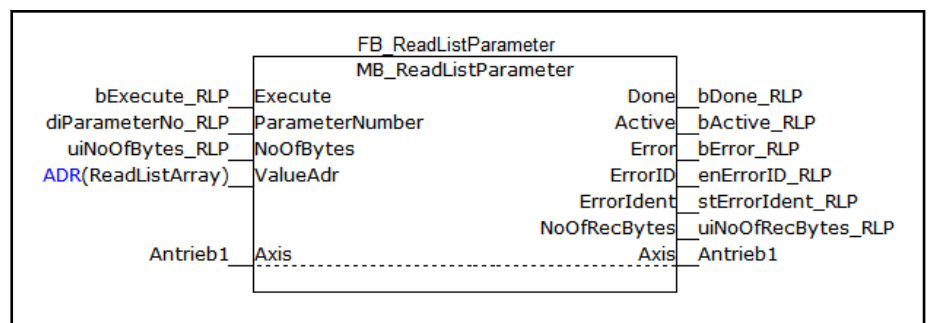


Fig. 6-115: FUP program example ReadListParameter

Program example in structured text

Program:

```

FB_ReadListParameter (
  Execute:=      bExecute_RLP,      (* Start ReadListParameter *)
  ParameterNumber:= diParameterNo_RLP, (* Parameter ControlCAM1 der MLC *)
  NoOfBytes:=    uiNoOfBytes_RLP,    (* DINT := 4 Bytes *)
  ValueAdr:=     ADR(ReadListArray), (* Address of the ReadListArray *)
  Axis:=         drive1,             (* drive address *)
  Done=>         bDone_RLP,          (* TRUE := FB ready *)
  Active=>       bActive_RLP,        (* The FB works *)
  Error=>        bError_RLP,         (* Error occur *)
  ErrorId=>      enErrorID_RLP,      (* Error class *)
  ErrorIdent=>   stErrorIdent_RLP:   (* Error definition *)
  NoOfRecBytes=> uiNoOfRecBytes_RLP); (* Number of received bytes *)

```

6.10.8 MB_ReadListParameter, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks](#), page 277.



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

Error handling

The function block MB_ReadListParameter generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#), page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#), page 343 - error when accessing the axis/the control.

6.10.9 MB_ReadLrealListParameter

Brief description

The function block returns the value of a user-specific parameter. A pointer addressing a memory space of no more than 64 kbytes as defined **by the user** is transferred to the block. In addition, the desired useful data length is transferred (maximum number of readable bytes).

The return value is the number of bytes that has actually been read.

ML_PLCOpen.library



In contrast to MB_ReadListParameter where an ARRAY of BYTE with n bytes is read, here an ARRAY of LREAL with m BYTES, or m/8 LREAL is read.

Library	Range
ML_PLCOpen.library	Parameters

Tab. 6-103: Library assignment

Interface description

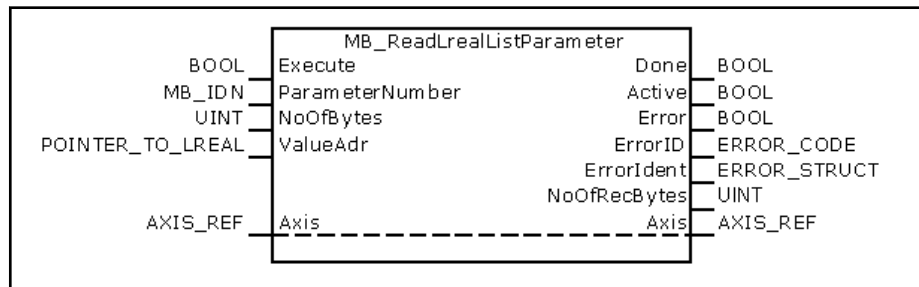


Fig. 6-116: MB_ReadLrealListParameter

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the desired control and the object controlled by this control.
VAR_INPUT	Execute	BOOL	The parameter is read, triggered by the positive edge of "Execute".
	Parameter-Number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
	NoOfBytes	UINT	Maximum number of data to be read (in bytes)
	ValueAdr	POINTER TO LREAL	Points to the first address of the area where data read is to be stored (maximum NoOfBytes).
VAR_OUTPUT	Done	BOOL	Valid data are available.
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error
	NoOf RecBytes	UINT	Number of bytes actually read

Tab. 6-104: Interface of FB MB_ReadLrealListParameter

Signal-time diagram

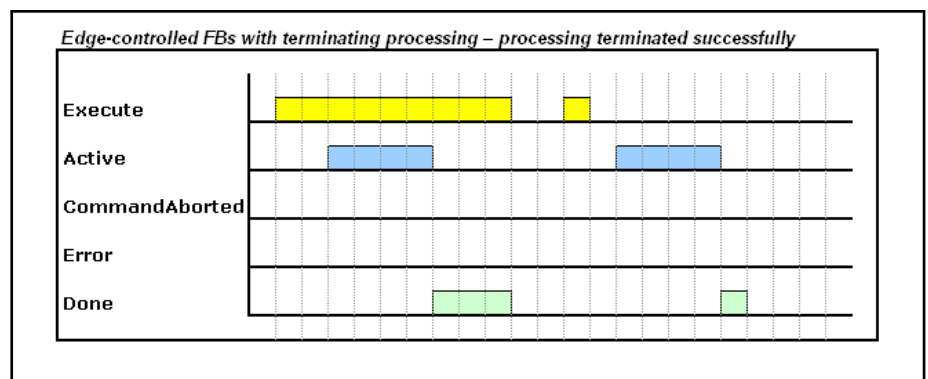


Fig. 6-117: Default signal time diagram

Program example MB_ReadLreal-ListParameter

The program example shows the reading of a list parameter (A-0-3027) of the MLC with 1,024 values (LREAL) and the saving to the ReadListArray.

Program:

```
PROGRAM ReadLrealListParameter
VAR IN_OUT
    Axis:          AXIS_REF;          (* drive1 is a AXIS_REF variable*)
END_VAR

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; (* List of the initial slave axis velocities*)
    uiNoOfBytes_RLP:  UINT := 8192;    (* LREAL := 8 bytes *)
    bDone_RLP:        BOOL;            (* TRUE := FB ready *)
    bActive_RLP:      BOOL;            (* The FB works *)
    bError_RLP:       BOOL;            (* Error occur *)
    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 plan

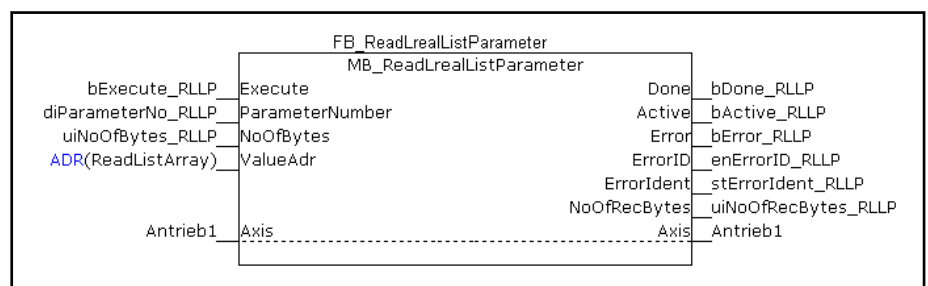


Fig. 6-118:

Program example in structured text

Program:

```
FB_ReadLrealListParameter(
    Execute:= bExecute_RLP,          (* Start ReadLrealListParameter *)
    ParameterNumber:= diParameterNo_RLP, (* List of the initial slave axis velocities*)
    NoOfBytes:= uiNoOfBytes_RLP,    (* LREAL := 8 bytes *)
    ValueAdr:= ADR(ReadListArray),  (* Address of the ReadListArray *)
    Axis:= drive1,                  (* drive address *)
```

ML_PLCOpen.library

```
Done=>          bDone_RLP,          (* TRUE := FB ready          *)
Active=>        bActive_RLP,        (* The FB works              *)
Error=>         bError_RLP,         (* Error occur               *)
ErrorId=>       enErrorID_RLP,      (* Error class               *)
ErrorIdent=>    stErrorIdent_RLP:   (* Error definition          *)
NoOfRecBytes=> uiNoOfRecBytes_RLP); (* Number of received bytes  *)
```

6.10.10 MB_ReadLrealListParameter, Special Features of SERCOS-based Axes

 For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks](#), page 277.

 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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

Error handling The function block MB_ReadLrealListParameter generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 343 - error when accessing the axis/the control.

6.10.11 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 the "Enable" signal.

Library	Range
ML_PLCOpen.library	Parameters

Tab. 6-105: Library assignment

Interface description

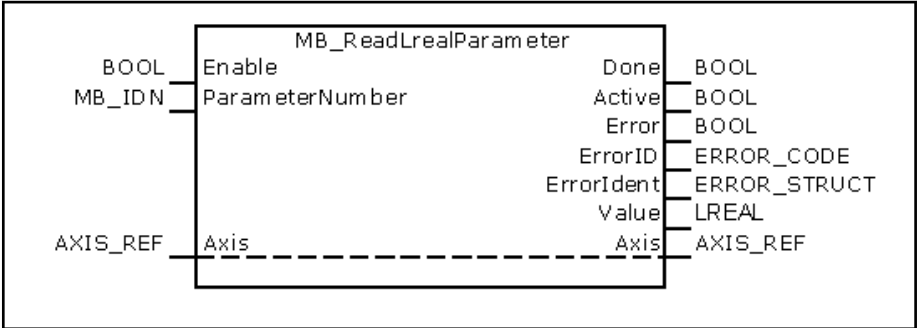


Fig. 6-119: MB_ReadLrealParameter FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the desired control and the object controlled by this control.
VAR_INPUT	Enable	BOOL	Continuous reading of the parameter as long as "Enable" is TRUE.
	Parameter-Number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error
	Value	LREAL	8 byte value represented as LREAL number

Tab. 6-106: Interface of FB MB_ReadLrealParameter

Signal-time diagram

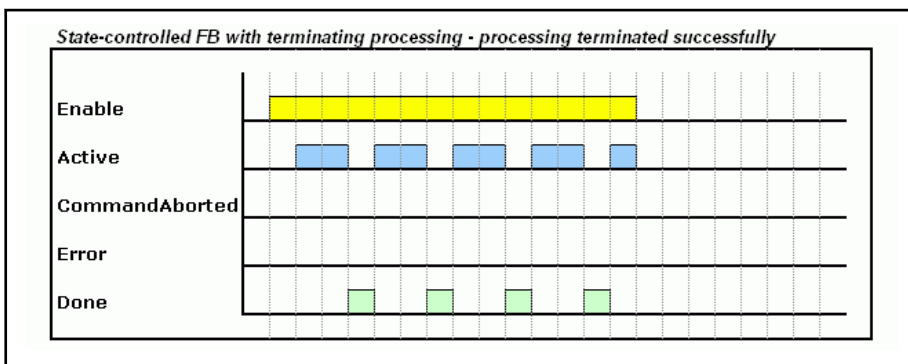


Fig. 6-120: Default signal time diagram

6.10.12 MB_ReadLrealParameter, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks, page 277](#).



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object, page 275](#).

Error handling

The function block MB_ReadLrealParameter generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , [page 340](#) - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 343](#) - error when accessing the axis/the control.

ML_PLCOpen.library

6.10.13 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
ML_PLCOpen.library	Parameters

Tab. 6-107: Library assignment

Interface description

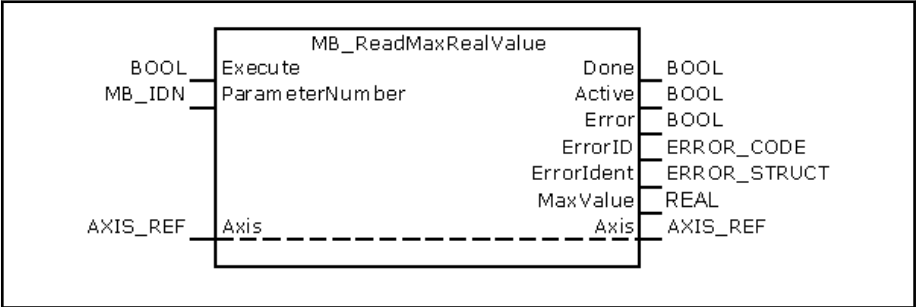


Fig. 6-121: MB_ReadMaxRealValue FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	The input variables (0/1 transition) are registered with a positive edge at "Execute". Then, reading runs irrespective of the variable value.
	Parameter number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
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 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. 6-108: Interface of the MB_ReadRealParameter FB

Brief Description

Signal-time diagram

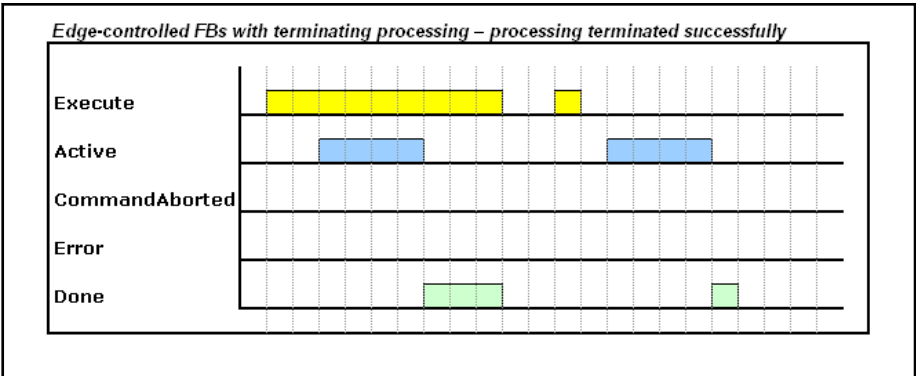


Fig. 6-122: Default signal-time diagram

6.10.14 MB_ReadMaxRealValue, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks](#), page 277.



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

Error handling

The function block MB_ReadMaxRealValue generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 343 - error when accessing the axis/the control.

6.10.15 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
ML_PLCopen.library	Parameters

Tab. 6-109: Library assignment

ML_PLCOpen.library

Interface description

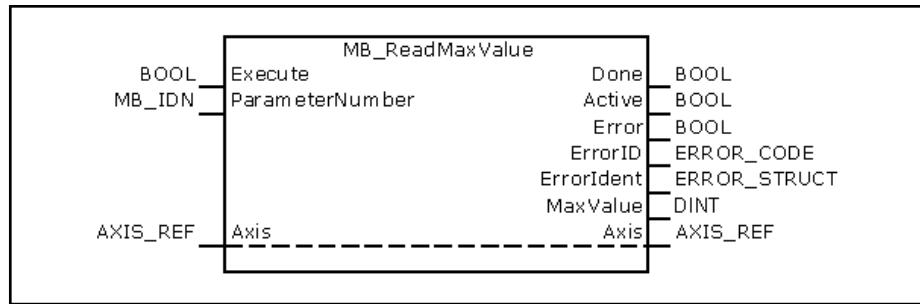


Fig. 6-123: MB_ReadMaxValue FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	The input variables (0/1 transition) are registered with a positive edge at "Execute". Then, reading runs irrespective of the variable value.
	Parameter number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
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 when the function block instance was called
	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. 6-110: Interface of the MB_ReadMaxValue FB

Brief Description Signal-time diagram

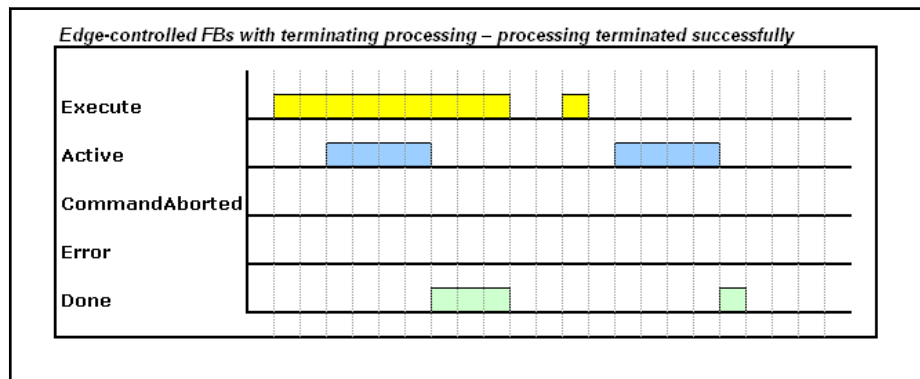


Fig. 6-124: Default signal-time diagram

6.10.16 MB_ReadMaxValue, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks](#), page 277.



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object, page 275](#).

Error handling

The function block MB_ReadMaxValue generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , [page 340](#) - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , [page 343](#) - error when accessing the axis/the control.

6.10.17 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
ML_PLCopen.library	Parameters

Tab. 6-111: Library assignment

Interface description

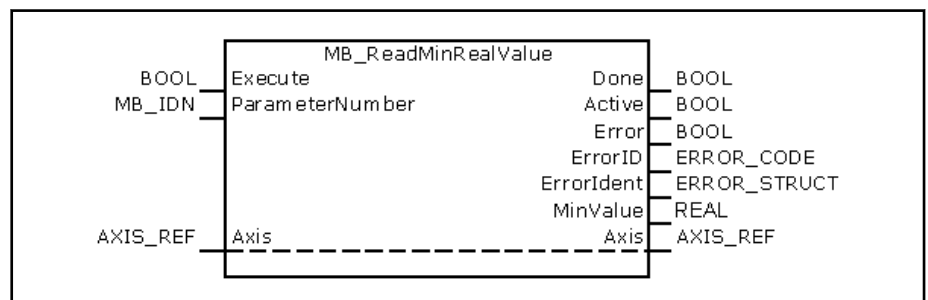


Fig. 6-125: MB_ReadMinRealValue FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	The input variables (0/1 transition) are registered with a positive edge at "Execute". Then, reading runs irrespective of the variable value.
	Parameter number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
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 when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause

ML_PLCOpen.library

	Name	Type	Comment
	ErrorIdent	ERROR_STRUCT	Detailed error information
	MinValue	REAL	If "Done" = TRUE, "MinValue" indicates the minimum input value allowed of the parameter selected.

Tab. 6-112: Interface of the MB_ReadMinRealValue FB

Brief Description

Signal-time diagram

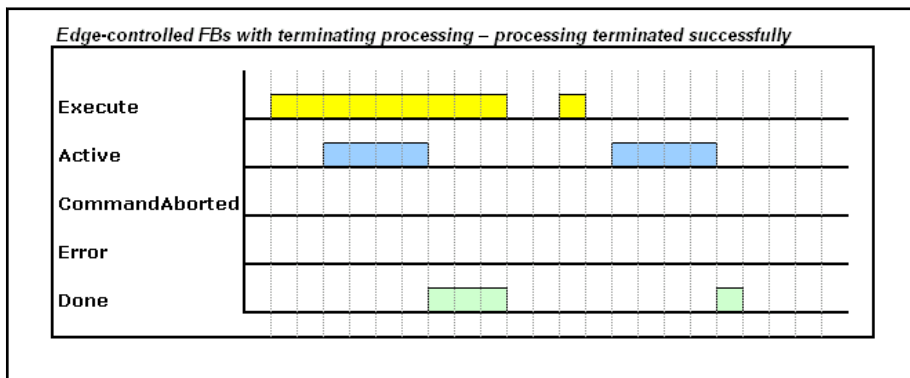


Fig. 6-126: Default signal-time diagram

6.10.18 MB_ReadMinRealValue, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks](#), page 277.



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

Error handling

The function block MB_ReadMinRealValue generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#), page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#), page 343 - error when accessing the axis/the control.

6.10.19 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
ML_PLCOpen.library	Parameters

Tab. 6-113: Library assignment

Interface description

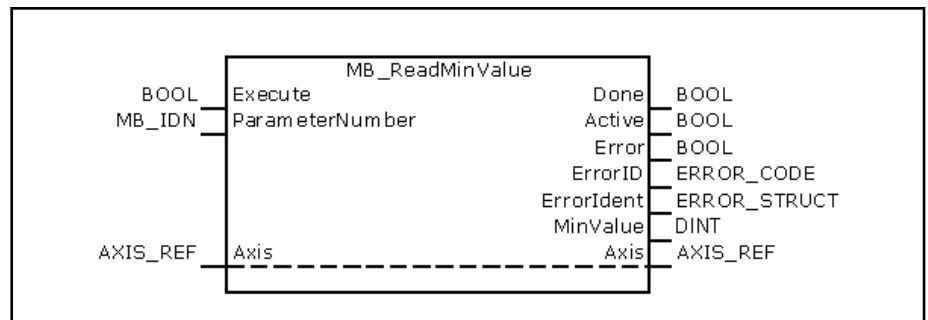


Fig. 6-127: MB_ReadMinValue FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	The input variables (0/1 transition) are registered with a positive edge at "Execute". Then, reading runs irrespective of the variable value.
	Parameter number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
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 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. 6-114: Interface of the MB_ReadMinValue FB

Brief Description Signal-time diagram

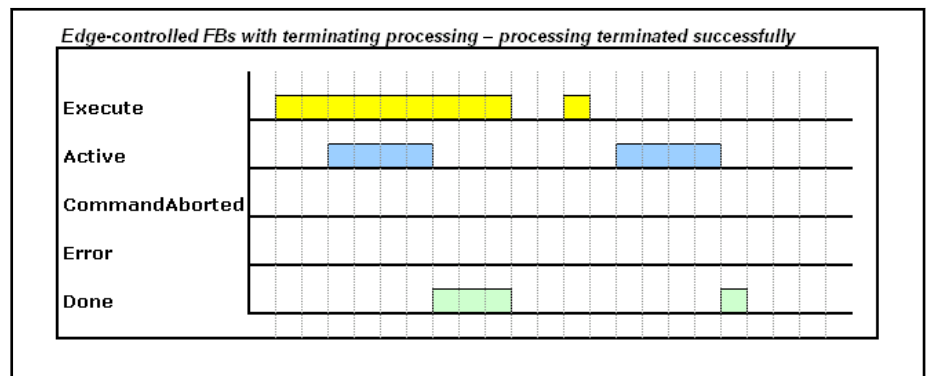


Fig. 6-128: Default signal-time diagram

6.10.20 MB_ReadMinValue, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks, page 277](#).

ML_PLCOpen.library



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

Error handling

The function block MB_ReadMinValue generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 343 - error when accessing the axis/the control.

6.10.21 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".

Library	Range
ML_PLCOpen.library	Parameters

Tab. 6-115: Library assignment

Interface description

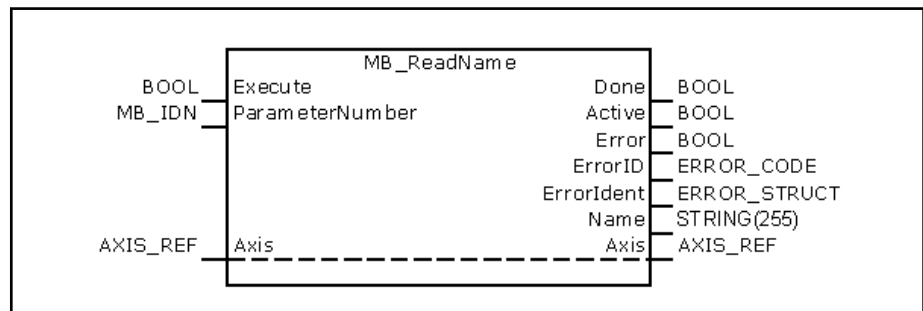


Fig. 6-129: MB_ReadName FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	The input variables (0/1 transition) are registered with a positive edge at "Execute". Then, reading runs irrespective of the variable value.
	Parameter number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
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 when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause

	Name	Type	Comment
	ErrorIdent	ERROR_STRUCT	Detailed error information
	Name	STRING(255)	If "Done" = TRUE, "MinValue" indicates the name of the parameter selected.

Tab. 6-116: Interface of MB_ReadName FB

Signal-time diagram

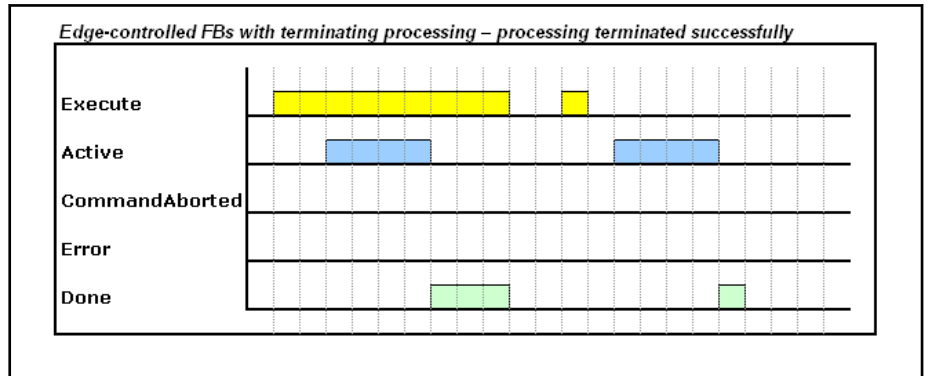


Fig. 6-130: Default signal-time diagram

6.10.22 MB_ReadName, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks](#), page 277.



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

Error handling

The function block MB_ReadName generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 343 - error when accessing the axis/the control.

6.10.23 MB_ReadParameter

Brief description

The function block returns the value of a user-specific parameter. The return value is represented as a 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" signal.

Library	Range
ML_PLCopen.library	Parameters

Tab. 6-117: Library assignment

ML_PLCOpen.library

Interface description

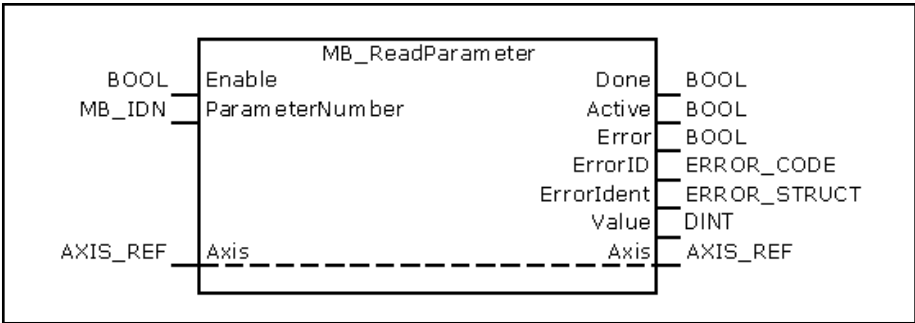


Fig. 6-131: FB MB_ReadParameter

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding 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 desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error
	Value	DINT	4 byte value, represented as DINT

Tab. 6-118: Interface of FB MB_ReadParameter

Signal-time diagram

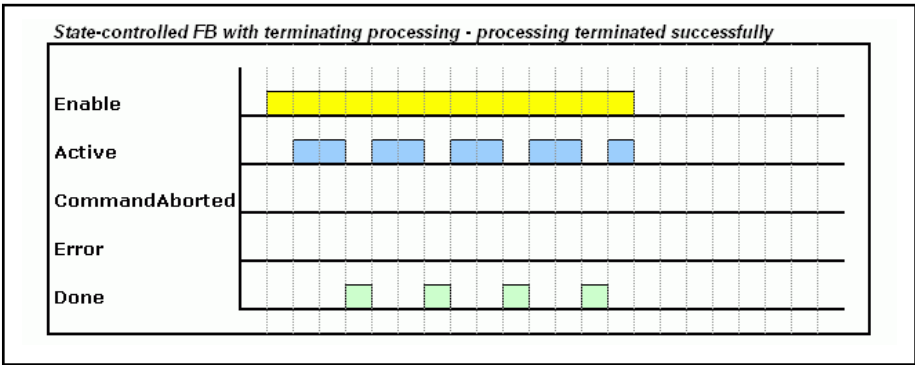


Fig. 6-132: Default signal time diagram

6.10.24 MB_ReadParameter, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks, page 277](#).



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object, page 275](#).

Error handling

The function block MB_ReadParameter generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 343 - error when accessing the axis/the control.

6.10.25 MB_ReadRealParameter

Brief description

The function block returns the value of a user-specific parameter. The return value is represented as a 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 the "Enable" signal.

Library	Range
ML_PLCopen.library	Parameters

Tab. 6-119: Library assignment

Interface description

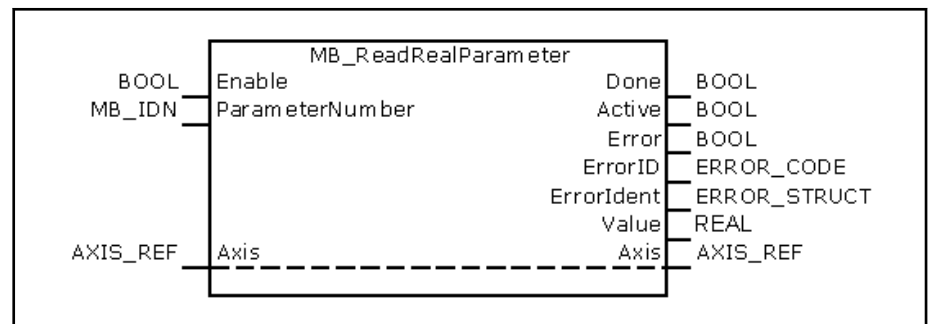


Fig. 6-133: FB MB_ReadRealParameter

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the desired control and the object controlled by this control.
VAR_INPUT	Enable	BOOL	Continuous reading of the parameter as long as "Enable" is TRUE.
	Parameter-Number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.

ML_PLCOpen.library

	Name	Type	Comment
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error
	Value	REAL	4 byte value represented as REAL number

Tab. 6-120: Interface of FB MB_ReadRealParameter

Signal-time diagram

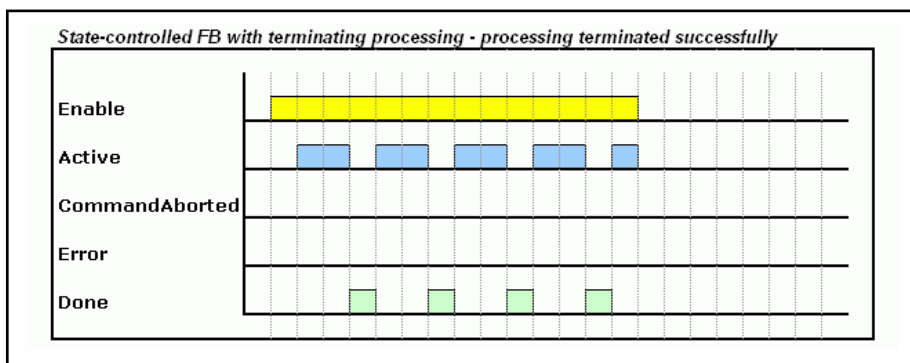


Fig. 6-134: Default signal time diagram

6.10.26 MB_ReadRealParameter, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks](#), page 277.



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

Error handling

The function block MB_ReadRealParameter generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#), page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#), page 343 - error when accessing the axis/the control.

6.10.27 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 the "Execute" signal.

Library	Range
ML_PLCOpen.library	Parameters

Tab. 6-121: Library assignment

Interface description

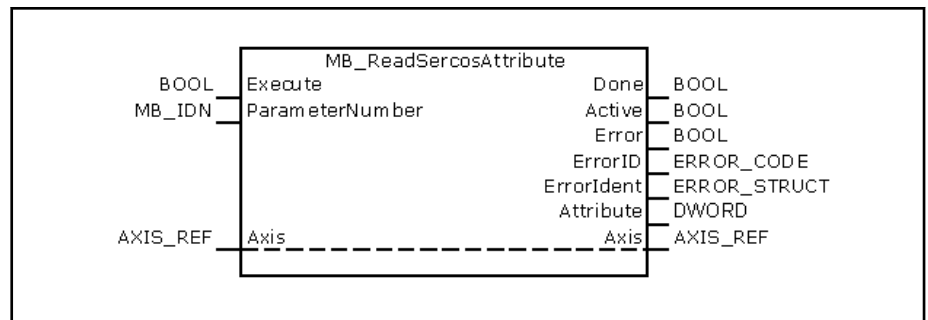


Fig. 6-135: FB MB_ReadSercosAttribute

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the actual axis
VAR_INPUT	Execute	BOOL	Reading of the parameter attributes after 0-1 transition.
	Parameter-Number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error
	Attribute	DWORD	If "Done"=TRUE, the 4 byte value contains SERCOS attributes of the parameter

Tab. 6-122: Interface of FB MB_ReadSercosAttribute

Dekodierung des Attributs Signal-time diagram

The "ML_Base.library" supports this FB with the FB "IL_SercosAttribute".

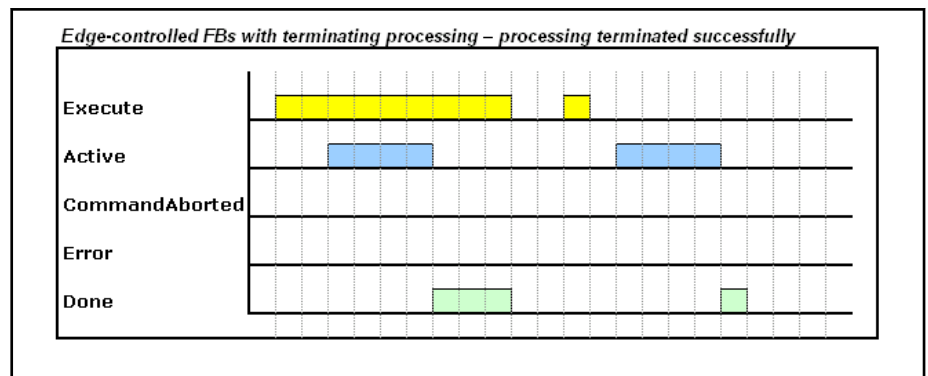


Fig. 6-136: Default signal time diagram

6.10.28 MB_ReadSercosAttribute, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks, page 277](#).

ML_PLCopen.library



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

Error handling

The function block MB_ReadSercosAttribute generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 343 - error when accessing the axis/the control.

In addition, the following can be displayed:

6.10.29 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 the "Enable" signal.

Library	Range
ML_PLCopen.library	Parameters

Tab. 6-123: Library assignment

Interface description

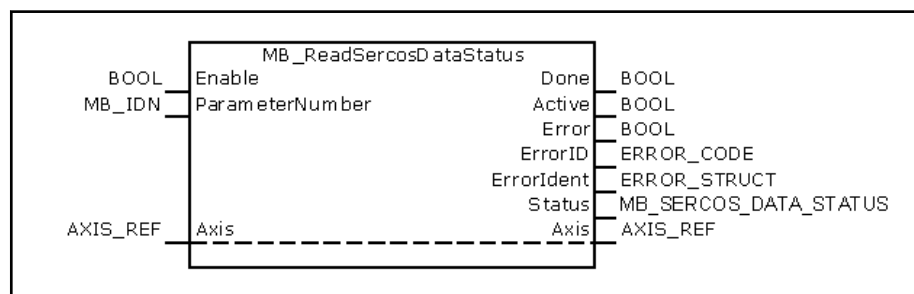


Fig. 6-137: FB MB_ReadSercosDataStatus

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the actual axis
VAR_INPUT	Enable	BOOL	Continuous reading of the parameter as long as "Enable" is TRUE.
	Parameter-Number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.
	ErrorID	ERROR_CODE	Brief indication of the cause for error

	Name	Type	Comment
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error
	Status	MB_SERCOS_DATA_STATUS	Status of the selected parameter

Tab. 6-124: Interface of FB MB_ReadSercosDataStatus

Brief description With this function block and a [MB_WriteParameter](#) page 312, it is possible to trigger drive commands and to track their processing:

- 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

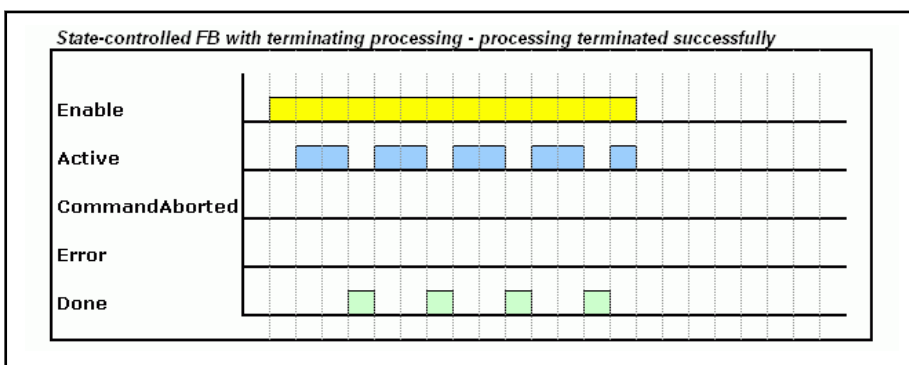


Fig. 6-138: Default signal time diagram

6.10.30 MB_ReadSercosDataStatus, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks](#), page 277.



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

Error handling

The function block MB_ReadSercosDataStatus generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 343 - error when accessing the axis/the control.

ML_PLCOpen.library

6.10.31 MB_ReadStringParameter

Brief description The function block returns the value of a user-specific parameter. The return value is displayed as a chain of characters (STRING) in accordance with its actual length with a maximum of 255 characters.

Library	Range
ML_PLCOpen.library	Parameters

Tab. 6-125: Library assignment

Interface description

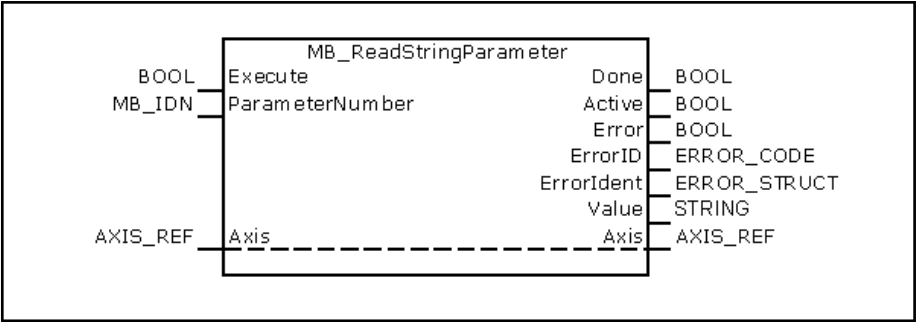


Fig. 6-139: FB MB_ReadStringParameter

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the desired control and the object controlled by this control.
VAR_INPUT	Execute	BOOL	Starts reading the text of the parameter if there is a rising edge
	Parameter-Number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error
	Value	STRING	Chain of characters with a maximum of 255 characters

Tab. 6-126: Interface of FB MB_ReadStringParameter

Signal-time diagram

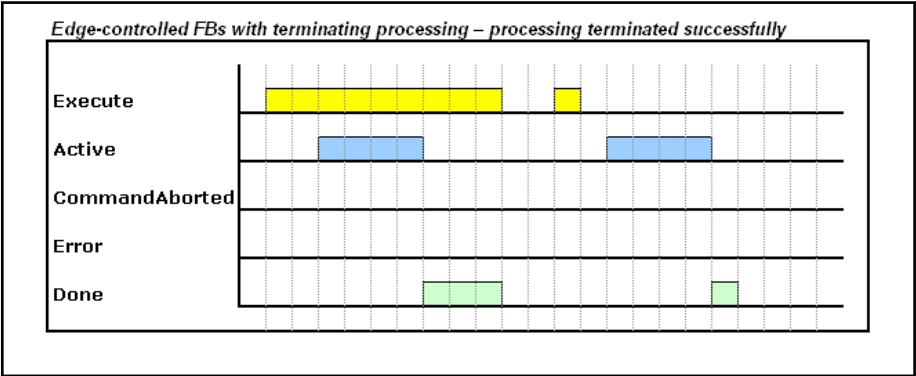


Fig. 6-140: Default signal time diagram

6.10.32 MB_ReadStringParameter, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks](#), page 277.



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

Error handling

The function block MB_ReadStringParameter generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 343 - error when accessing the axis/the control.

6.10.33 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".

Library	Range
ML_PLCopen.library	Parameters

Tab. 6-127: Library assignment

Interface description

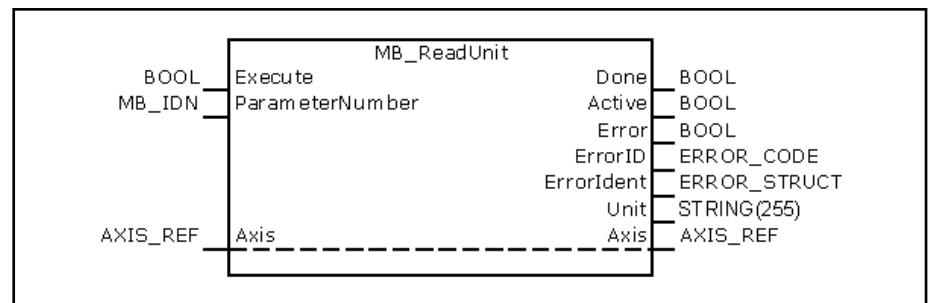


Fig. 6-141: MB_ReadUnit FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	The input variables (0/1 transition) are registered with a positive edge at "Execute". Then, reading runs irrespective of the variable value.
	Parameter number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)

ML_PLCOpen.library

	Name	Type	Comment
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 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.

Tab. 6-128: Interface of the MC_ReadStatus FB

Signal-time diagram

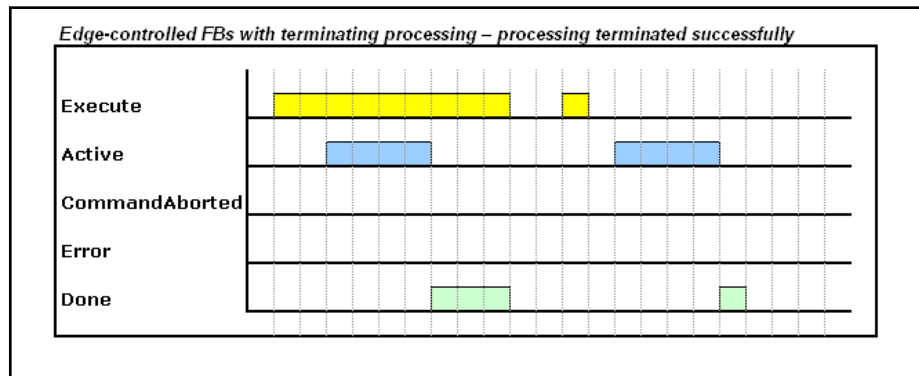


Fig. 6-142: Default signal-time diagram

6.10.34 MB_ReadUnit, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks](#), page 277.



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

Error handling

The function block MB_ReadUnit generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#), page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#), page 343 - error when accessing the axis/the control.

6.10.35 MB_WriteLintParameter

Brief description The function block writes a value in a parameter specified by the user.

Library	Range
ML_PLCopen.library	Parameters

Tab. 6-129: Library assignment

Interface description

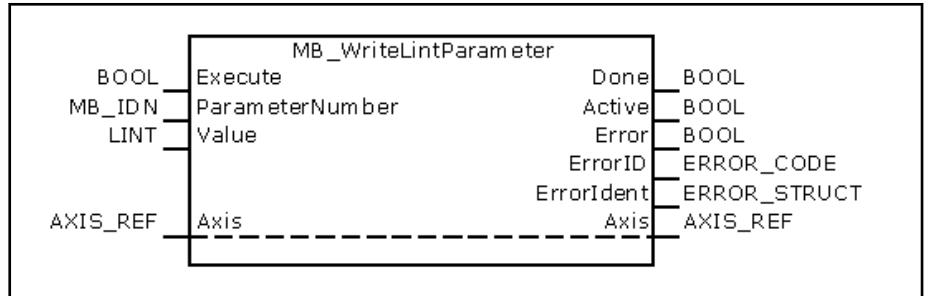


Fig. 6-143: MB_WriteLintParameter

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the desired control and the object controlled by this control.
VAR_INPUT	Execute	BOOL	Starts writing the value that occurs when there is a rising edge to the parameter.
	Parameter-Number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
	Value	LINT	Value to be written to the parameter
VAR_OUTPUT	Done	BOOL	The value has been written successfully to the control/the drive.
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error

Tab. 6-130: Interface of FB MB_WriteLintParameter

Signal-time diagram

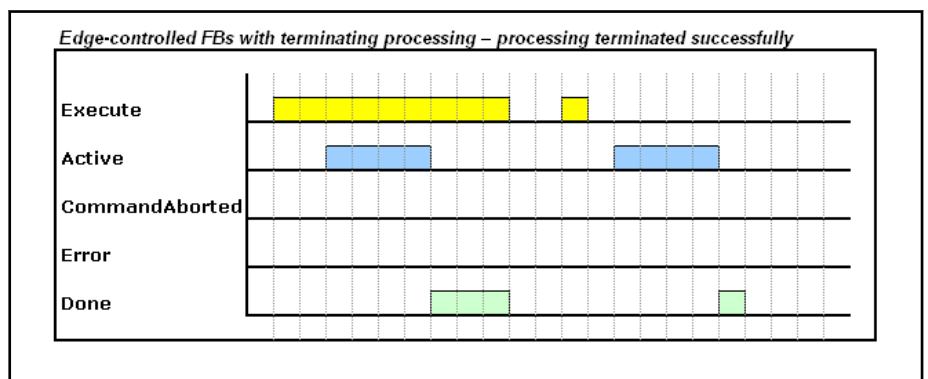


Fig. 6-144: Default signal time diagram

6.10.36 MB_WriteLintParameter, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks, page 277](#).

ML_PLCopen.library



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

Error handling

The function block MB_WriteListParameter generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 343 - error when accessing the axis/the control.

6.10.37 MB_WriteListParameter

Brief description

This function block writes memory contents having a defined length (NoOf-Bytes) and provided with the starting address (ValueAdr) to a user-specific parameter.



The specified number of bytes must be exactly the same as the data volume to be transferred!

If this number is too great, 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 be the same.

Library	Range
ML_PLCopen.library	Parameters

Tab. 6-131: Library assignment

Interface description

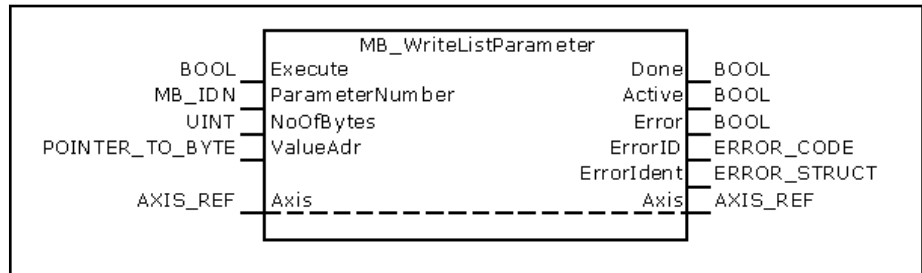


Fig. 6-145: FB MB_WriteListParameter

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the desired control and the object controlled by this control.
VAR_INPUT	Execute	BOOL	Starts writing the value that occurs when there is a rising edge to the parameter.
	Parameter-Number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
	NoOfBytes	UINT	Number of bytes to be written

	Name	Type	Comment
	ValueAdr	POINTER TO BYTE	Pointer to the start address of the memory where the data to be written are saved.
VAR_OUTPUT	Done	BOOL	The value has been written successfully to the control/the drive.
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error

Tab. 6-132: Interface of FB MB_WriteListParameter

Signal-time diagram

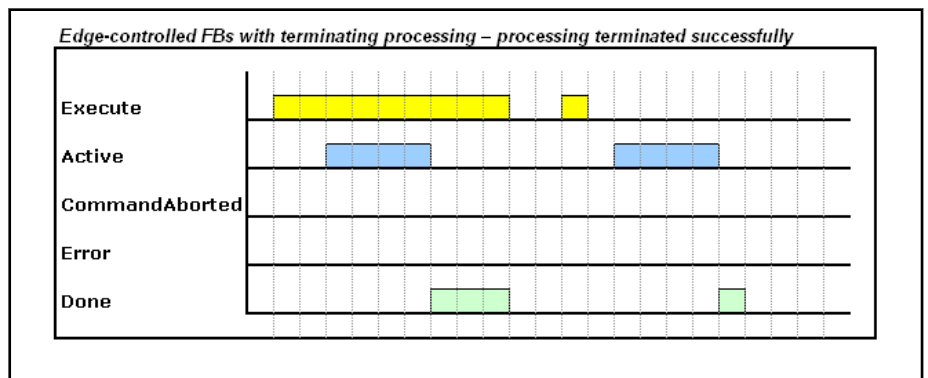


Fig. 6-146: Default signal time diagram

Program Example MB_WriteList-Parameter

This program example shows the writing of the list parameter (C-0-2001) of the MLC with 1,024 values (DINT) from the WriteListArray.

Program:

```

PROGRAM WriteListParameter
VAR IN_OUT
    Axis:                AXIS_REF;                (* drive1 is a AXIS_REF variable*)
END_VAR

VAR
    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:= FB_C_0_2001;    (* Parameter ControlCAM1 der MLC *)
    uiNoOfBytes_WLP:      UINT := 4096;           (* DINT := 4 Bytes *)
    bDone_WLP:           BOOL;                    (* TRUE := FB ready *)
    bActive_WLP:         BOOL;                    (* The FB works *)
    bError_WLP:          BOOL;                    (* Error occur *)
    enErrorID_WLP:        ERROR_CODE;             (* Error class *)
    stErrorIdent_WLP:     ERROR_STRUCT;           (* Error definition *)
END_VAR

```

Program example in the function plan

ML_PLCOpen.library

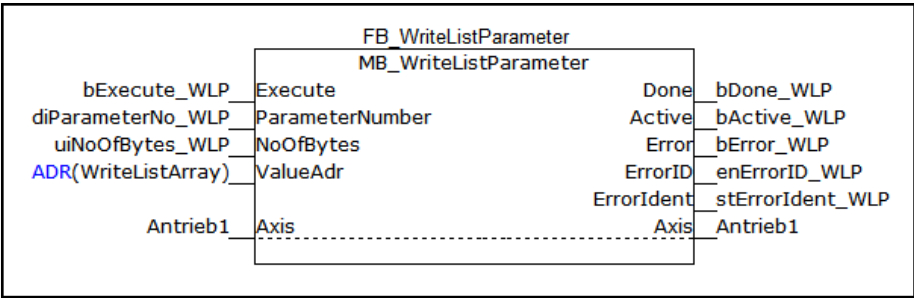


Fig. 6-147: FUP program example WriteListParameter

Program example in structured text

Program:

```
FB_WriteListParameter(  
  Execute:=      bExecute_WLP,      (* Start WriteListParameter *)  
  ParameterNumber:= diParameterNo_WLP, (* Parameter ControlCAM1 der MLC *)  
  NoOfBytes:=    uiNoOfBytes_WLP,    (* DINT := 4 Bytes *)  
  ValueAdr:=     ADR(WriteListArray), (* Adresse des WriteListArray *)  
  Axis:=         drive1,             (* drive address *)  
  Done=>         bDone_WLP,          (* TRUE := FB ready *)  
  Active=>       bActive_WLP,        (* The FB works *)  
  Error=>        bError_WLP,         (* Error occur *)  
  ErrorId=>      enErrorID_WLP,      (* Error class *)  
  ErrorIdent=>   stErrorIdent_WLP:   (* Error definition *)  
)
```

6.10.38 MB_WriteListParameter, Special Features of SERCOS-based Axes

-  For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks](#), page 277.
-  For the assignment
 - of the axis input with A, C, M, N O parameters and/or
 - of the axis group input with K parameterssee [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

- Error handling
- The function block MB_WriteListParameter generates the following types of error messages:
- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
 - [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 343 - error when accessing the axis/the control.

6.10.39 MB_WriteLrealParameter

Short Description This function block writes a value to a user-specific parameter.

Library	Range
ML_PLCOpen.library	Parameter

Tab. 6-133: Library assignment

Interface Description

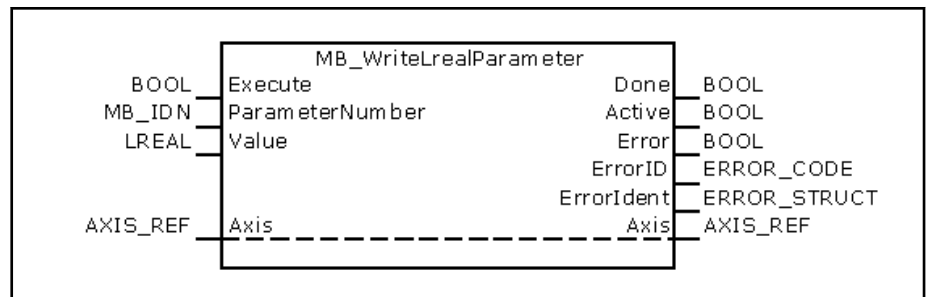


Fig. 6-148: FB MB_WriteRealParameter

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information on the desired control and the object controlled by it.
VAR_INPUT	Execute	BOOL	Starts writing the value that occurs at the rising edge to the parameter.
	Parameter-Number	MB_IDN	Number of the parameter desired, (refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
	Value	LREAL	Value to be written to the parameter
VAR_OUTPUT	Done	BOOL	The value has been written successfully to the control/the drive.
	Active	BOOL	Data processing runs after preprocessing has been completed
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 6-134: Interface of the MB_WriteRealParameter FB

Signal-time diagram

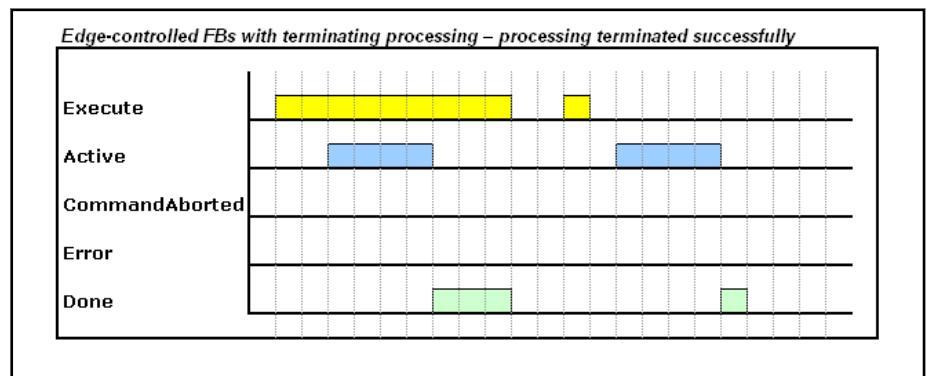


Fig. 6-149: Default signal-time diagram

6.10.40 MB_WriteRealParameter, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks, page 277](#).

ML_PLCOpen.library



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

Error handling

The MB_WriteRealParameter FB generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 343 - error when accessing the axis/the control.

6.10.41 MB_WriteParameter

Brief description The function block writes a value in a parameter specified by the user.

Library	Range
ML_PLCOpen.library	Parameters

Tab. 6-135: Library assignment

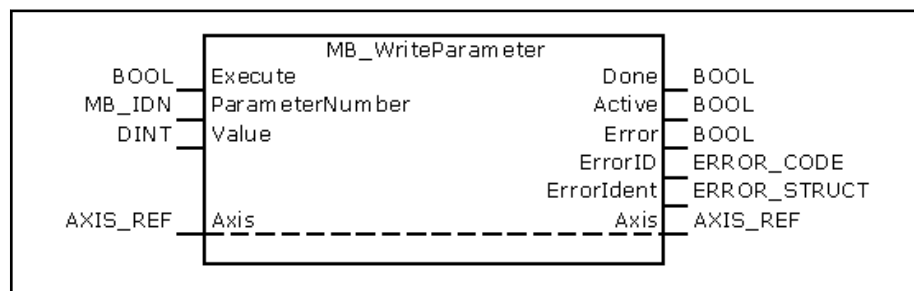
Interface description

Fig. 6-150: FB MB_WriteParameter

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the desired control and the object controlled by this control.
VAR_INPUT	Execute	BOOL	Starts writing the value that occurs when there is a rising edge to the parameter.
	Parameter-Number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
	Value	DINT	Value to be written to the parameter
VAR_OUTPUT	Done	BOOL	The value has been written successfully to the control/the drive.
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error

Tab. 6-136: Interface of FB MB_WriteParameter

Signal-time diagram

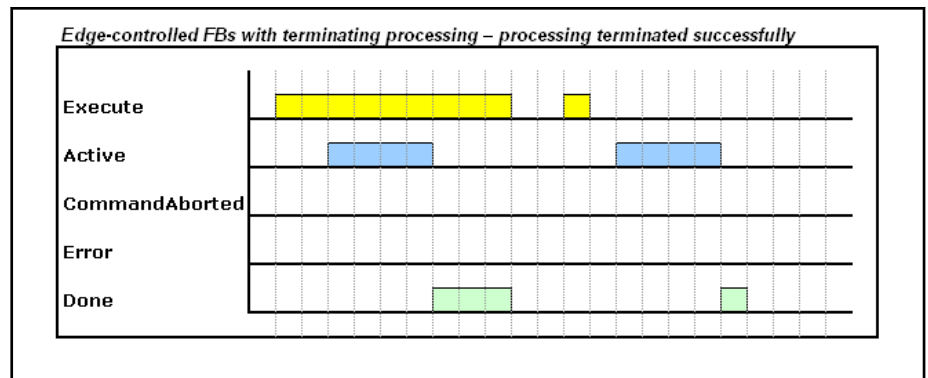


Fig. 6-151: Default signal time diagram

6.10.42 MB_WriteParameter, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks](#), page 277.



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

Error handling

The function block MB_WriteParameter generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#), page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#), page 343 - error when accessing the axis/the control.

6.10.43 MB_WriteRealParameter

Brief description This function block writes a value to a user-specific parameter.

Library	Range
ML_PLCopen.library	Parameters

Tab. 6-137: Library assignment

Interface description

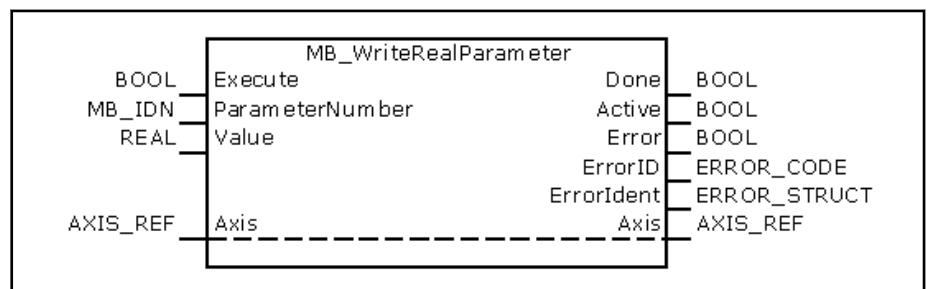


Fig. 6-152: FB MB_WriteRealParameter

ML_PLCOpen.library

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the desired control and the object controlled by this control.
VAR_INPUT	Execute	BOOL	Starts writing the value that occurs when there is a rising edge to the parameter.
	Parameter-Number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
	Value	REAL	Value to be written to the parameter
VAR_OUTPUT	Done	BOOL	The value has been written successfully to the control/the drive.
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error

Tab. 6-138: Interface of FB MB_WriteRealParameter

Signal-time diagram

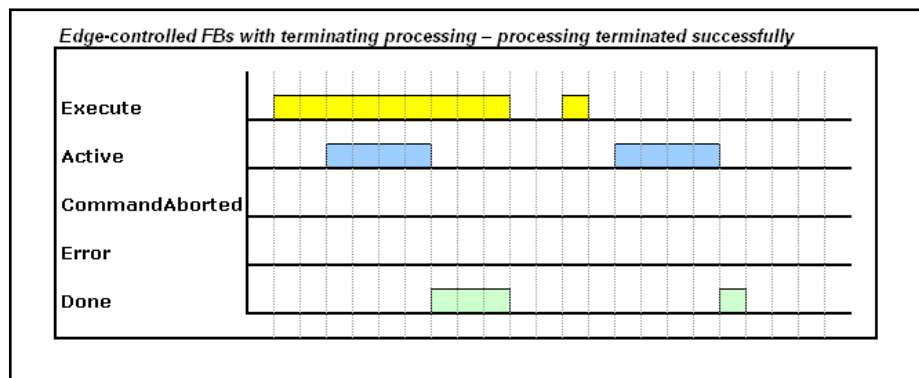


Fig. 6-153: Default signal time diagram

6.10.44 MB_WriteRealParameter, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks, page 277](#).



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object, page 275](#).

Error handling

The MB_WriteRealParameter FB generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010, page 340](#) - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030, page 343](#) - error when accessing the axis/the control.

6.10.45 MB_WriteStringParameter

Brief description The function block writes a chain of characters with a maximum of 255 characters in a user-specific parameter.

Library	Range
ML_PLCopen.library	Parameters

Tab. 6-139: Library assignment

Interface description

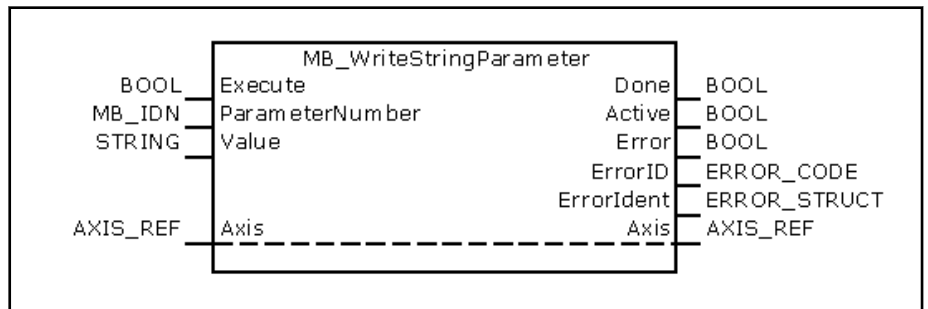


Fig. 6-154: FB MB_WriteStringParameter

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the desired control and the object controlled by this control.
VAR_INPUT	Execute	BOOL	Starts writing the value that occurs when there is a rising edge to the parameter.
	Parameter-Number	MB_IDN	Number of the parameter desired, (also refer to Access to Data of an Axis, Control, Kinematics (Axis Group) or another Object page 275)
	Value	STRING	Value to be written to the parameter
VAR_OUTPUT	Done	BOOL	The value has been written successfully to the control/the drive.
	Active	BOOL	Processing of data runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error

Tab. 6-140: Interface of FB MB_WriteStringParameter

Signal-time diagram

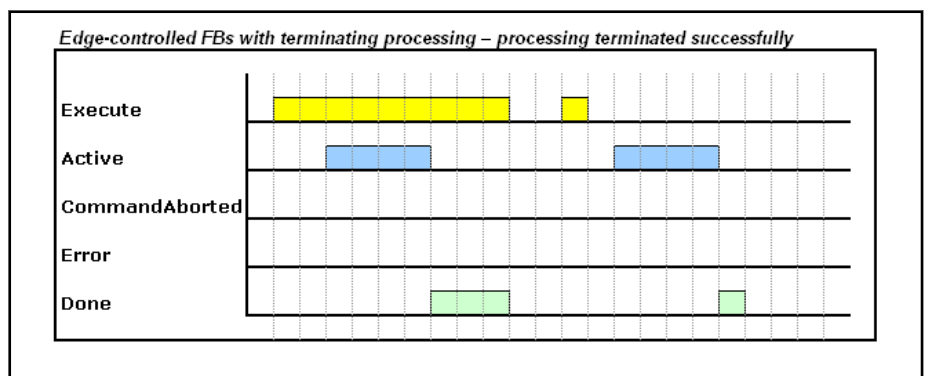


Fig. 6-155: Default signal time diagram

ML_PLCOpen.library

6.10.46 MB_WriteStringParameter, Special Features of SERCOS-based Axes



For the time required for the data transmission, see [Time Required for Data Transport with Read and Write Parameter Function Blocks](#), page 277.



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 [Access to Data of an Axis, Control, Kinematics \(Axis Group\) or another Object](#), page 275.

Error handling

The function block MB_WriteStringParameter generates the following types of error messages:

- [SERCOS_TABLE, ERROR_TABLE 16#0010](#) , page 340 - Error when accessing the S-parameter or P-parameter (e.g. parameter is not supported).
- [MLC_TABLE, ERROR_TABLE 16#0030](#) , page 343 - error when accessing the axis/the control.

6.10.47 MC_ReadActualPosition

Brief description

The function block reads the current position of the drive.



A modification of the axis at the entrance of the function block requires another 0/1 transition at "Enable".

Library	Range
ML_PLCOpen.library	Parameters

Tab. 6-141: Library assignment

Interface description

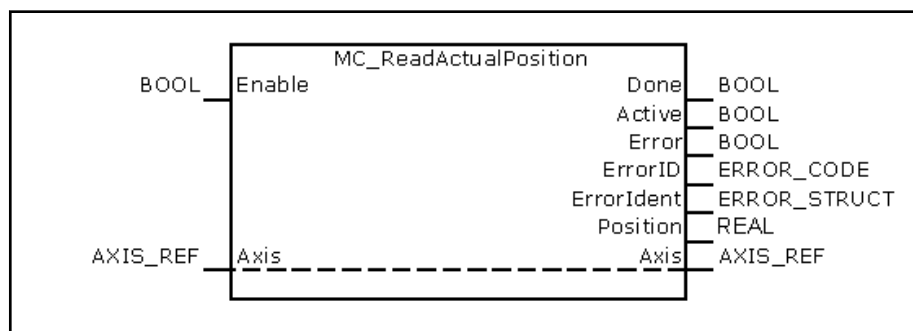


Fig. 6-156: FB MC_ReadActualPosition

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the actual 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 runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred in the FB instance.

	Name	Type	Comment
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error
	Position	REAL	Current absolute position of the drive ¹¹⁾

Tab. 6-142: Interface of FB MC_ReadActualPosition

Signal-time diagram

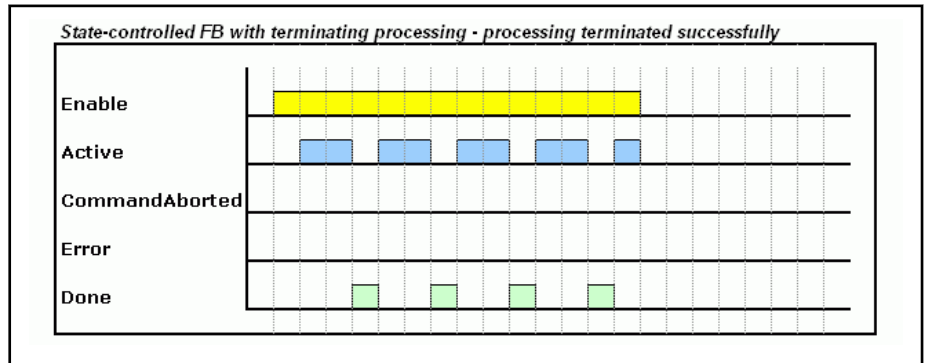


Fig. 6-157: Default signal time diagram

6.10.48 MC_ReadActualPosition, Special Features of SERCOS-Based Axes

Parameters used The function block reads the current position of the drive, (A-0-0100, position actual value).

Error handling The function block MC_ReadActualPosition generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

6.10.49 MC_ReadAxisError

Brief description The function block calls for 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 entrance of the function block requires another 0/1 transition at "Enable".

Library	Range
ML_PLCopen.library	Parameters

Tab. 6-143: Library assignment

Interface description

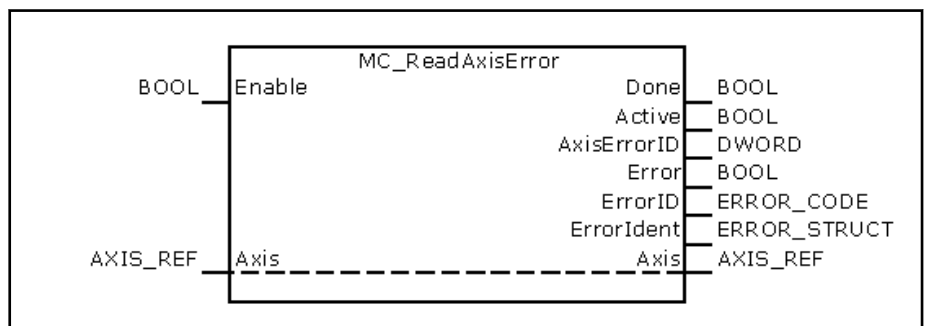


Fig. 6-158: FB MC_ReadAxisError

¹¹⁾ The specifications of the time, position, velocity, acceleration and delay depend on the effective weighting.

ML_PLCOpen.library

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the actual axis
VAR_INPUT	Enable	BOOL	Continuous reading of the position value as long as "Enable" is TRUE.
VAR_OUTPUT	Done	BOOL	Value available
	Active	BOOL	Processing of data runs after preprocessing is completed
	AxisErrorID	DWORD	Indicates the error of the drive of the axis
	Error	BOOL	Indicates that an error has occurred when the FB instance was called.
	ErrorID	ERROR_CODE	Short note to the error source of the block call, not of the axis error!
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error of the block call, not the axis error!

Tab. 6-144: Interface of FB MC_ReadActualPosition

Signal-time diagram

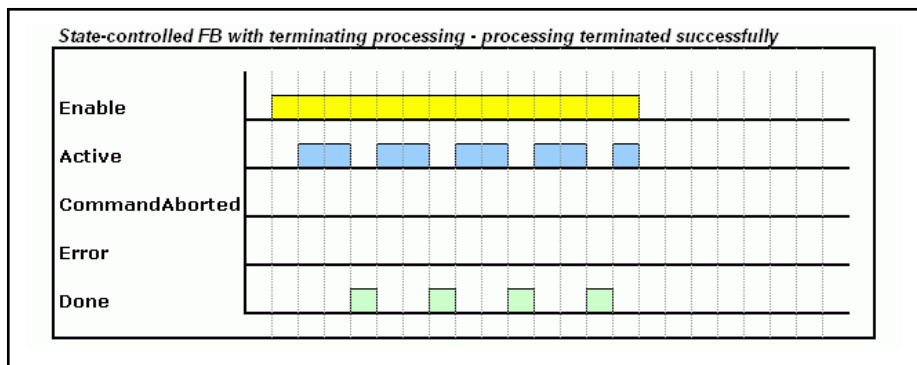


Fig. 6-159: Default signal time diagram

6.10.50 MC_ReadAxisError, Special Features of SERCOS-Based Axes

Parameters used This function block retrieves the error status (A-0-0020 Axis diagnostic message) of the axis addressed.

Error handling The function block MC_ReadAxisError generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

6.10.51 MC_ReadStatus

Brief description This function block reads the current status of the axis and emits this in a decoded way.



A modification of the axis at the entrance of the function block requires another 0/1 transition at "Enable".



Due to the extension of the block, the block was modified in a way which is incompatible with the previous version!

Library	Range
ML_PLCOpen.library	Parameters

Tab. 6-145: Library assignment

Interface description

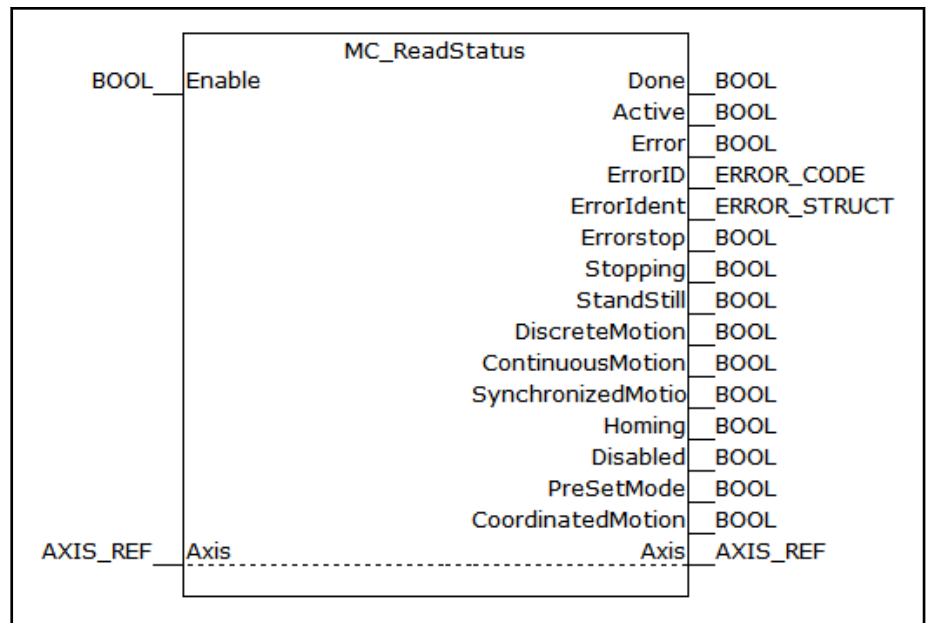


Fig. 6-160: FB MC_ReadStatus

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	Contains information regarding the actual 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 runs after preprocessing is completed
	Error	BOOL	Indicates that an error has occurred in the FB instance.
	ErrorID	ERROR_CODE	Brief indication of the cause for error
	ErrorIdent	ERROR_STRUCT	Detailed information regarding the error
	Errorstop	BOOL	Drive/axis in 'Errorstop' state
	Stopping	BOOL	Drive/axis in 'Stopping' state
	StandStill	BOOL	Drive/axis in 'Standstill' state
	Discrete Motion	BOOL	Drive/axis in 'DiscreteMotion' state
	Continuous motion	BOOL	Drive/axis in 'ContinuousMotion' state
	SynchronizedMotion	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

Tab. 6-146: Interface of FB MC_ReadStatus

ML_PLCopen.library

Signal-time diagram

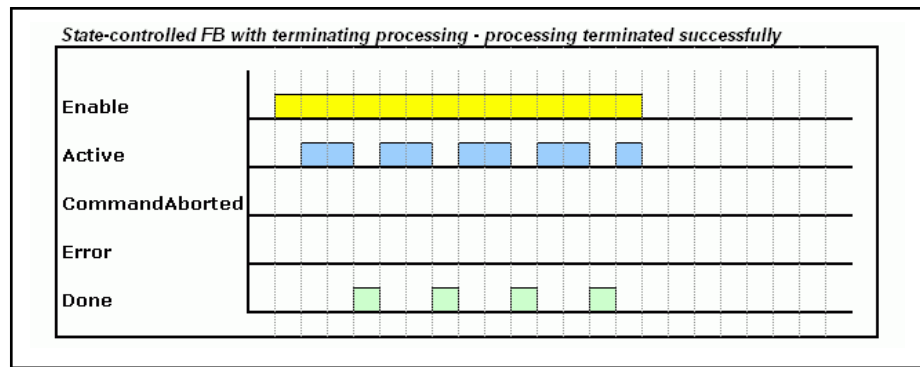


Fig. 6-161: Default signal time diagram

6.10.52 MC_ReadStatus, Special Features of sercos-Based Axes

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 it and outputs it. (Also see [State diagram of a real axis, page 172.](#))

Error handling The function block MC_ReadStatus generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the axis/control.

6.11 Function blocks to use the touch probe function

6.11.1 TouchProbe and Data Function Blocks for Using 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.

Global variable lists and data types

- [ProbeSelect, page 172](#), defines the filed which contains the data of the touch probes 1..100.
- [PROBECONFIG, page 172](#), 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 322](#),; used to initialize and configure the touch probe function.
- [ML_WriteExpectWindow, page 334](#) configures the expectation windows without a complete initialization run with the ML_InitTouchProbe being necessary.
- [ML_TouchProbe, page 329](#), used to activate the unique time position and time measurement and provides an automated time difference measurement.
- [ML_TouchProbeContinuous, page 331](#), used to activate the continuous time position and time measurement and provides an automated time difference measurement.
- [ML_AbortTrigger, page 321](#), completes an individual as well as a continuous measurement.

6.11.2 ML_AbortTrigger

Brief description The function block ML_AbortTrigger ends

- a single shot measurement is triggered by [ML_TouchProbe, page 329](#) or
- a continuous measurement is triggered by [ML_TouchProbeContinuous, page 331](#).

Library	Range
ML_PLCopen.library	TouchProbe

Tab. 6-147: Library assignment

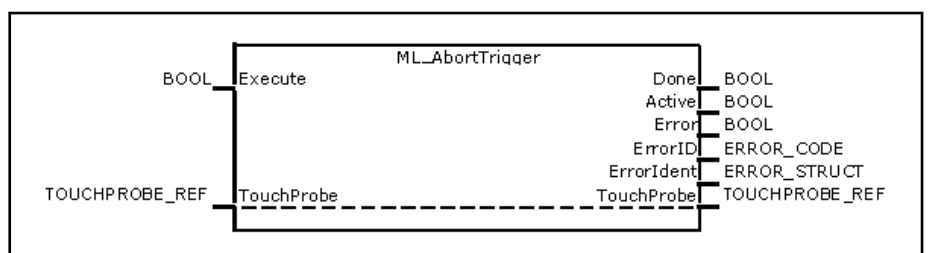
Interface description

Fig. 6-162: Function block ML_AbortTrigger

	Name	Type	Comment
VAR_IN_OUT	TouchProbe	TOUCHPROBE_REF	It contains information on the actual touch probe
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

ML_PLCopen.library

	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

Tab. 6-148: Interface of the ML_AbortTrigger FB

Signal-time diagram

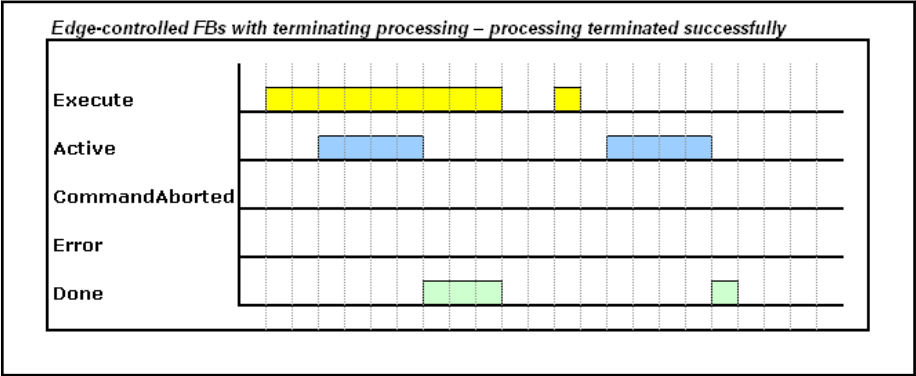


Fig. 6-163: Signal-time diagram: ML_AbortTrigger

Error handling: The ML_AbortTrigger function block generates the following error messages:

- [MLC_TABLE, page 343](#) - 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 329](#), or [ML_TouchProbeContinuous, page 331](#).

After its processing, the expectation window can be adapted online via [ML_WriteExpectWindow, page 334](#),



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 write 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).

Library	Range
ML_PLCopen.library	TouchProbe

Tab. 6-149: Library assignment

Interface description

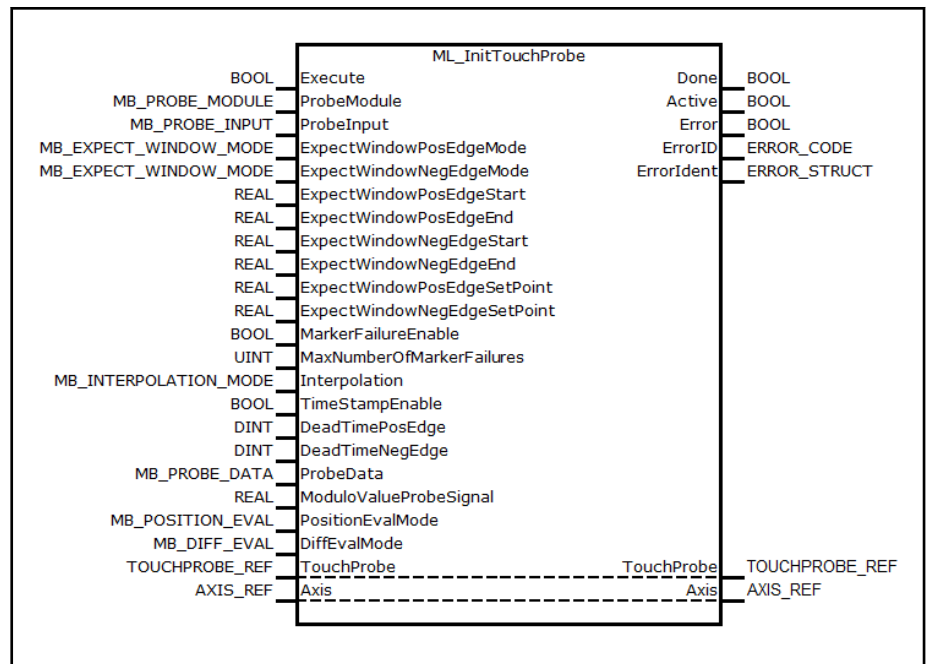


Fig. 6-164: ML_InitTouchProbe function block

	Name	Type	Comment ¹²⁾
VAR_IN_OUT	TouchProbe	TOUCHP-ROBE_REF	It contains information on the actual touch probe
	Axis	AXIS_REF	Contains information on the concrete axis
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, start 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.*

ML_PLCopen.library

	Name	Type	Comment ¹²⁾
	ExpectWindow-NegEdgeStart	REAL	Start position expectation window, negative edge M-0-0007, control word M-0-0204, start 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, M-0-0202, M-0-0203
	ExpectWindow-NegEdgeSet-Point	REAL	Set point of the expectation window, negative edge M-0-0007, M-0-0204, M-0-0205
	MarkerFailureEnable	BOOL	Marker monitoring: 0:= Not active; 1:= Active M-0-0007, control word
	MaxNumberOfMarkerFailure	UINT	Maximum number of marker failures 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, Signal selection
	ModuloValue-ProbeSignal	REAL	Modulo value of the signal to be measured (reference source) (see tab. 6-151 "Measuring signal selection (ProbeData)" on page 325)
	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

¹²⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

	Name	Type	Comment ¹²⁾
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

Tab. 6-150: 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 of the axis, encoder 1, 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 of the axis, encoder 1	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, Synchronous position command value	
TP_ELS_CMD_POS	A-0-2655, ELS, position command value	
TP_RES_MASTER_AXIS_POS	A-0-2650, 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, Effective master axis position	
TP_CAM_SHAFT_PROF_ACCESS_ANGLE	A-0-2656, ELS, cam, access phase	

¹²⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

ML_PLCOpen.library

MB_PROBE_DATA - Element	Parameters	Axis reference
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. 6-151: Measuring signal selection (ProbeData)

- ModuloValueProbeSignal: The modulo value of the measuring signal is used to compute a relative expectation window (see [tab. 6-152 "Expectation window, calculation rule" on page 327](#))
- "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. 6-151 "Measuring signal selection \(ProbeData\)" on page 325](#))

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,

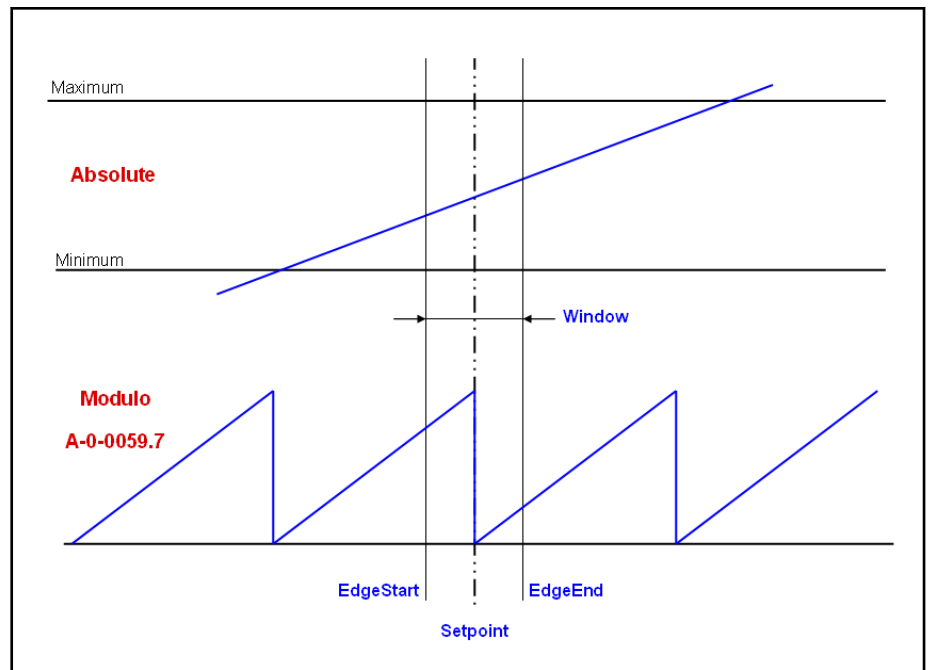


Fig. 6-165: Expectation window of an edge

	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 + Modulo value, modulo value)
		End:=ExpectWindowPosEdgeEnd	End:= MOD(ExpectWindowPosEdgeSetPoint + ExpectWindowPosEdgeEnd, modulo value)
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 + modulo value, modulo value)
		End:=ExpectWindowNegEdgeEnd	End:= MOD(ExpectWindowNegEdgeSetPoint + ExpectWindowNegEdgeEnd, modulo value)

Tab. 6-152: 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.

ML_PLCopen.library

- 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:

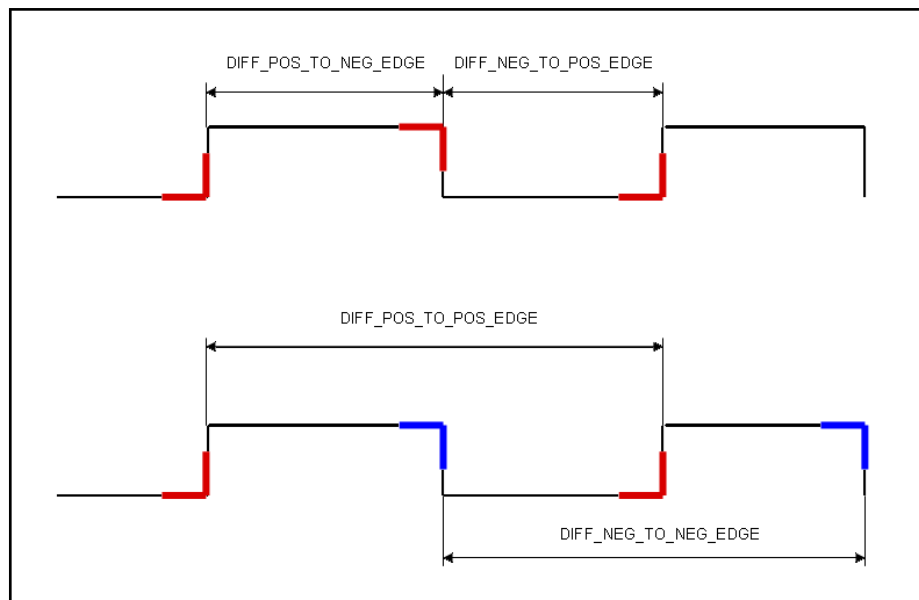


Fig. 6-166: Difference measurement methods

Signal-time diagram

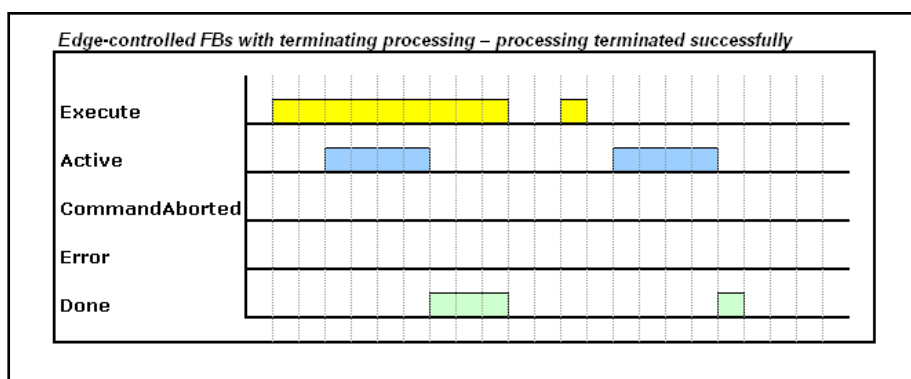


Fig. 6-167: Signal-time diagram: ML_InitTouchProbe

Troubleshooting: The ML_InitTouchProbe function block generates the following messages:

- [MLC_TABLE, page 343](#) - 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 322](#), .

After the initialization, the expectation window can be modified using the function block [ML_WriteExpectWindow, page 334](#),.



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 write 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).

Library	Range
ML_PLCopen.library	TouchProbe

Tab. 6-153: Library assignment

ML_PLCopen.library

Interface description

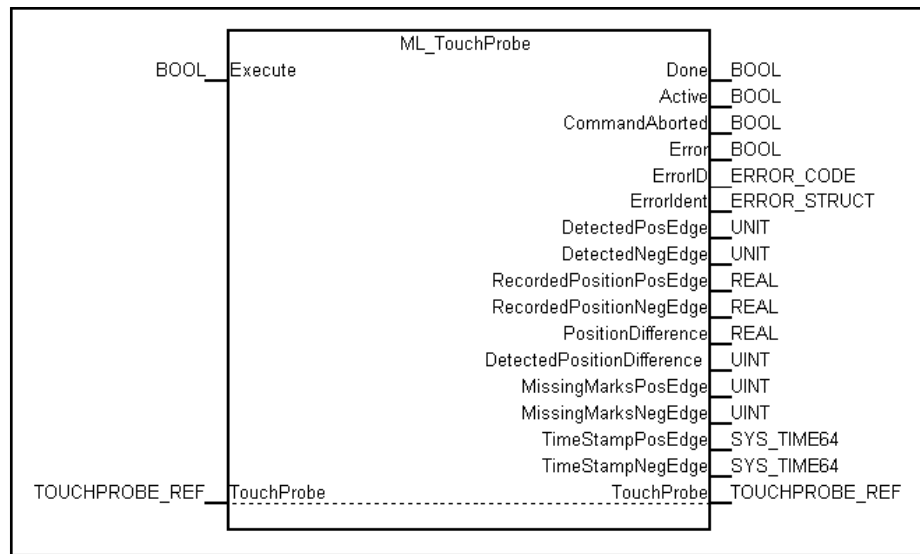


Fig. 6-168: ML_TouchProbe function block

	Name	Type	Comment ¹³⁾
VAR_IN_OUT	TouchProbe	TOUCHPROBE_REF	It contains information on the actual touch probe
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	Measurement value counter for positive edge M-0-0409, Counter measured values, positive edge
	DetectedNegEdge	UINT	Measurement value counter for negative edge M-0-0410, Counter 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 difference values

¹³⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

	Name	Type	Comment ¹³⁾
	MissingMark-sPosEdge	UINT	Number of marker failures, positive edge M-0-0224, Number of marker failures, positive edge
	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, Timestamp, negative edge

Tab. 6-154: Interface of the ML_TouchProbe FB

Signal-time diagram

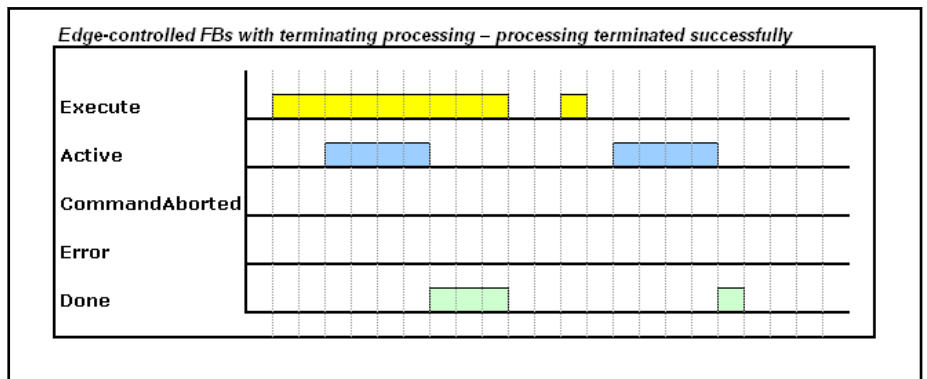


Fig. 6-169: Signal-time diagram: ML_TouchProbe

Functional description

Measured value acquisition mode

- **Single shot measurement** - After enabling the respective touch probe (M-0-0405), 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 331,, a new measured value is recorded with every measurement value trigger.

Troubleshooting: The function block ML_AbortTrigger generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the touch probe/control.

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 322](#).

After the initialization, the expectation window can be modified using the function block [ML_WriteExpectWindow, page 334](#).

¹³⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

ML_PLCOpen.library



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).

Library	Range
ML_PLCOpen.library	TouchProbe

Tab. 6-155: Library assignment

Interface description

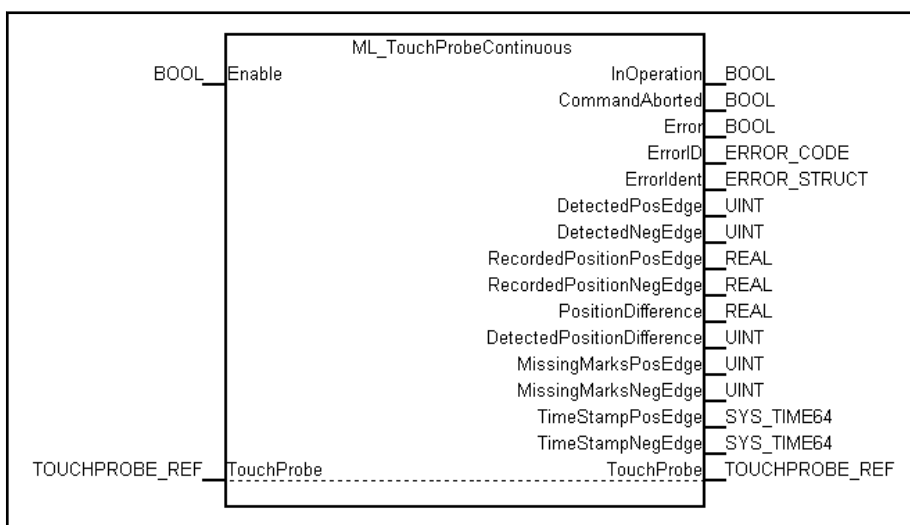


Fig. 6-170: ML_TouchProbeContinuous FB

	Name	Type	Comment ¹⁴⁾
VAR_IN_OUT	TouchProbe	TOUCHPROBE_REF	It contains information on the actual touch probe
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
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information
	DetectedPosEdge	UINT	Measurement value counter for positive edge M-0-0409, Counter measured values, positive edge
	DetectedNegEdge	UINT	Measurement value counter for negative edge M-0-0410, Counter measured values, negative edge

¹⁴⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

	Name	Type	Comment ¹⁴⁾
	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 difference values
	MissingMarksPosEdge	UINT	Number of marker failures, positive edge M-0-0224, Number of marker failures, positive edge
	MissingMarksNegEdge	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, Timestamp, negative edge

Tab. 6-156: Interface of the ML_TouchProbeContinuous FB

Signal-time diagram

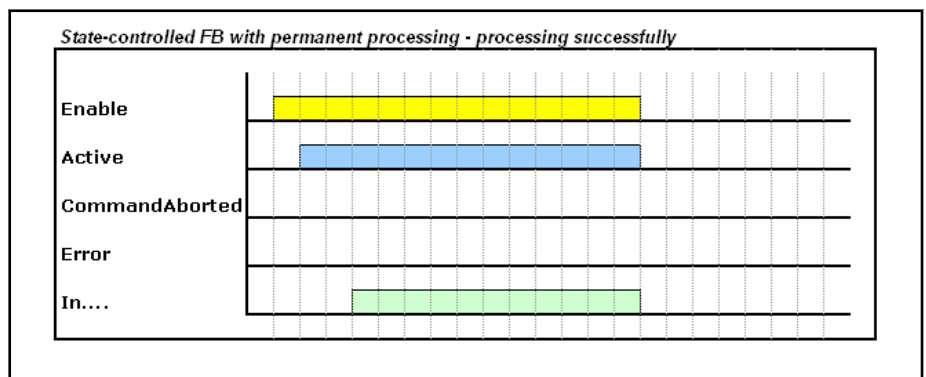


Fig. 6-171: Signal-time diagram: ML_TouchProbeContinuous

Functional description

Measured value acquisition mode

- **Single shot measurement** - After enabling the respective touch probe (M-0-0405), [ML_TouchProbe](#), page 329., 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 function block ML_AbortTrigger generates the following error messages:

- [MLC_TABLE](#), page 343 - error when accessing the touch probe/control.

¹⁴⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

ML_PLCOpen.library

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 322, being necessary.



If the "Enable" of the FB is activated, modifications of the inputs are accepted **immediately**.

Library	Range
ML_PLCOpen.library	TouchProbe

Tab. 6-157: Library assignment

Interface description

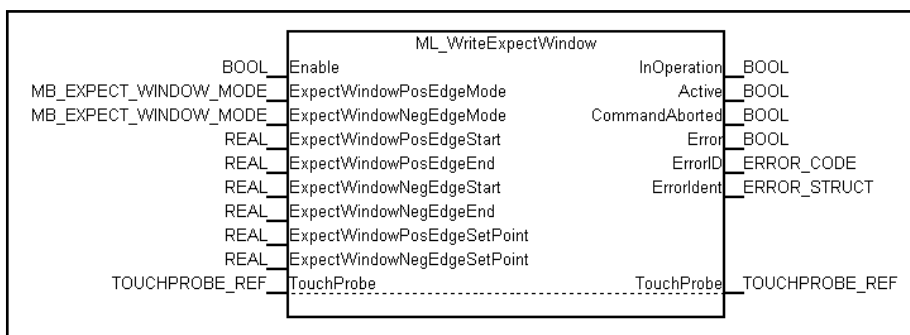


Fig. 6-172: ML_WriteExpectWindow FB

	Name	Type	Comment ¹⁵⁾
VAR_IN_OUT	TouchProbe	TOUCHPROBE_REF	It contains information on the actual touch probe
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, start 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, start position of the expectation window, negative edge

¹⁵⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

	Name	Type	Comment ¹⁵⁾
	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, M-0-0202, M-0-0203
	ExpectWindow-NegEdgeSet-Point	REAL	Set point of the negative expectation window M-0-0007, M-0-0204, M-0-0205
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

Tab. 6-158: Interface of FB ML_WriteExpectWindow

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,

¹⁵⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

ML_PLCopen.library

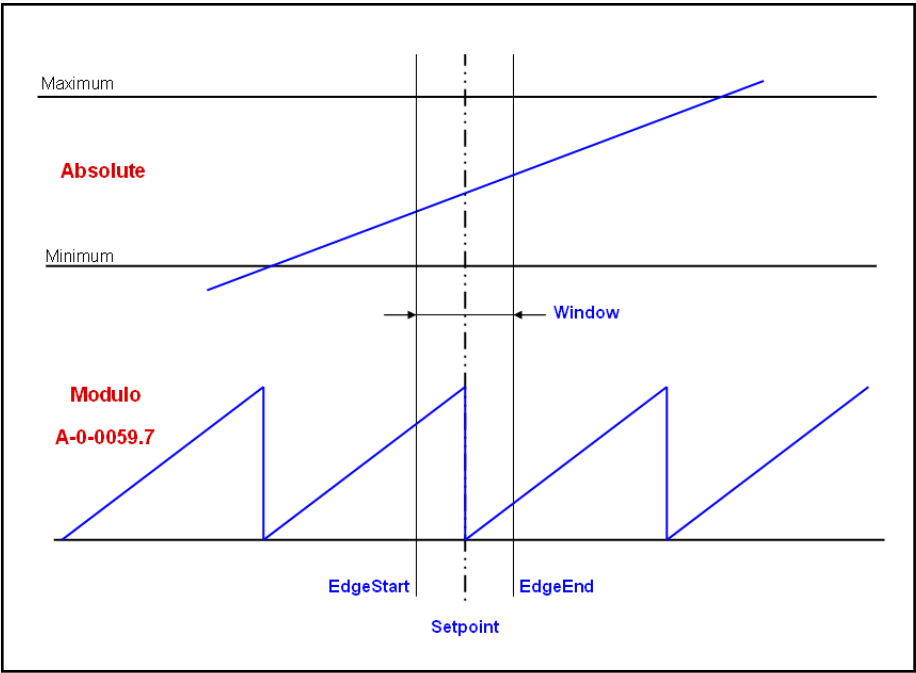


Fig. 6-173: Expectation window of an edge

	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. 6-159: Expectation window, calculation rule

Signal-time diagram

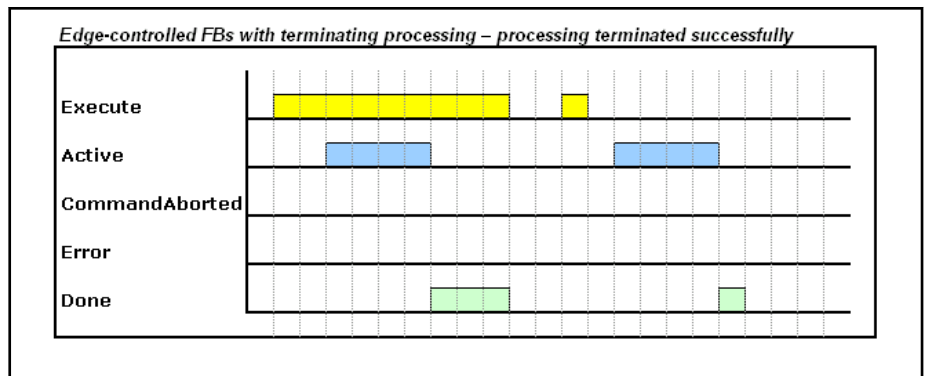


Fig. 6-174: Signal-time diagram: ML_WriteExpectWindow

Error handling: The ML_AbortTrigger function block generates the following error messages:

- [MLC_TABLE, page 343](#) - error when accessing the touch probe/control.

6.12 Time Required to Process Motion Function Blocks in the IndraMotion MLC

A servos 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 FB and leaving it averages 4.6 µs. The FB 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.

Action / T _{avr} [µs]	13VRS, L25	13VRS, L45	13VRS, L65
MC_Power, real axis	25,9	11,2	4,8
MC_MoveAbsolute, real axis	25,6	11,5	4,6
MC_MoveAbsolute, virtual axis	12,7	5,7	2,4
MC_MoveRelative, real axis	25,2	11,6	4,8

ML_PLCOpen.library

Action / T _{avr} [μs]	13VRS, L25	13VRS, L45	13VRS, L65
MC_MoveRelative, virtual axis	13,0	5,4	2,4
MC_MoveAdditive, real axis	26,8	11,8	5,4
MC_MoveAdditive, virtual axis	9,7	4,8	2
MC_MoveVelocity, real axis	26,8	11,3	4,8
MC_MoveVelocity, virtual axis	12,3	5,6	2,3
MC_Stop, real axis	29,1	11,8	5,6
MC_Stop, virtual axis	12,1	5,8	2,6
MC_GearIn	29,2	12	5
MC_GearOut	27,6	10,4	4,8
Data access, direct variable	18,3	5,7	2,8
Data access, variable in the shared memory	3,7	1,2	0,6
Onboard I/Os		2000,8	2000,5
Inline I/Os	8000,1	7429,4	4000,1
PROFIBUS I/Os	10501,9	19888	10463,5

Tab. 6-160: 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 FB 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.

6.13 Axis types and function blocks supported by these axis 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).



13VRS

- If a real axis, IndraDrive, MPx06 / MPx07 / MPx08 / MPx16 / MPx17 / MPx18, 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, is calculated on the control using the Pack-Profile; refer to the column **SercosDrive**

Function block	Real axes					Virtual axes	Encode/ link* axis	Controller axis*
	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(3)
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

ML_PLCopen.library

Function block	Real axes					Virtual axes	Encode/link* axis	Controller axis*
	IndraDrive		HydraulicDrive*		sercos-Drive			
	Interpolation Drive	Interpolation Control	Interpolation Drive	Interpolation Control				
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	-	-	-	-
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	-	-	-	-
ML_WriteCyclicPosition	X	X	X	X	X	X	-	X
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	-	-	-	-
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	-	-	-	-
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 IndraMotion MLC

(1)

Not MPx03

(2)

Also with OpenLoop

(3)

Depends on the implementation of the controller

(4)

For 13VRS and higher

(5)

For 13VRS and > MPx06

Tab. 6-161: Function blocks in IndraLogic XLC and IndraMotion MLC

6.14 Error Tables (Error_Tables)

6.14.1 SERCOS_TABLE, ERROR_TABLE 16#0010

SERCOS_TABLE - ERROR_TABLE, Table number 16#0010

IndraLogic XLC /IndraMotion MLC extend the four-digit sercos error codes as eight-digit MLC_Error codes as follows:

Supplements are "F" for error, "0" as severity (severity of the error) and "2D" as a sign for the origin. Then, the four sercos digits follow.

Severity 0 signifies no reaction 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
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

ML_PLCopen.library

SERCOS_ TABLE ErrorIdent / Additional1	Description	MLC_ TABLE ErrorIdent / Additional1
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
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 (operation 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

SERCOS_ TABLE ErrorIdent / Additional1	Description	MLC_ TABLE ErrorIdent / Additional1
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
8096	sercos: Device deactivated	F02D8096
8097	sercos: Device does not exist	F02D8097

Tab. 6-162: sercos errors and their use in IndraLogic XLC / IndraMotion MLC

6.14.2 MLC_TABLE, ERROR_TABLE 16#0030

MLC_TABLE - ERROR_TABLE, Table number 16#0030

The entries in the following table match the MLC troubleshooting guide (diagnostics).

ErrorIdent / Additional1	Diagnostics
F0220200	F2229200 General drive error
F0220201	Invalid control number
F0220202	Invalid axis number
F0220203	F2229203 Invalid command type
F0220204	F2229204 Command not supported by axis
F0220205	F2229205 Command rejected, no power
F0220206	F2229206 Command rejected while 'Stopping'
F0220207	F2229207 Command rejected while 'Homing'
F0220208	F2229208 Invalid command number
F0220210	F2229210 Reset still 'ErrorStop'
F0220211	F2229211 Command rejected while 'Diskrete Motion'
F0220212	F2229212 Command rejected while 'Continuous Motion'
F0220213	F2229213 Command rejected while 'Synchronized Motion'
F0220214	F2229214 Command rejected while 'ErrorStop'
F0220216	F2229216 No power available
F0220217	F2229217 Command rejected while 'StandStill' and 'PowerOn'
F0220218	F2229218 Command rejected while 'StandStill'
F0220223	F2229223 Velocity below minimum
F0220224	F2229224 Velocity above maximum
F0220225	F2229225 Acceleration below minimum
F0220226	F2229226 Acceleration above maximum

ML_PLCopen.library

ErrorIdent / Additional1		Diagnostics
F0220227	F2229227	Deceleration below minimum
F0220228	F2229228	Deceleration above maximum
F0220229	F2229229	Jerk below minimal
F0220230	F2229230	Jerk above maximum
F2220231		Invalid master axis data
	F2229231	Invalid direction
F0220232	F2229232	Invalid master control number
F0220233	F2229233	Invalid master axis number
F0220234	F2229234	Invalid CamTable ID
	F2229235	Error in MotionProfile
F0220263	F2229263	RatioNumerator below minimum
F0220264	F2229264	RatioNumerator above maximum
F0220265	F2229265	RatioDenominator below minimum
F0220266	F2229266	RatioDenominator above maximum
F0220267	F2229267	MasterFineAdjust below minimum
F0220268	F2229268	MasterFineAdjust above maximum
F0220269	F2229269	Invalid SyncMode
F0220270	F2229270	Invalid StartMode
F0220271	F2229271	Cam distance below minimum value
F0220272	F2229272	Cam distance above maximum value
F0220273	F2229273	Command rejected while 'Parameter Mode'
F0220274	F2229274	Axis not referenced
F0220275	F2229275	Command aborted
F0220276	F2229276	Control error
F0220277	F2229277	Direct master axis change
F0220278	F2229278	Command 'Disabled axis' active
F0220279	F2229279	Command 'parking axis' active
F0220280	F2229280	Command rejected while 'Coordinated Motion'
F0220281	F2229281	Invalid float data
F0220282	F2229282	MotionProfile, SetSelection below minimum value
F0220283	F2229283	MotionProfile, SetSelection above maximum value
	F2229284	Cam table is write-protected
	F2229285	CAM table, usage counter out of range
	F2229286	Choice relative positioning above max. value
F0220287		Unreferenced cam table object
	F2229287	Can not assign axis to kinematics

ErrorIdent / Additional1		Diagnostics
F0220288		ChangeCamTable: above maximum value
F0220289		ChangeCamTable: below minimum value
F0220290		ChangeCamTable: invalid ID
F0220291		Unreferenced MotionProfile object
F0220292		ChangeMotionProfileSet: SetNumber below minimum value
F0220293		ChangeMotionProfileSet: SetNumber above maximum value
F0220294		ChangeMotionProfileSet: Selected profile write-protected
F0220295		ChangeMotionProfileStep, invalid StepNumber
	F2229296	Absolute entry point in FlexProfile not found
	F2229297	Absolute entry in FlexProfile not possible
	F2229298	Master velocity of FlexProfile is negative

Tab. 6-163: Diagnostics of the Motion function blocks (axis state) I



Depending on the pre-selected axis reaction, the error **F*22*xxx** appears as

- F0220xxx, if no error reaction by the axis
- F2229xxx, if "axis stop", best shutdown possible.

ErrorIdent / Additional1		Diagnostics
F0220300	F2229300	Invalid command channel
F0220301	F2229301	Command rejected while 'Controlled Motion'
	F2229310	Error in calculation of FlexProfile step
	F2229311	FlexProfile: Invalid switching position
	F2229312	FlexProfile: Invalid FlexProfile startmode
	F2229313	FlexProfile: Invalid FlexProfile slave axis offset
	F2229314	Sum of dist. not 0 or mod. value, cyclic exec. not pos.
	F2229315	Invalid FlexProfile: Execution not possible
F0220316		ChangeFlexProfile: Length of FlexProfile lists inconsist.
F0220317		ChangeFlexProfile: No steps in profile
F0220318		ChangeFlexProfile: Unknown master
F0220319		ChangeFlexProfile: Motion law unknown
F0220320		ChangeFlexProfile: Step mode unknown
F0220321		ChangeFlexProfile: Step range 0 or negative
F0220322		ChangeFlexProfile: Invalid comb. of boundary value types
F0220323		ChangeFlexProfile: Invalid boundary values
F0220324		ChangeFlexProfile: Invalid Hub
F0220325		ChangeFlexProfile: Invalid value for turnaround point disp
F0220326		ChangeFlexProfile: Invalid slope

ML_PLCopen.library

ErrorIdent / Additional1		Diagnostics
F0220327		ChangeFlexProfile: Invalid Cam table
F0220328		ChangeFlexProfile: Motion law + selected step mode
F0220329		ChangeFlexProfile: Velocity jump in profile
F0220330		ChangeFlexProfile: Acceleration jump in profile
F0220331		ChangeFlexProfile: Jerk jump in profile
F0220332		ChangeFlexProfile: Velocity limit invalid
F0220333		ChangeFlexProfile: Acceleration limit invalid
F0220334		Unreferenced FlexProfile object
F0220335	F2229335	FlexProfile: Set is write-protected
	F2229336	FlexProfile: Invalid step number
	F2229337	FlexProfile: Set number out of range
	F2229338	ChangeFlexProfile: Invalid nominal master velocity
	F2229339	ChangeFlexProfile: invalid operation mode
F0220341	F2229341	FlexEvent: Set is write-protected
	F2229342	FlexEvent: Set number out of range
F0220343		ChangeFlexEvent: Length of FlexProfile lists inconsistent
F0220344		ChangeFlexEvent: Step Number assignment is invalid
F0220345		ChangeFlexEvent: Trigger-Mode is invalid
F0220346		ChangeFlexEvent: Action-Mode is invalid
	F2229347	FlexProfile: Invalid. Single execution not possible
	F2229348	FlexProfile: Invalid. Cyclic execution not possible
F0220349		ChangeFlexProfile: Jerk limit invalid
F0220350		ChangeFlexProfile: Limit too low
	F2229351	Synchronisation velocity below minimum
	F2229352	Synchronisation velocity above maximum
	F2229353	Synchronisation acceleration below minimum
	F2229354	Synchronisation acceleration above maximum
F0220355		ChangeFlexProfile: Invalid sequence of Flex-Steps
F2220356		Invalid filter time const., additive master position
F2220357		Invalid filter time const., angle offset begin of profile
F2220358		Invalid filter time const., additive slave position
	F2229359	FlexProfile: Invalid master axis offset
F0220360	F2229360	Command rejected while 'Controlled Motion'
F0220361		ChangeFlexProfile: Invalid master axis source
	F2229361	Command not supported by axis (Configuration)

ErrorIdent / Additional1		Diagnostics
	F2229362	Command not supported by axis (Functional Package)
	F2229363	MotionProfile invalid

Tab. 6-164: Diagnostics of the Motion function blocks (axis state) II



Depending on the pre-selected axis reaction, the error **F*22*xxx** appears as

- F0220xxx, if no error reaction by the axis
- F2229xxx, if "axis stop", best shutdown possible.

ErrorIdent / Additional1		Diagnostics
F0220501		Wrong data type
F0220502		ProbeModule out of range
F0220503		ProbeInput out of range
F0220504		ExpectWindowPosEdgeMode out of range
F0220505		ExpectWindowNegEdgeMode out of range
F0220506		ExpectWindowPosEdgeStart out of range
F0220507		ExpectWindowPosEdgeEnd out of range
F0220508		ExpectWindowNegEdgeStart out of range
F0220509		ExpectWindowNegEdgeEnd out of range
F0220510		ExpectWindowPosEdgeSetPoint out of range
F0220511		ExpectWindowNegEdgeSetPoint out of range
F0220512		MarkerFailureEnable out of range
F0220513		MaxNumberOfMarkerFailure out of range
F0220514		Interpolation out of range
F0220515		TimeStampEnable out of range
F0220516		DeadTimePosEdge out of range
F0220517		DeadTimeNegEdge out of range
F0220518		ProbeData out of range
F0220519		PositionEvalMode out of range
F0220520		DiffEvalMode out of range
F0220521		TOUCHPROBE_REF is not valid or wrong configuration
F0220522		AXIS_REF is not valid or wrong configuration of axis

Tab. 6-165: Diagnostics of the Motion function blocks (axis state) III



Depending on the pre-selected axis reaction, the error **F*22*xxx** appears as

- F0220xxx, if no error reaction by the axis
- F2229xxx, if "axis stop", best shutdown possible.

ML_PLCopen.library

ErrorIdent / Additional1	Diagnostics
F0220600	Path / file name unknown
F0220601	Unsupported file system
F0220602	Input 'StartIdx' has an invalid value
F0220603	Invalid state of state machine
F0220604	Error in file structure
F0220605	Input 'ArrayDimension' has an invalid value
F0220606	Point not found
F0220607	Wrong number of coordinates per point
F0220608	Wrong number of kinematics
F0220609	Wrong point type, point in world or joint coordinates
F2220610	Invalid master axis

Tab. 6-166: Diagnostics of the Motion function blocks (axis state) IV



Depending on the pre-selected axis reaction, the error **F*22*xxx** appears as

- F0220xxx, if no error reaction by the axis
- F2229xxx, if "axis stop", best shutdown possible.

ErrorIdent / Additional1		Diagnostics
F00D1002		Cyclic position channel already open
F00D1003		Cyclic force channel already open
F00D1004		Cyclic velocity channel already open
F00D1005		Cyclic analog channel already open
F00D100A		Error while reading C-0-0483
F00D100B		Error while reading C-0-0484
F00D100C		Error while opening cyclic channel
F00D100D		Error while opening cyclic channel
F00D1014		No reference for opening cyclic channel available
	F20D1015	Error while closing cyclic channel
F00D1016		Wrong #-address of axis data
F00D1017		Wrong #-address of axis data
F00D1018		Wrong #-address of axis data
F00D1019		Error while closing cyclic channel
	F20D1025	Error while closing cyclic channel
F00D1027		Error while closing cyclic channel
	F20D1029	Error opening cyclic channel
F00D102A		ListParameter requested, ReadParameter
	F20D102B	Error opening cyclic channel (READ-ACCESS)

ErrorIdent / Additional1		Diagnostics
	F20D102C	Error reading access (DirectReadReal)
	F20D102D	Error reading access (DirectReadWord)
	F20D102E	Error reading access (DirectReadDword)
F00D1030		Error while closing cyclic channel
	F20D1034	Error write access (DirectWriteWord)
	F20D1036	Axis has not been found (ML_AXISDATA)
	F20D1037	Invalid write/read access to ML_AXISDATA
F00D103B		Invalid control number
F00D103C		Invalid axis number
F00D1205		Error, cannot handle a single parameter
F00D1207		Error, data greater 64KBytes
F00D1210		Error, cannot handle a single parameter
F00D1211		Error ReadSERCOSDataStatus, no memory available
F00D1218		Error, no memory available
F00D1219		Error, cannot handle a single parameter
F00D1220		Unsupported selection of sercos element
F00D1221		Timeout on parameter access
F00D1222		Too many data to transmit
F00D1224		Error, no memory available
F00D1225		Error, cannot handle a single parameter
F00D1227		Error while writing date
F00D1230		Error ReadSERCOSAttribute, no memory available

Tab. 6-167: Diagnostics MB_ReadxxxParameter / MB_WritexxxParameter / direct access

ML_PLCopen.library

ErrorIdent / Additional1	MB_ReadxxxParameter / MB_WritexxxParameter / direct access
F90D1021	Error ReadParameter, no memory available
F90D1023	Error WriteParameter
F90D1024	Error WriteParameter, no memory available
F90D1032	No access to axes (1)
F90D1033	No access to axes (2)
F90D1038	Error getting MessageQueue
F90D1039	Thread could not be started
F90D103A	Left endless While-Loop
F90D1204	Error, no memory available
F90D1209	Error, no memory available

Tab. 6-168: Fatal system errors of MB_ReadxxxParameter / MB_WritexxxParameter / direct access



Depending on the pre-selected axis reaction, the error **F*0Dxxxx** appears as

- F00Dxxxx, if no error reaction by the axis
- F20Dxxxx, if "axis stop", best shutdown possible
- F90Dxxxx, no error reaction to the drive; torque disable

ErrorIdent / Additional1	Diagnostics
F0290001	Transformation not valid
F0290400	F2299400 General kinematics error
F0290401	F2299401 Invalid control number
F0290402	F2299402 Invalid kinematics number
F0290403	F2299403 Invalid command type
F0290404	F2299404 Command not supported by kinematics
F0290405	F2299405 Command rejected while 'StandStill Incomplete'
F0290406	F2299406 Command rejected while 'PM Program Stopping'
F0290407	F2299407 Command rejected while 'StandStill Complete'
F0290408	F2299408 Invalid command number
F0290409	F2299409 Command not supported in this state
F0290410	F2299410 Reset still 'ErrorStop'
F0290411	F2299411 Command rejected while 'PM Program Running'
F0290412	F2299412 Error while selecting RCL program
F0290413	F2299413 Command rejected while 'Parameter Mode'
F0290414	F2299414 Command rejected while 'ErrorStop'
F0290415	F2299415 Command aborted

ErrorIdent / Additional1		Diagnostics
F0290416	F2299416	Switching to 'StandStill' not possible
F0290417	F2299417	RCL process does not exist
F0290501	F2299501	Two PLC-MotionCommands per motion cycle not allowed
F0290502	F2299502	Command rejected - buffer full
F0290503	F2299503	Invalid motion command
F0290505	F2299505	Synchronisation to more than one belt not possible

Tab. 6-169: Kinematic state diagnostics of the MLC_Table I



Depending on the pre-selected axis reaction, the error **F*29*xxx** appears as

- F0290xxx, if no error reaction by the axis
- F2299xxx, if "axis stop", best shutdown possible
- F9299xxx, no error reaction to the drive; torque disable

ErrorIdent / Additional1		Diagnostics
F0290501	F2299501	Two PLC-MotionCommands per motion cycle not allowed
F0290502	F2299502	Command rejected - buffer full
F0290503	F2299503	Invalid motion command
F0290505	F2299505	Synchronisation to more than one belt not possible
	F2299506	Slope type is not supported
	F2299507	Invalid set mode
	F2299508	Invalid coordinate system
	F2299509	Belt config not allowed temporary
	F2299510	Belt not configured
	F2299511	Belt synchronization error
	F2299512	Invalid belt synchronisation mode
	F2299513	Bad belt error reaction mode
	F2299514	Belt begin length invalid
	F2299515	Belt begin length greater total length
	F2299516	Wrong PCS index
	F2299517	Wrong CS selected
	F2299518	The TCP is not within belt working area
	F2299528	TeachIn: No name given
	F2299529	TeachIn: Name too long
	F229952A	TeachIn: Wrong name
	F229952B	TeachIn: Point already exist
	F229952C	TeachIn: Point doesn't exist
	F229952D	TeachIn: Wrong option

ML_PLCopen.library

ErrorIdent / Additional1	Diagnostics
F229952E	TachIn: File corrupted
F229952F	TeachIn: Wrong coordinate
F2299530	TeachIn: Wrong coordinate system
F2299531	TeachIn: File could not be opened
F2299532	Values for relative desynchronisation invalid
F2299533	Command buffer full
F2299534	Transformation did not supported freedom degree

Tab. 6-170: Kinematic state diagnostics of the MLC_Table II



Depending on the pre-selected axis reaction, the error **F*29*xxx** appears as

- F0290xxx, if no error reaction by the axis
- F2299xxx, if "axis stop", best shutdown possible
- F9299xxx, no error reaction to the drive; torque disable

ErrorIdent / Additional1	Diagnostics
F9299519	JCS not set
F929951A	BCS not set
F929951B	WCS not set
F929951C	PCS not set
F929951D	Trafo manager not set
F929951E	Preshape execute error
F929951F	Shape execute error
F9299520	Postshape execute error
F9299521	Move not allowed for this cmd
F9299522	No robot set
F9299523	Belt no time set
F9299524	Desync init failed
F9299525	lpo cycle time zero
F9299526	lpo distance below zero
F9299527	lpo too much elements

Tab. 6-171: Kinematic state diagnostics of the MLC_Table III, Fatal System Errors



Depending on the pre-selected axis reaction, the error **F*29*xxx** appears as

- F0290xxx, if no error reaction by the axis
- F2299xxx, if "axis stop", best shutdown possible
- F9299xxx, no error reaction to the drive; torque disable

7 RMB_PLCopenFieldBus.library

7.1 RMB_PLCopenFieldBus.library, General

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 constants, page 353,](#)
- [Data types, page 354,](#)
- [Motion_FBs, page 357,](#)
- [Parameter_FBs, page 377,](#) and
- [System_FBs, page 413.](#)

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 353,](#)
- [GVL_P_Param_ID, page 354](#)

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.

RMB_PLCopenFieldBus.library

Library	Range
RMB_PLCopenFieldbus	Global Variable Lists/parameter ID

Tab. 7-1: 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 355](#)

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.

Library	Range
RMB_PLCopenFieldbus	Global Variable Lists/parameter ID

Tab. 7-2: 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 355](#)

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 355](#), and
- [AXIS_REF \(field bus drives\), page 355](#), 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 The data type allows to define a system-specific parameter length. These are 64 bits for the parameters of the IndraLogic XLC or IndraMotion MLC. S- and P-parameters are automatically filled with up to 64 bits.

Library	Range
RMB_PLCopenFieldbus	Data Types

Tab. 7-3: Library assignment

Program:

```
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. 7-4: 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 which is defined in the "RMB_PLCopenFieldBus.library".



Only for field bus drives/axes connected to the PROFIBUS DP.

Library	Area
RMB_PLCopenFieldBus	Data Types

Tab. 7-5: Library assignment

Program:

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

RMB_PLCopenFieldBus.library

```

(* It is used as a VAR_IN_OUT in all 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;
  IndraDriveParameter: MP_DriveParameter;
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.

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 in the `RIL_CommonTypes.lib`, folder "Datatypes 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 "Datatypes 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 reasonable to compare the PROFIBUS 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.

They can only be used for IndraDrive C/M from MPx2VRS, IndraDrive Cs from MPx16VRS coupled via PROFIBUS.

State diagram

The following diagram normatively defines the behavior of the MotionControl axes from the user's point of view as well as the reaction on exceptional cases 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 concrete axis. 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 408](#),
- [MC_ReadAxisError page 410](#)
- [MC_ReadStatus page 411](#),

RMB_PLCopenFieldBus.library

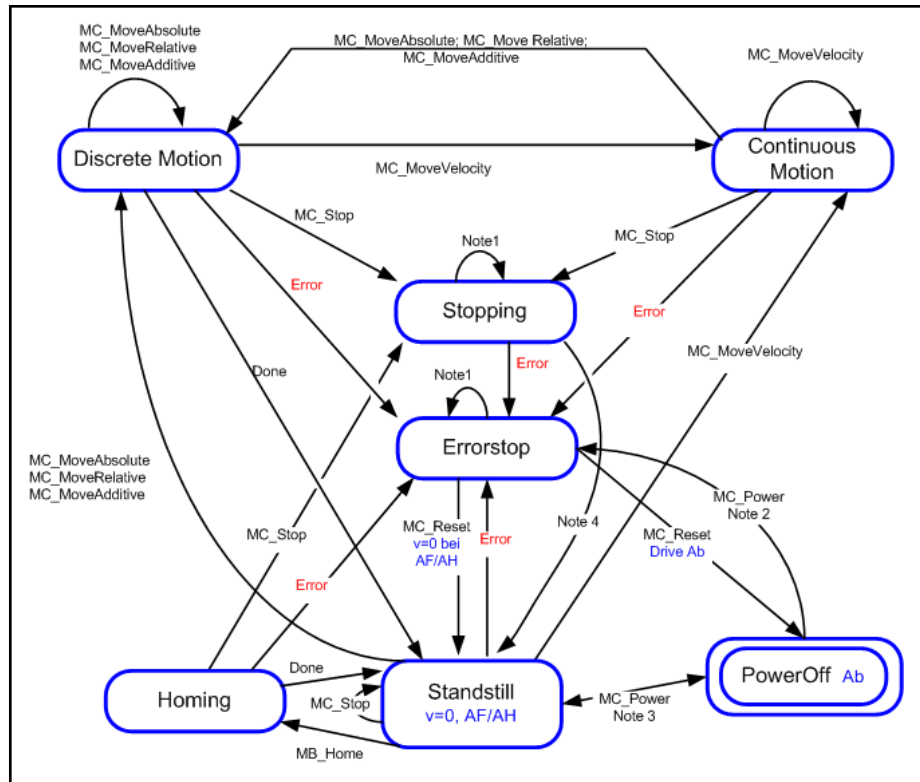


Fig. 7-1: MotionControl state diagram of an field bus axis

Key (Note 1) - All function blocks can be called without being processed except for MC_Reset and Error. They initiate 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 time depending on the programmed error reaction. However, the state machine remains in ErrorStop and when power is switched off, MC_Reset results in Power-Off state.

	Function
MC_Power page 359	controls the power adding (on or off).
MC_Reset page 360	implements the transition from the "ErrorStop" state to the "StandStill" state by resetting all axis-internal errors.
MC_Stop page 362	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. Another command can only be started after the axis has reached the "Stopping" state.
MB_Stop page 363	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. As long as "ExecuteLock" is TRUE, the axis remains in the "Stopping" state. In this state, the axis cannot be moved by any other command.

	Function
MB_Home page 365	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 366	The function block copies the absolute predetermined "position" to the "target position" and approaches it.
MC_MoveAdditive page 368	The function block adds the "distance" to the "target position" and approaches it.
MC_MoveRelative page 371	The function block determines the new "target position" by adding the "distance" to the previous "actual position" and approaching it.
MC_MoveVelocity page 374	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. 7-6: 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** is required for this function block.
The activation of the function block is a precondition for every motion, acceleration or deceleration.
A special situation is given for the function block [MC_Stop page 362](#). The position is kept active by the drive. That means that the MC_Power function block cannot be switched off when MC_Stop is active.



In association with a virtual axis, the function block leads to an error message.

Library	Area
RMB_PLCOpenFieldBus	Motion_FBs

Tab. 7-7: Library assignment

Interface

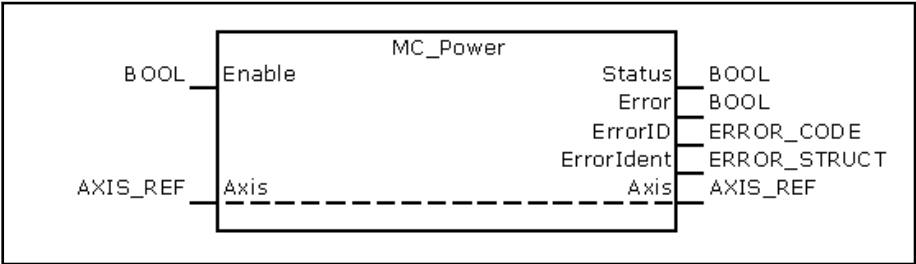


Fig. 7-2: MC_Power FB

RMB_PLCOpenFieldBus.library

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Enable	BOOL	As long as "Enable" is TRUE, power is added on
VAR_OUTPUT	Status	BOOL	Actual state of power adding
	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

Tab. 7-8: Interface of the MC_Power FB

7.4.3 MC_Power, Special Features of Field Bus Drives

- Error handling:**
- An error message is issued if the "Status" is still FALSE after a waiting period has elapsed with "Enable" set. This indicates that a hardware problem occurred when power was added.
 - An error message is also issued if the attempt is made 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 appears

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_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	Area
RMB_PLCOpenFieldBus	Motion_FBs

Tab. 7-9: Library assignment

Interface description

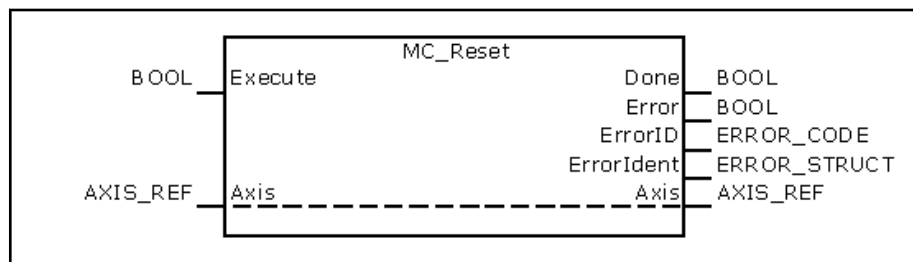


Fig. 7-3: MC_Reset FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	It starts the action at a rising edge
VAR_OUTPUT	Done	BOOL	Errors are reset
	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

Tab. 7-10: Interface of the MC_Reset FB

Signal-time diagram

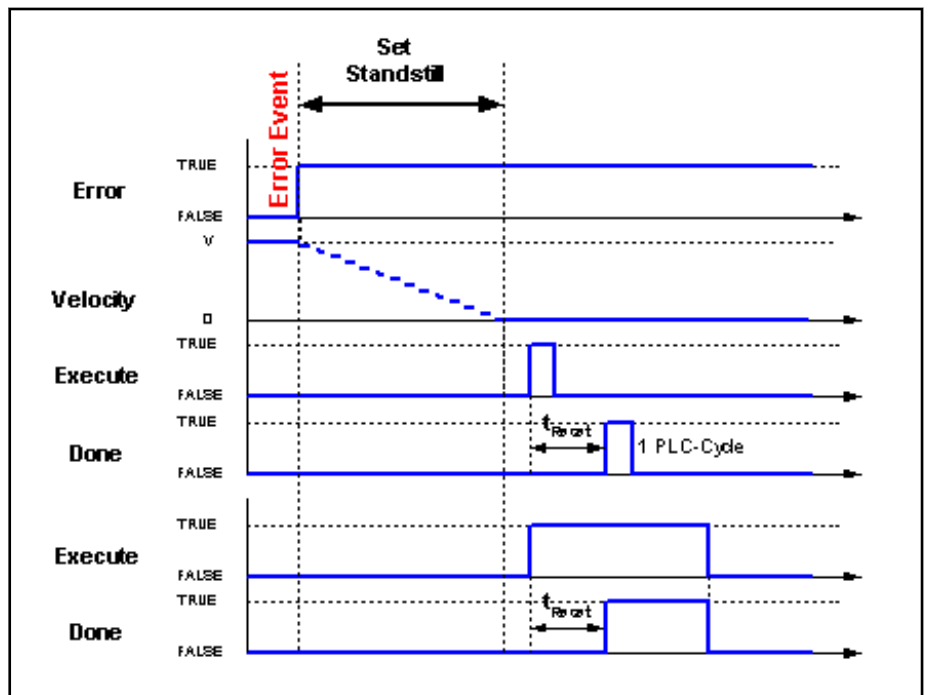


Fig. 7-4: Time diagram of MC_Reset

7.4.5 MC_Reset, Special Features of Field Bus Drives

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.

Error handling

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.

RMB_PLCopOpenFieldBus.library

7.4.6 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.

 As long as "Execute" is TRUE, the axis remains in "Stopping" state.

Another command can only be started after the axis has reached the "Stopping" state.

In the "Stopping" state, it is not possible to switch off the axis power ([MC_Power page 359](#)). 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_PLCopOpenFieldBus	Motion_FBs

Tab. 7-11: Library assignment

Interface Description

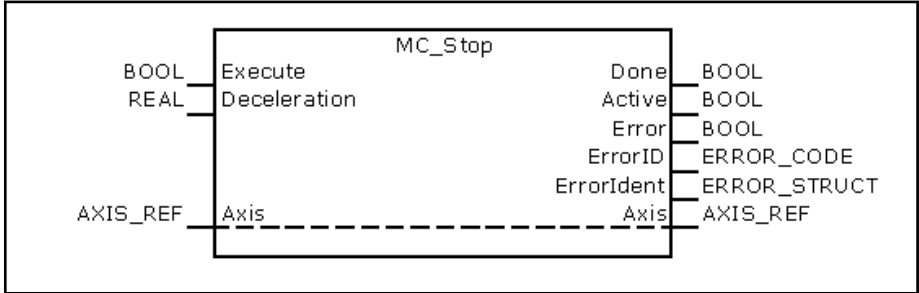


Fig. 7-5: MC_Stop function block

	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 then 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 the error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 7-12: 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

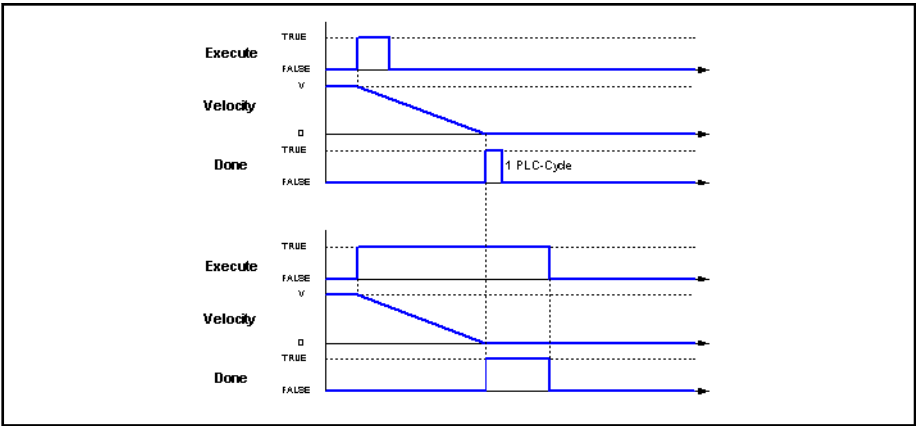


Fig. 7-6: Time diagram of MC_Stop

7.4.7 MC_Stop, Special Features of Field Bus Drives

Error handling 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.

7.4.8 MB_Stop

Brief Description This function block implements a controlled motion stop and transfers the axis to the 'Stopping' state. As a result, any function block processing is stopped. When standstill is reached, the "Done" output is set. The position is kept active.



As long as "ExecuteLock" is TRUE, the axis remains in the "Stopping" state. In this state, the axis cannot be moved by any other command.

In contrast to "MC_Stop page 362", 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 359). 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	Area
RMB_PLCOpenFieldBus	Motion_PLCOpen

Tab. 7-13: Library assignment

RMB_PLCopOpenFieldBus.library

Interface description

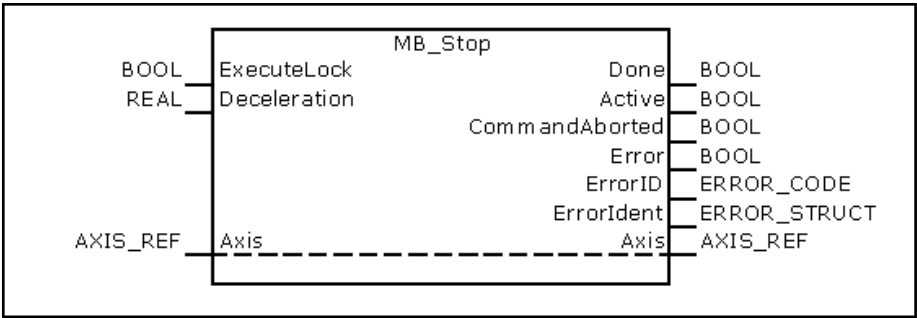


Fig. 7-7: MC_Stop FB

	Name	Type	Comment ²⁾
VAR_IN_OUT	Axis	AXIS_REF	It 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 then 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	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

Tab. 7-14: Interface of the MB_Stop FB

Signal-time diagram

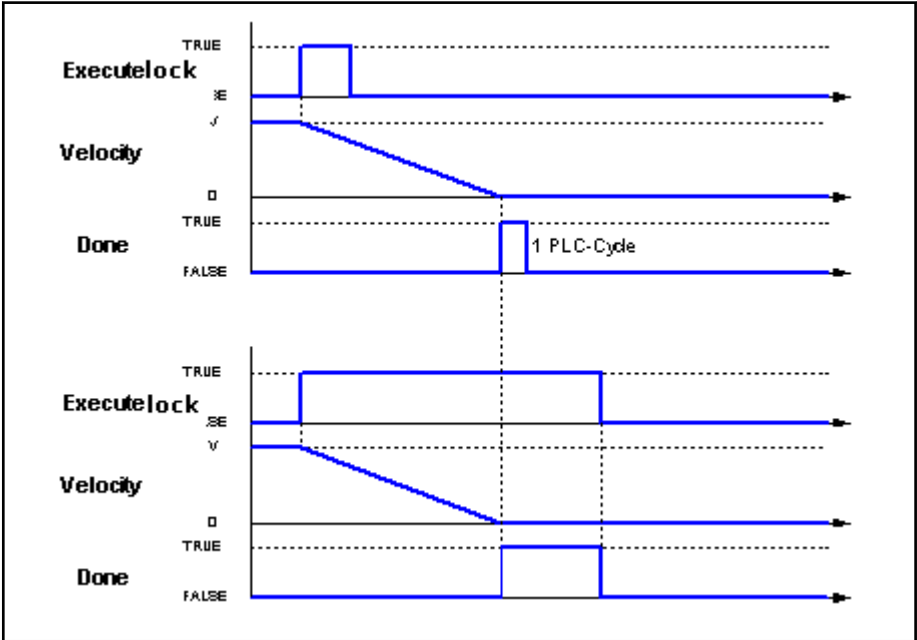


Fig. 7-8: Time diagram of MB_Stop

2) The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

7.4.9 MB_Stop, Special Features of Field Bus Drives

Error handling 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.

7.4.10 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.

Library	Area
RMB_PLCoPenFieldBus	Motion_FBs

Tab. 7-15: Library assignment

Interface description

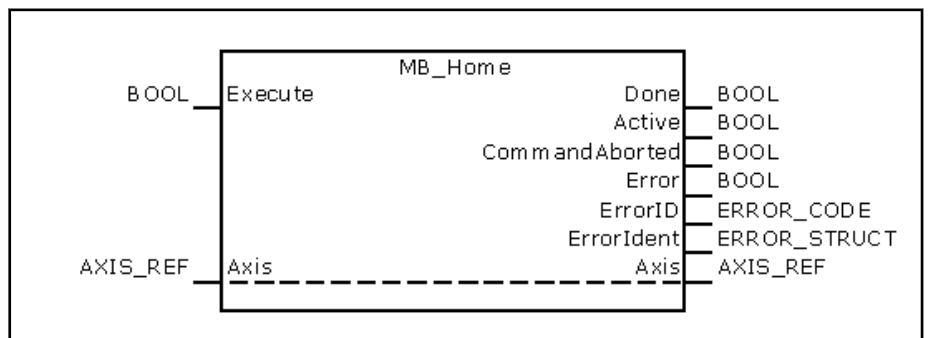


Fig. 7-9: MB_Home FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	It 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	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

Tab. 7-16: Interface of the MB_Home FB

RMB_PLCopenFieldBus.library

Signal-time diagram

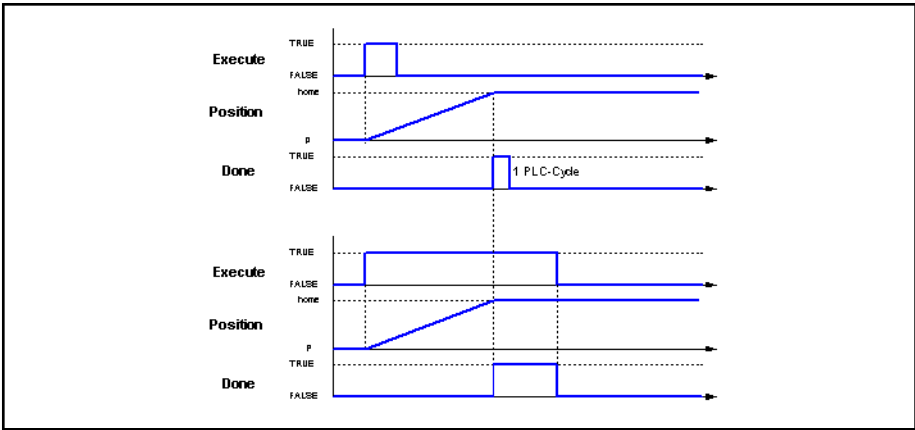


Fig. 7-10: Time diagram of MB_Home

7.4.11 MB_Home, Special Features of Field Bus Drives

Error handling 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.

7.4.12 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 359, (for real axes).

Library	Area
RMB_PLCopenFieldBus	Motion_FBs

Tab. 7-17: Library assignment

Interface description

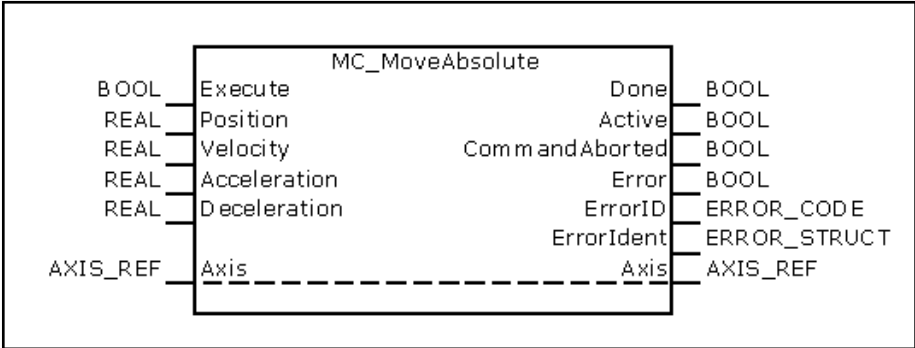


Fig. 7-11: MC_MoveAbsolute FB

	Name	Type	Comment ³⁾
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	It 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	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

Tab. 7-18: Interface of the MC_MoveAbsolute FB

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 activates Instance2 with the target position 10,000.
2. The right side of the time diagram shows how Instance2 is already activated while Instance1 is still running. In this case, 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.

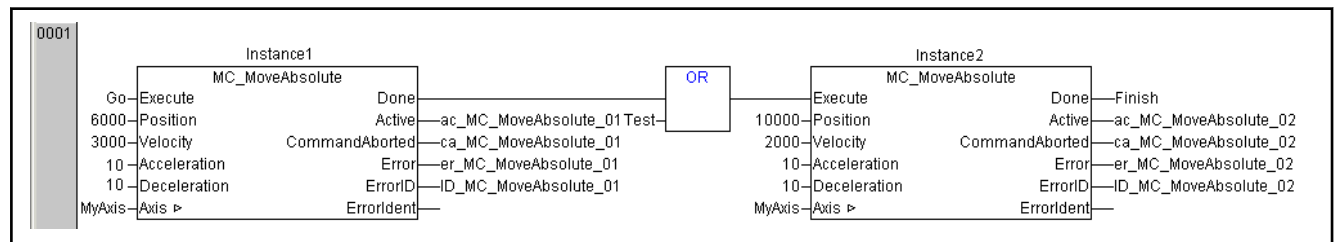


Fig. 7-12: Two instances of MC_MoveAbsolute

³⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

RMB_PLCopenFieldBus.library

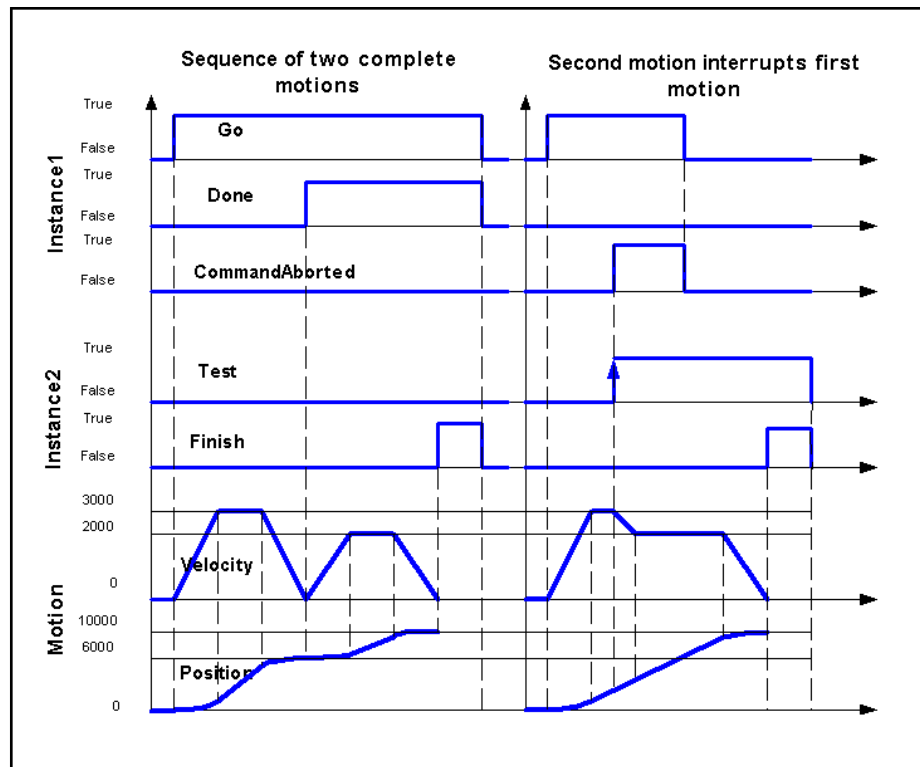


Fig. 7-13: Time diagram: Two instances of MC_MoveAbsolute

7.4.13 MC_MoveAbsolute, Special Features of Field Bus Drives

- 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 just 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.
- Error handling** 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.

7.4.14 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 359, (for real axes).

Library	Area
RMB_PLCOpenFieldBus	Motion_FBs

Tab. 7-19: Library assignment

Interface description

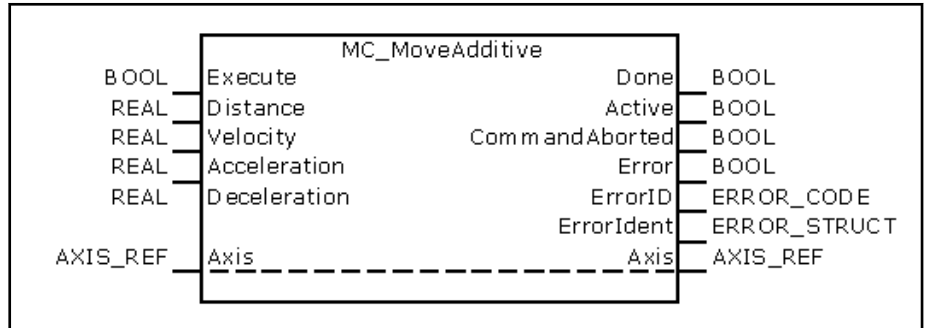


Fig. 7-14: MC_MoveAdditive FB

	Name	Type	Comment ⁴⁾
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	It 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	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

Tab. 7-20: Interface of the MC_MoveAdditive FB

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 side of the time diagram shows a case in which the second function block is started after the first. The first function block reaches the desired distance of 6,000 (at a velocity of 0). The "Done" output causes the second function block to travel the distance of 10,000.

⁴⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

RMB_PLCoenFieldBus.library

2. The right side 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 6,000 and moves the axis to the resulting position 10,000 at a velocity specified at the second function block.

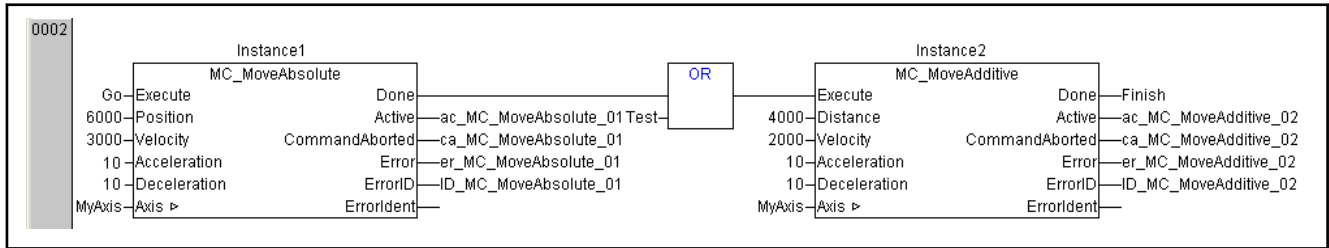


Fig. 7-15: Instances of MC_MoveAbsolute and MC_MoveAdditive

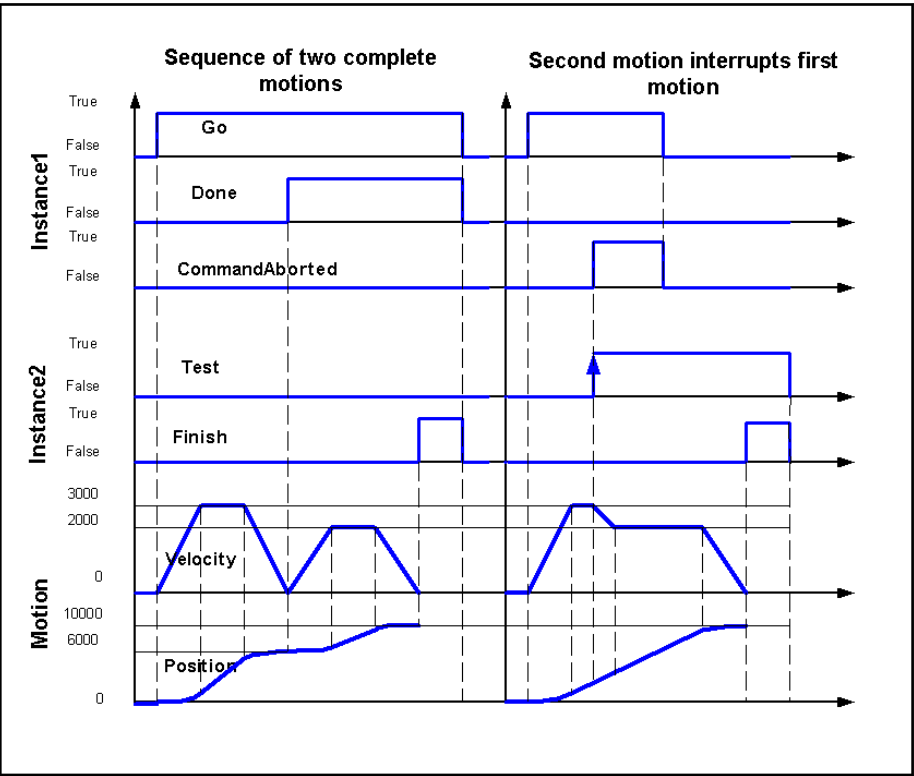


Fig. 7-16: Time diagram: Instances of MC_MoveAbsolute and MC_MoveAdditive

7.4.15 MC_MoveAdditive, Special Features of Field Bus Drives

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 just interrupted. The target position remains in the axis.

If MC_MoveAdditive (Vcom ≠ 0) is called subsequently, the remaining distance is traveled. The target position is only overwritten when another Motion function block is called.

Error handling 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.

7.4.16 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 359](#) (for each real axes).

Library	Area
RMB_PLCOpenFieldBus	Motion_FBs

Tab. 7-21: Library assignment

Interface description

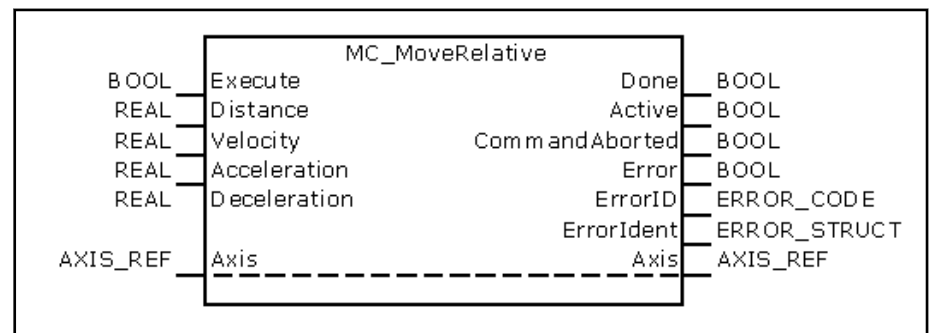


Fig. 7-17: MC_MoveRelative FB

	Name	Type	Comment ⁵⁾
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	It 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

⁵⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

RMB_PLcOpenFieldBus.library

	Name	Type	Comment ⁵⁾
	CommandAborted	BOOL	Command aborted by subsequent command
	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

Tab. 7-22: Interface of the MC_MoveRelative FB

Signal-time diagram

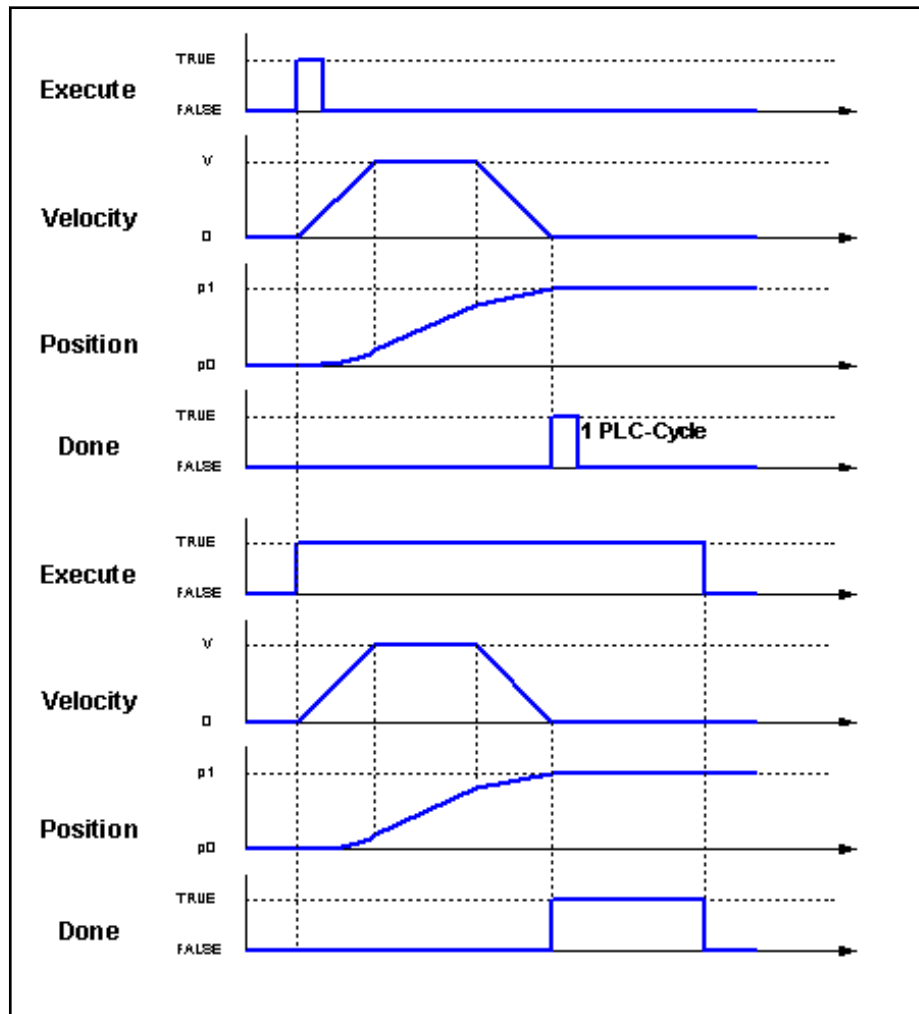


Fig. 7-18: Time diagrams for MC_MoveRelative

Implementation Example

The following example shows the combination of two MC_MoveRelative function block instances.

1. The left side of the time diagram shows a case in which Instance2 is called after Instance1 has been 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 side shows how Instance2 is already activated by the "Test" signal while Instance1 is still running. In this case, the first motion is

⁵⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

aborted during the constant velocity of Instance1. Instance1 adds its distance of 4,000 to the current position (3,250). The axis moves to the resulting position of 7,250.

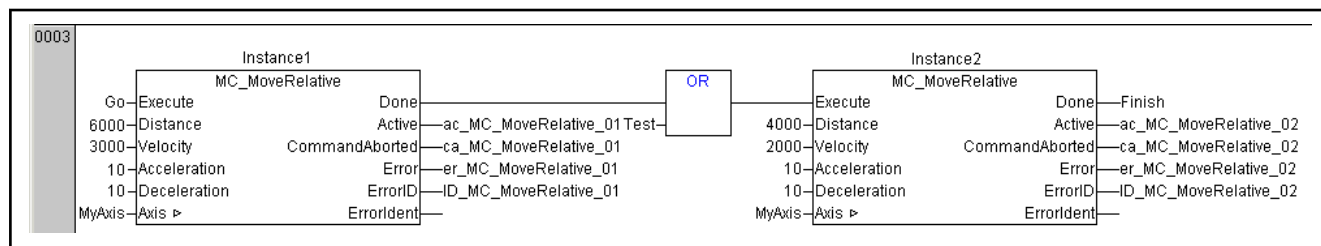


Fig. 7-19: Two instances of MC_MoveRelative

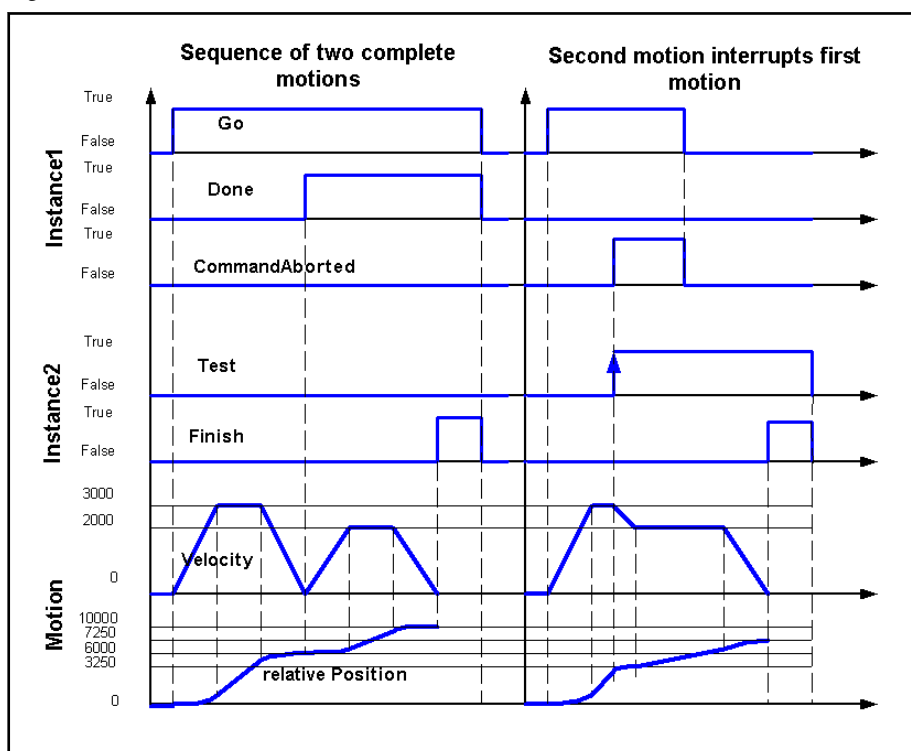


Fig. 7-20: Time diagram: Two instances of MC_MoveRelative

7.4.17 MC_MoveRelative, Special Features of Field Bus Drives

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 just interrupted. The target position remains in the axis.

If MC_MoveAdditive (Vcom ≠ 0) is called subsequently, the remaining distance is traveled. The target position is only overwritten when another Motion function block is called.

Error handling The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

RMB_PLCopenFieldBus.library

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.

7.4.18 MC_MoveVelocity

Brief Description This function block controls an "endless" motion at the respectively defined velocity. To end the motion, the function block instance must 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 359, (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 deactivated, the drive continues to rotate at the same velocity.

Library	Area
RMB_PLCopenFieldBus	Motion_FBs

Tab. 7-23: Library assignment

Interface description

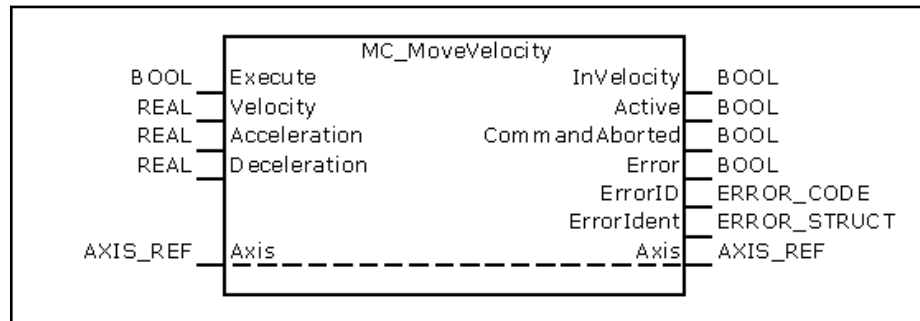


Fig. 7-21: MC_MoveVelocity FB

	Name	Type	Comment ⁶⁾
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	It 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)

⁶⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

	Name	Type	Comment ⁶⁾
	Active	BOOL	Data processing runs when preprocessing is complete
	CommandAborted	BOOL	Command aborted by subsequent command
	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

Tab. 7-24: Interface of the MC_MoveVelocity FB

Signal-time diagram

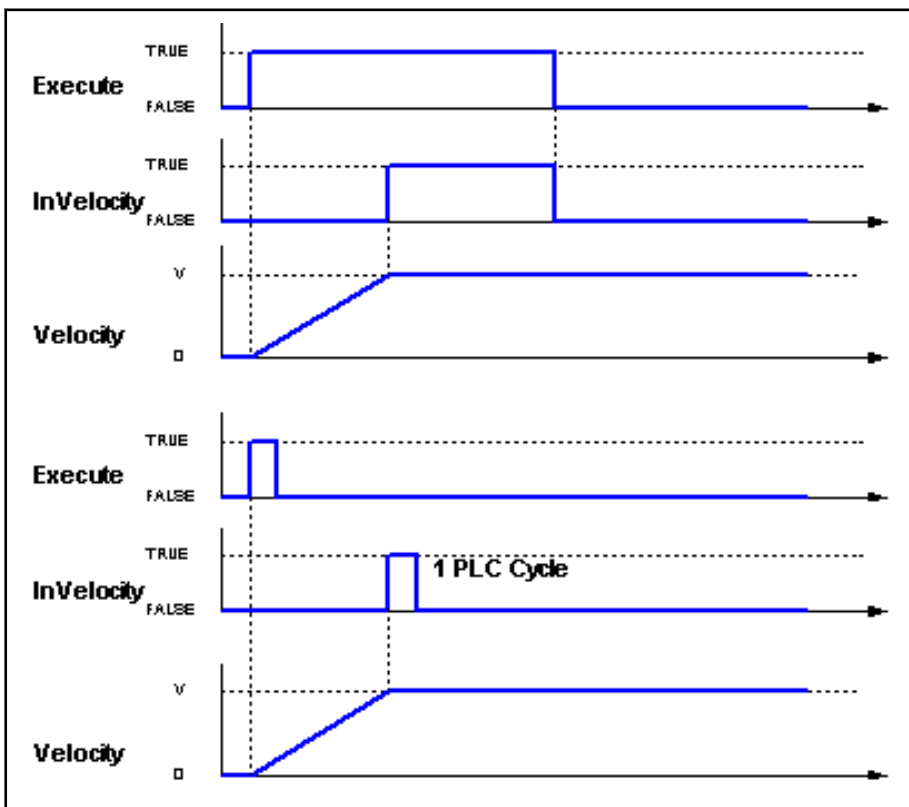


Fig. 7-22: Time diagram of MC_MoveVelocity (without aborted command)

⁶⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

RMB_PLCopenFieldBus.library

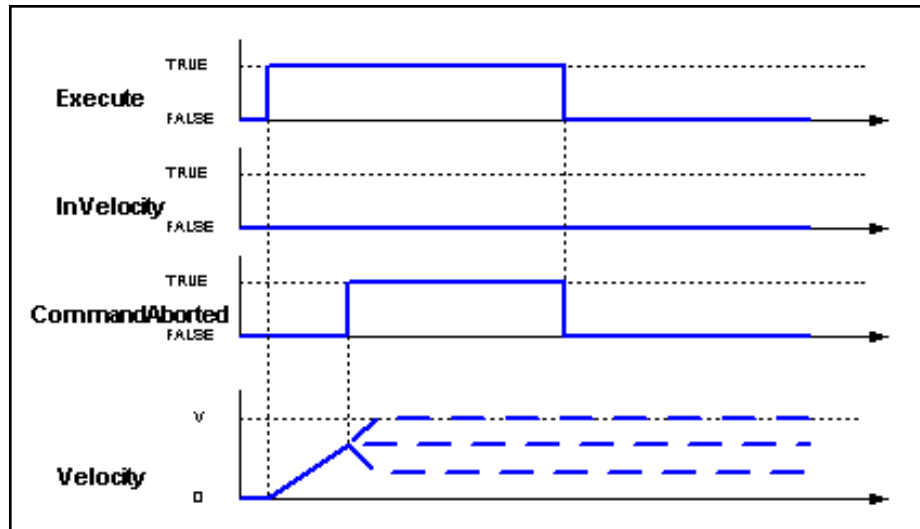


Fig. 7-23: Time diagram of MC_MoveVelocity (with aborted command)

Implementation example:

The figure below shows two examples of two combined instances of the MC_MoveVelocity function block:

1. The left side of the time diagram shows a case in which the second instance is called after the job of the first instance has been completed. 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 desired 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. Now, the second function block is running and decelerates to a velocity of 2,000.

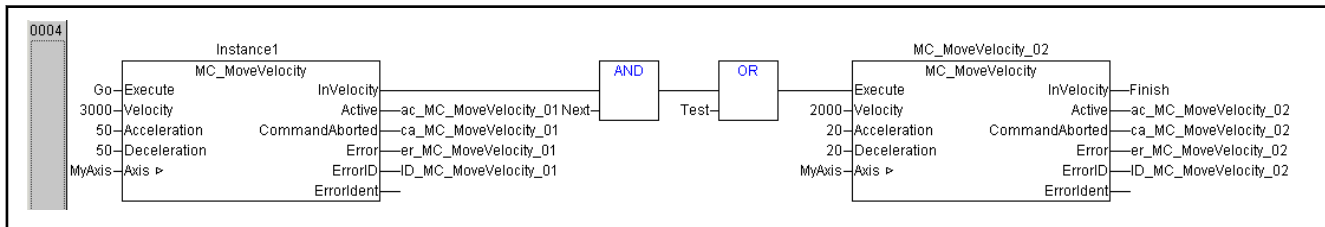


Fig. 7-24: Two instances of MC_MoveVelocity

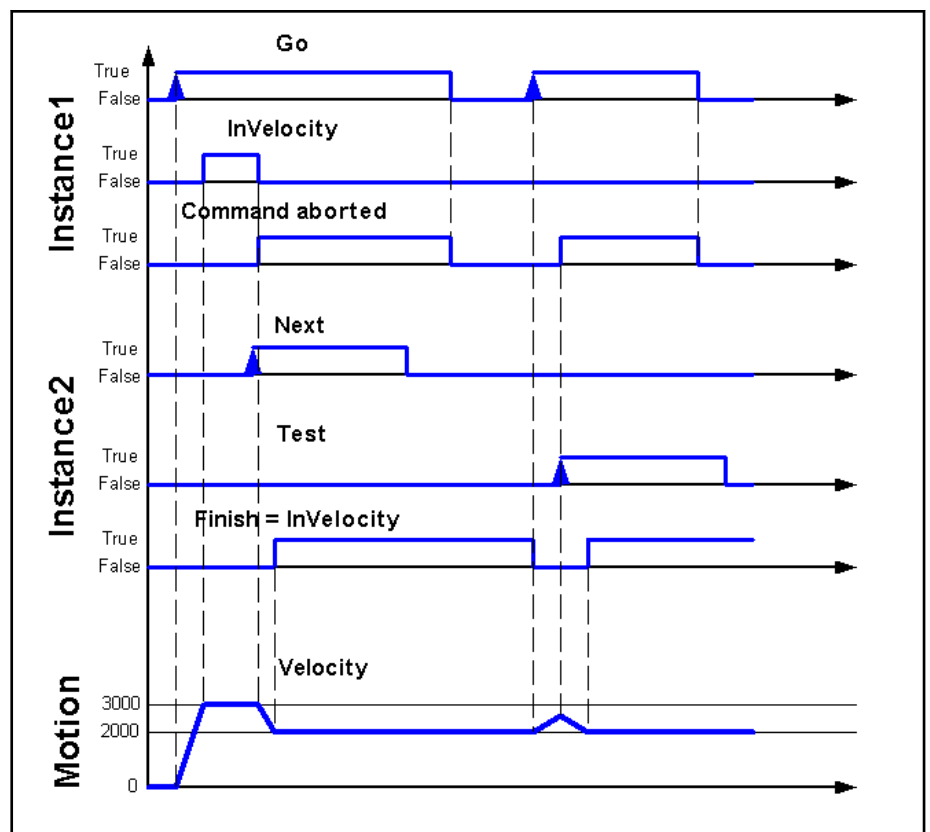


Fig. 7-25: Time diagram: Two instances of MC_MoveVelocity

7.4.19 MC_MoveVelocity, Special Features of Field Bus Drives

Error handling 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. The size of the parameter channel can be set to 4...16 bytes. The parameter channel consists of a control or status word and up to 7 words for data.

The parameter channel is always at the start of the cyclic data channel. The parameter channel length is configured in the master, the XLC/MLC

RMB_PLCoenFieldBus.library

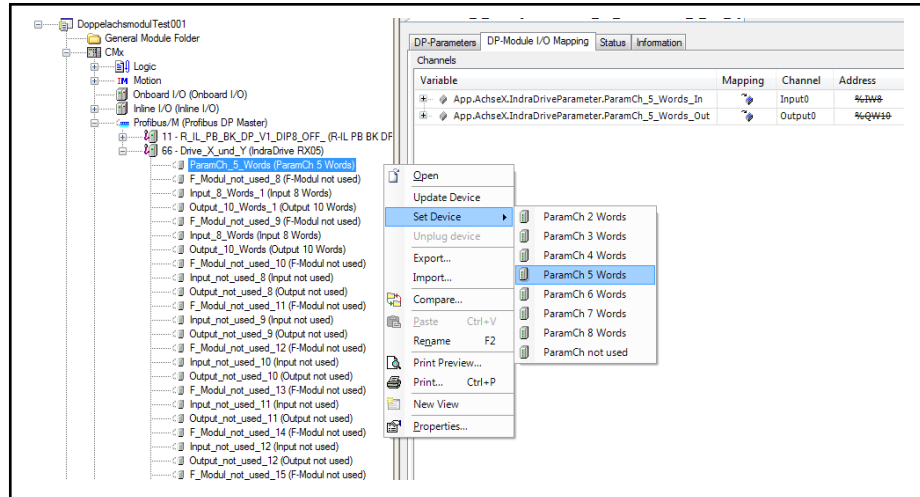


Fig. 7-26: Parameter channel in the cyclic data channel



Fig. 7-27: Parameter channel configuration



The function blocks mentioned in the following are used to transfer parameters between control and drive.

They can only be used for IndraDrive C/M from MPx2VRS, IndraDrive Cs from MPx16VRS coupled via PROFIBUS.

	Function
MB_ReadListParameter, page 381,	returns the value of a user-specific parameter. A pointer addressing a memory space of no more than 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 384,	returns the values of a section of a user-specific parameter. A pointer addressing a memory space of no more than 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 387	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 388	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 390	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 391	The function block returns the minimum value allowed of a parameter specified by the user. The return value is represented as DINT number.

	Function
MB_ReadRealParameter, page 394	returns the value of a user-specific parameter. The return value is represented as REAL number.
MB_ReadParameter, page 393	returns the value of a user-specific parameter. The return value is represented as DINT number.
MB_ReadSercosAttribute, page 396	allows reading the sercos attributes of a parameter. The return value is represented as a DWORD.
MB_ReadStringParameter, page 397,	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 399	writes memory content of a defined length (NoOfBytes) provided from the initial address (ValueAdr) to a user-specific parameter.
MB_WriteListSegParameterPtr, page 401	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 404	writes a DINT value in a parameter specified by the user.
MB_WriteRealParameter, page 405	writes a REAL value to a user-specific parameter.
MB_WriteStringParameter, page 407	function block writes a characters string with a maximum of 255 characters to a user-specific parameter.
MC_ReadActualPosition, page 408	reads the current position of the drive.
MC_ReadAxisError, page 410,	queries the error state of the addressed axis. One instance of the function block is to be used for each axis.
MC_ReadStatus, page 411	reads the current status of the axis and outputs it in a decoded way.

Tab. 7-25: 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 353,](#)
- [GVL_P_Param_ID, field bus, page 354,](#)

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
```

RMB_PLCopenFieldBus.library

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. 7-26: *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

```

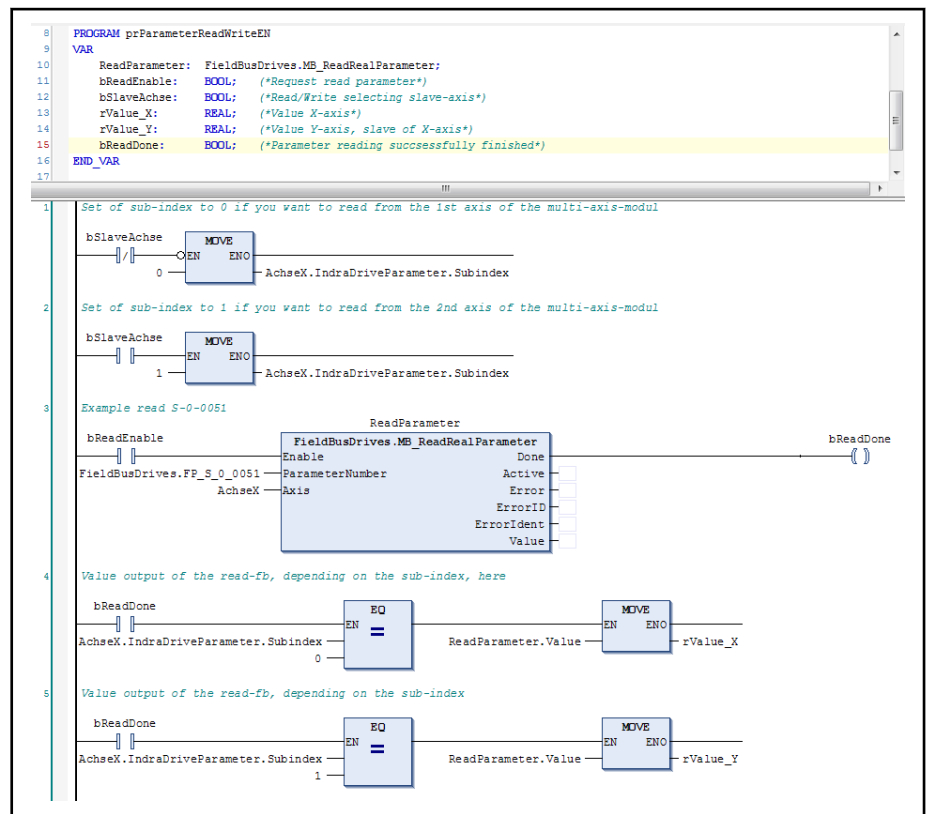


Fig. 7-28: Program sample parameter reading of twin or multi-axis module

For all PLCopen parameter blocks, the subindex (default = 0) has to be specified in the AXIS_REF of the multi-axis module (default = 0).

This means that access to only one axis is possible in case of a twin axis or CDD link.

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 no more than 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.

RMB_PLCopenFieldBus.library

Library	Area
RMB_PLCopenFieldBus	Parameter_FBs

Tab. 7-27: Library assignment

Interface description

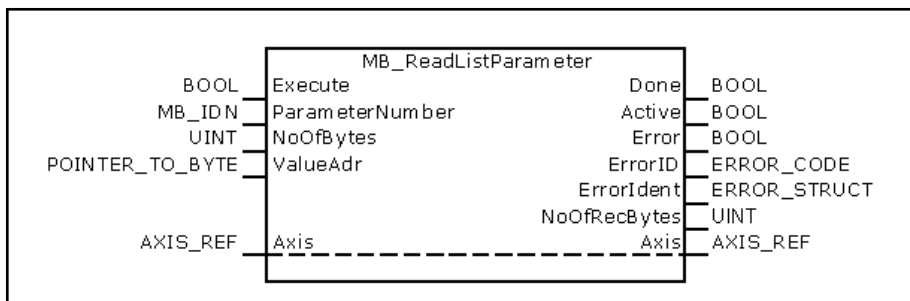


Fig. 7-29: MB_ReadListParameter FB

	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 desired (e.g. FP_S_0_0000, also refer to Parameter Constants page 379)
	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	Data processing runs when preprocessing is complete
	Error	BOOL	It 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_ReadListParameter FB

Signal-time diagram

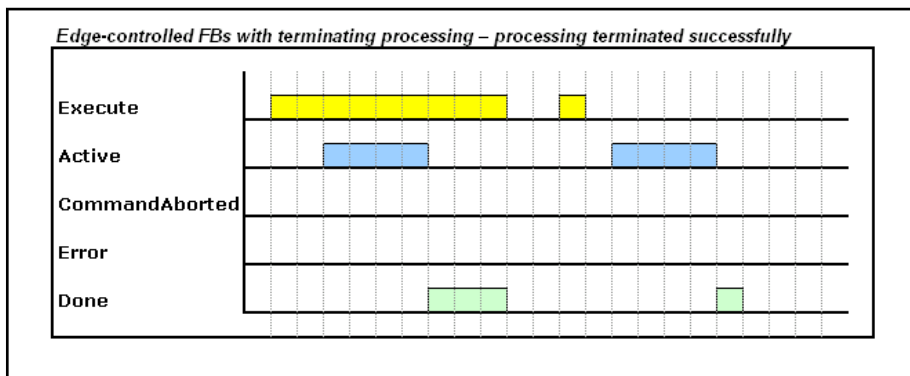


Fig. 7-30: Default signal-time diagram

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

```

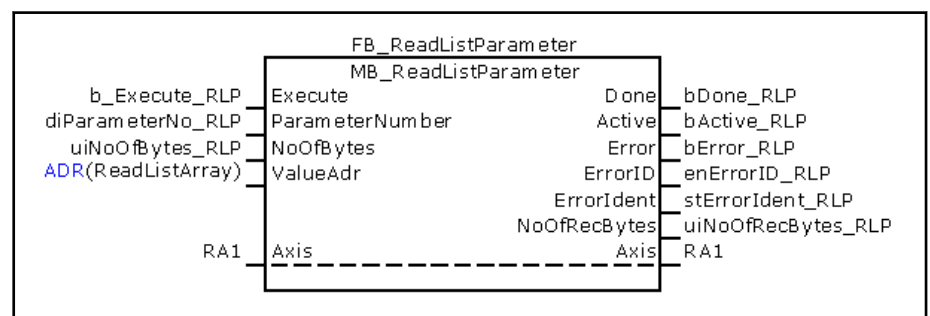
Program example in the function block diagram

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 *)

```

7.5.5 MB_ReadListParameter, Special Features of the Field Bus Drives



The MB_ReadListParameter function block only runs together with the IndraDrive drives.

Error handling

The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

RMB_PLCOpenFieldBus.library

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_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	Area
RMB_PLCOpenFieldBus	Parameter_FBs

Tab. 7-29: Library assignment

Interface description

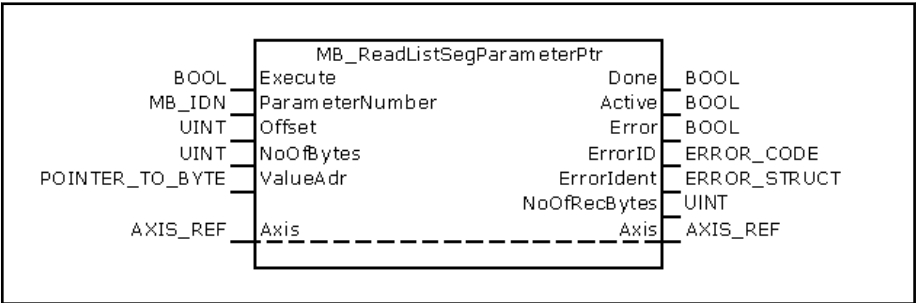


Fig. 7-32: MB_ReadListSegParameterPtr FB

	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 desired (e.g. FP_S_0_0000, also refer to Access to Data of a Drive (IndraDrive) page 379)
	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)
	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	Data processing runs when preprocessing is complete
	Error	BOOL	It 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. 7-30: Interface of the MB_ReadListSegParameterPtr FB

Signal-time diagram

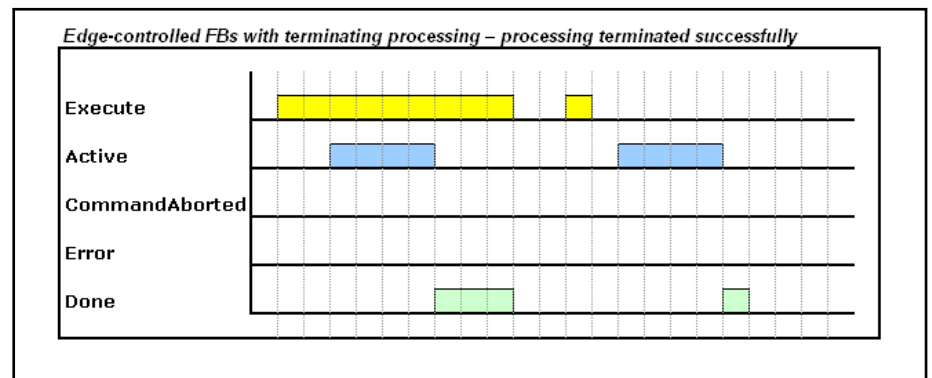


Fig. 7-33: Default signal-time diagram

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

```


RMB_PLCLopenFieldBus.library

```

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

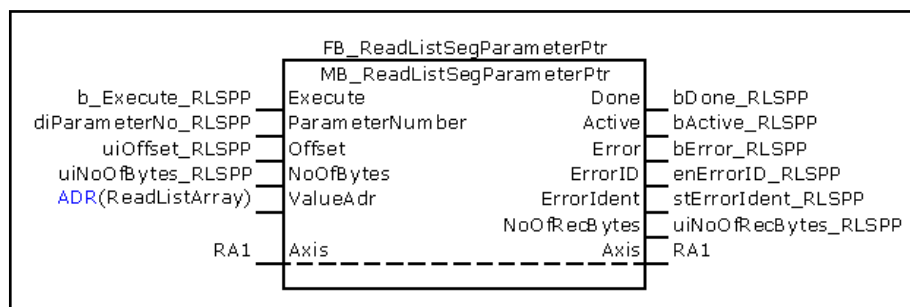


Fig. 7-34: FBD program example: MB_ReadListSegParameterPtr

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
)

```

7.5.7 MB_ReadListSegParameterPtr, Special Features of the Field Bus Drives



The MB_ReadListSegParameterPtr function block only runs together with the IndraDrive drives.

Error handling

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_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	Area
RMB_PlcOpenFieldBus	Parameter_FBs

Tab. 7-31: Library assignment

Interface description

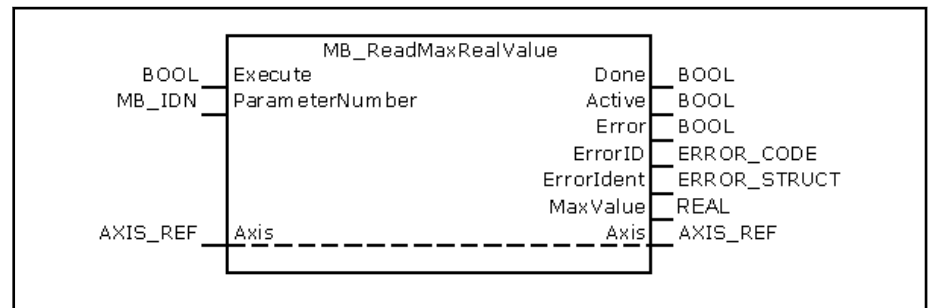


Fig. 7-35: MB_ReadMaxRealValue FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	The input variables (0/1 transition) are registered with a positive edge at "Execute". Then, reading runs irrespective of the variable value.
	Parameter number	MB_IDN	Number of the parameter desired, (also refer to Parameter Constants page 379)
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 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. 7-32: Interface of the MB_ReadMaxRealValue FB

RMB_PLCoOpenFieldBus.library

Signal-time diagram

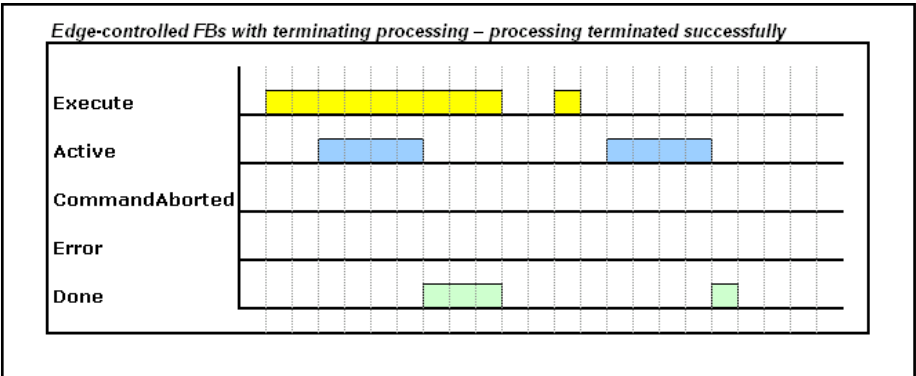


Fig. 7-36: Default signal-time diagram

7.5.9

MB_ReadMaxRealValue, Special Features of the Field Bus Drives

The MB_ReadMaxRealValue function block from the RMB_PLCoOpenFieldBus.Library only runs together with the IndraDrive drives.

Error handling

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_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	Area
RMB_PlcOpenFieldBus	Parameter_FBs

Tab. 7-33: Library assignment

Interface description

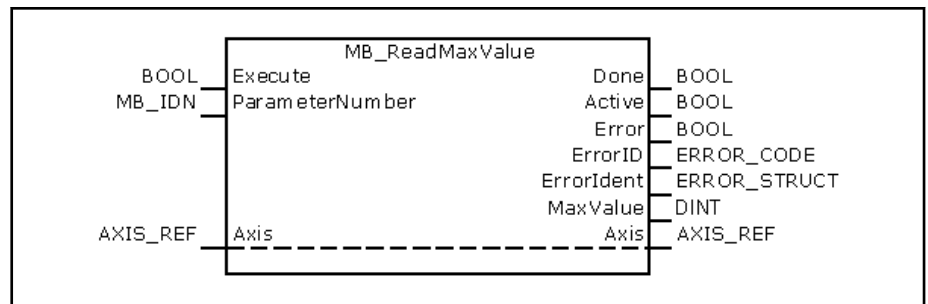


Fig. 7-37: MB_ReadMaxValue FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	The input variables (0/1 transition) are registered with a positive edge at "Execute". Then, reading runs irrespective of the variable value.
	Parameter number	MB_IDN	Number of the parameter desired, (also refer to Parameter Constants page 379)
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 when the function block instance was called
	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. 7-34: Interface of the MB_ReadMaxValue FB

Signal-time diagram

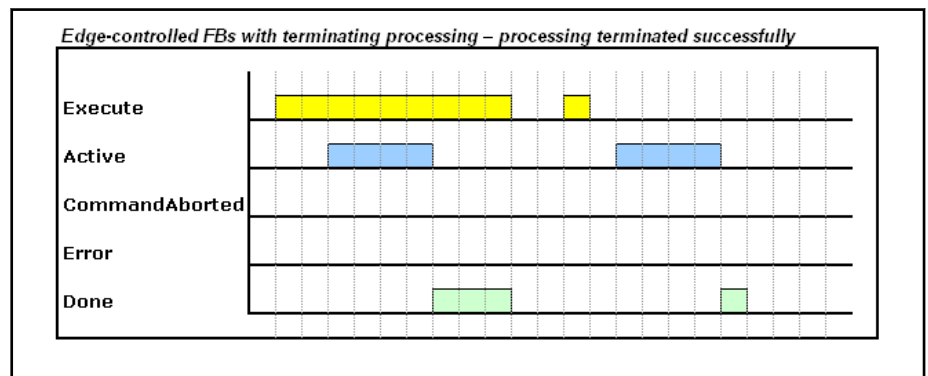


Fig. 7-38: Default signal-time diagram

7.5.11 MB_ReadMaxValue, Special Features of the Field Bus Drives



The MB_ReadMaxValue function block from the RMB_PLCOpenFieldBus.Library only runs together with the IndraDrive drives.

Error handling

The function block generates error messages only at a positive edge or if TRUE is set at the "Execute" input.

RMB_PLcOpenFieldBus.library

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_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	Area
RMB_PlcOpenFieldBus	Parameter_FBs

Tab. 7-35: Library assignment

Interface description

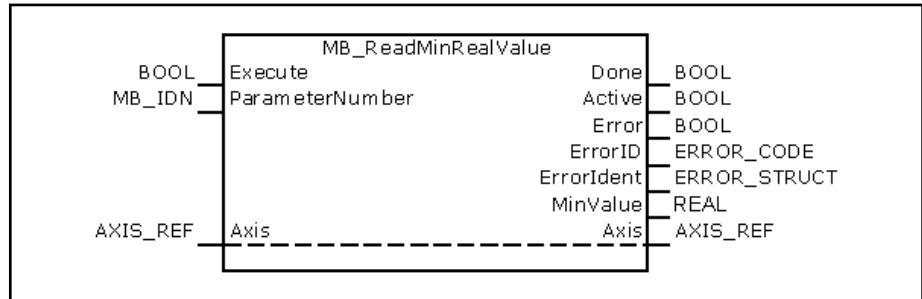


Fig. 7-39: MB_ReadMinRealValue FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	The input variables (0/1 transition) are registered with a positive edge at "Execute". Then, reading runs irrespective of the variable value.
	Parameter number	MB_IDN	Number of the parameter desired, (also refer to Parameter Constants page 379)
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 when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause

	Name	Type	Comment
	ErrorIdent	ERROR_STRUCT	Detailed error information
	MinValue	REAL	If "Done" = TRUE, "MinValue" indicates the minimum input value allowed of the parameter selected.

Tab. 7-36: Interface of the MB_ReadMinRealValue FB

Signal-time diagram

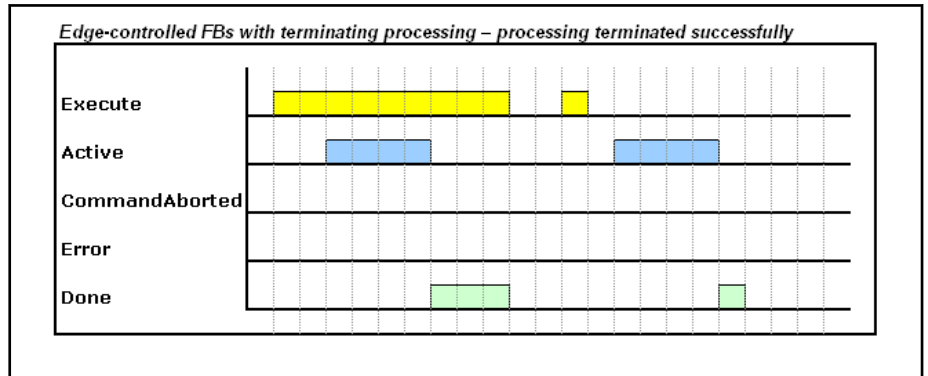


Fig. 7-40: Default signal-time diagram

7.5.13 MB_ReadMinRealValue, Special Features of the Field Bus Drives



The MB_ReadMinRealValue function block from the RMB_PLCoPenFieldBus.Library only runs together with the IndraDrive drives.

Error handling

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_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".

RMB_PLcOpenFieldBus.library

Library	Area
RMB_PlcOpenFieldBus	Parameter_FBs

Tab. 7-37: Library assignment

Interface description

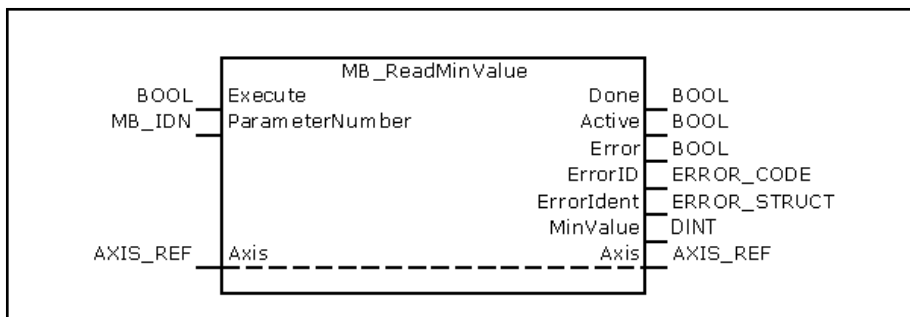


Fig. 7-41: MB_ReadMinValue FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Execute	BOOL	The input variables (0/1 transition) are registered with a positive edge at "Execute". Then, reading runs irrespective of the variable value.
	Parameter number	MB_IDN	Number of the parameter desired, (also refer to Parameter Constants page 379)
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 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. 7-38: Interface of the MB_ReadMinValue FB

Signal-time diagram

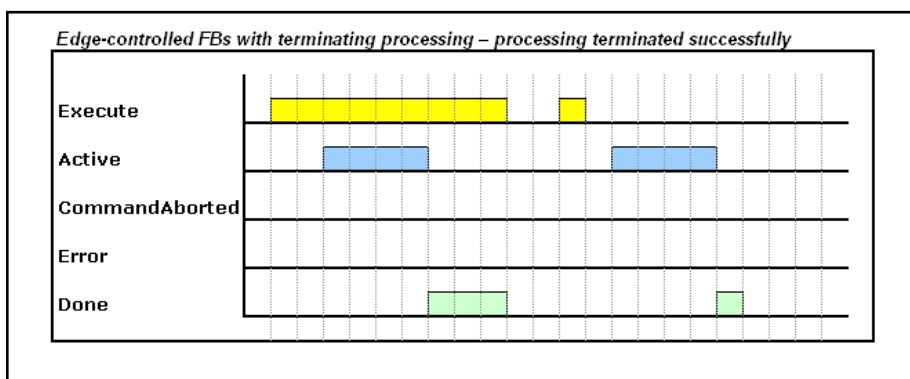


Fig. 7-42: Default signal-time diagram

7.5.15 MB_ReadMinValue, Special Features of the Field Bus Drives



The MB_ReadMinValue function block from the RMB_PLcOpenFieldBus.Library only runs together with the IndraDrive drives.

Error handling 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_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	Area
RMB_PlcOpenFieldBus	Parameter_FBs

Tab. 7-39: Library assignment

Interface description

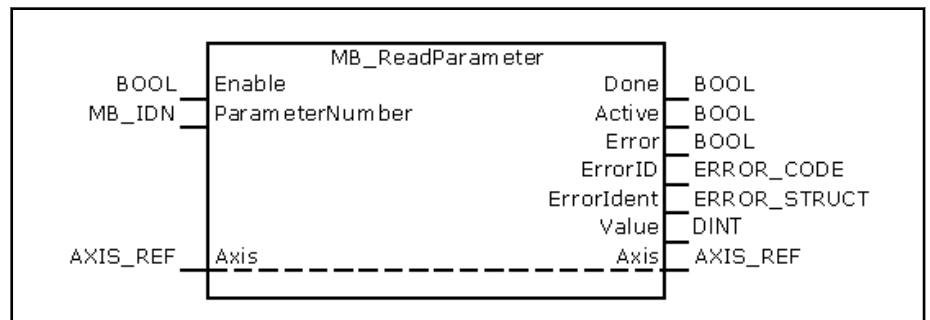


Fig. 7-43: MB_ReadParameter FB

	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 desired (e.g. FP_S_0_0000, also refer to Parameter Constants page 379)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Data processing runs when preprocessing is complete

RMB_PLCopenFieldBus.library

	Name	Type	Comment
	Error	BOOL	It 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. 7-40: Interface of the MB_ReadParameter FB

Signal-time diagram

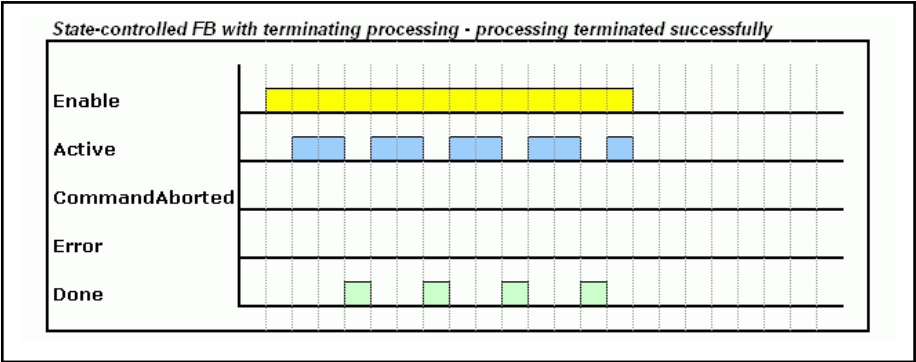


Fig. 7-44: Default signal-time diagram

7.5.17 MB_ReadParameter, Special Features of the Field Bus Drives

The MB_ReadRealParameter function block only runs together with the IndraDrive drives.

Error handling 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.

7.5.18 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	Area
RMB_PLCopenFieldBus	Parameter_FBs

Tab. 7-41: Library assignment

Interface description

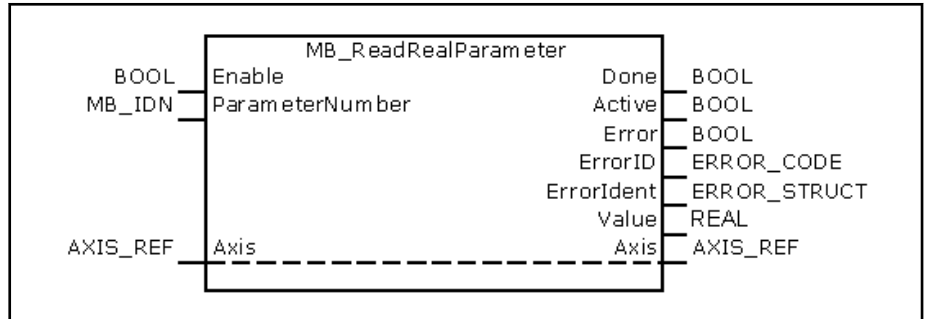


Fig. 7-45: MB_ReadRealParameter FB

	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 desired (e.g. FP_S_0_0000, also refer to Access to Data of a Drive (IndraDrive) page 379)
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 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. 7-42: Interface of the MB_ReadRealParameter FB

Signal-time diagram

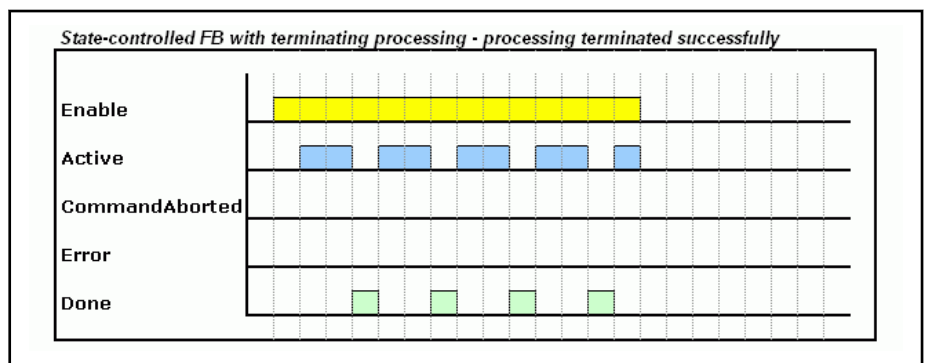


Fig. 7-46: Default signal-time diagram

7.5.19 MB_ReadRealParameter, Special Features of the Field Bus Drives



The MB_ReadRealParameter function block only runs together with the IndraDrive drives.

RMB_PLCopenFieldBus.library

Error handling 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.

7.5.20 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 "Enable".

Library	Area
RMB_PLCopenFieldBus	Parameter_FBs

Tab. 7-43: Library assignment

Interface description

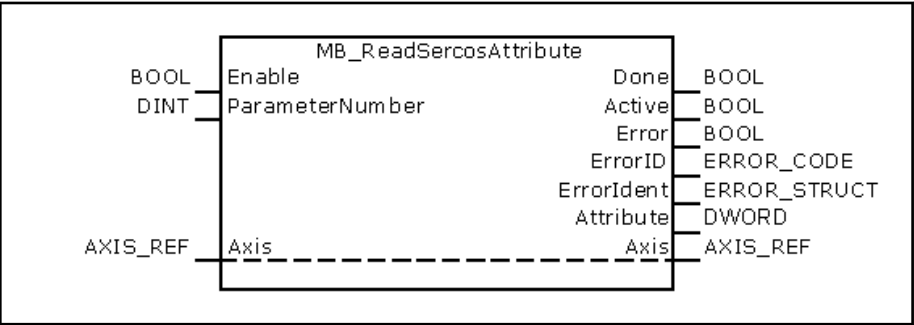


Fig. 7-47: MB_ReadSercosAttribute FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It contains information on the concrete axis
VAR_INPUT	Enable	BOOL	Continuous reading of the parameter attributes as long as "Enable" is TRUE.
	Parameter number	MB_IDN	Number of the parameter desired (e.g. FP_S_0_0000, also refer to Access to Data of a Drive (IndraDrive) page 379)
VAR_OUTPUT	Done	BOOL	A valid value is available
	Active	BOOL	Data processing runs when preprocessing is complete

	Name	Type	Comment
	Error	BOOL	It 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. 7-44: Interface of the MB_ReadSercosAttribute FB

Decoding the attribute

The "IL_SercosAttribute" function block is provided in the "RIL_Utility.library" library for decoding purposes.

Signal-time diagram

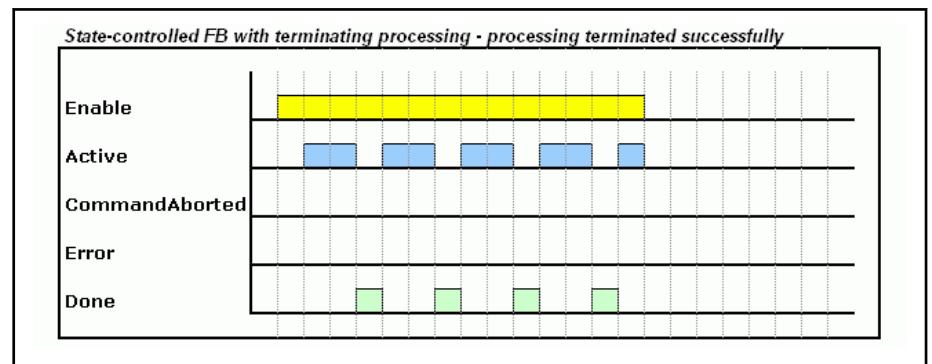


Fig. 7-48: Default signal-time diagram

7.5.21 MB_ReadSercosAttribute, Special Features of the Field Bus Drives



The MB_ReadSercosAttribute function block only runs together with the IndraDrive drives.

Error handling

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.

7.5.22 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.

RMB_PLCopenFieldBus.library

Library	Area
RMB_PLCopenFieldBus	Parameter_FBs

Tab. 7-45: Library assignment

Interface description

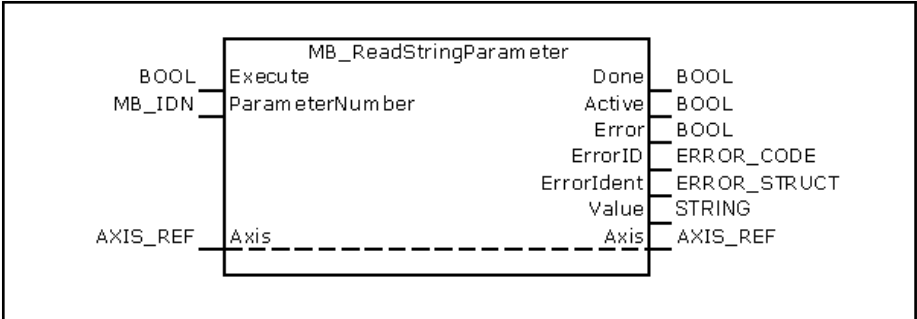


Fig. 7-49: MB_ReadStringParameter FB

	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 desired (e.g. FP_S_0_0000, also refer to Access to Data of a Drive (IndraDrive) page 379)
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 when the function block instance was called
	ErrorID	ERROR_CODE	Brief note on error cause
	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. 7-46: Interface of the MB_ReadStringParameter FB

Signal-time diagram

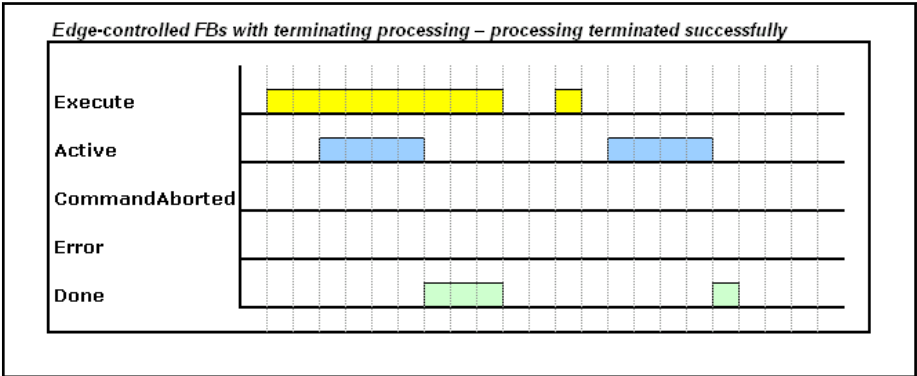


Fig. 7-50: Default signal-time diagram

7.5.23 MB_ReadStringParameter, Special Features of the Field Bus Drives



The MB_ReadStringParameter function block only runs together with the IndraDrive drives.

Error handling 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.

7.5.24 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	Area
RMB_PLCOpenFieldBus	Parameter_FBs

Tab. 7-47: Library assignment

Interface description

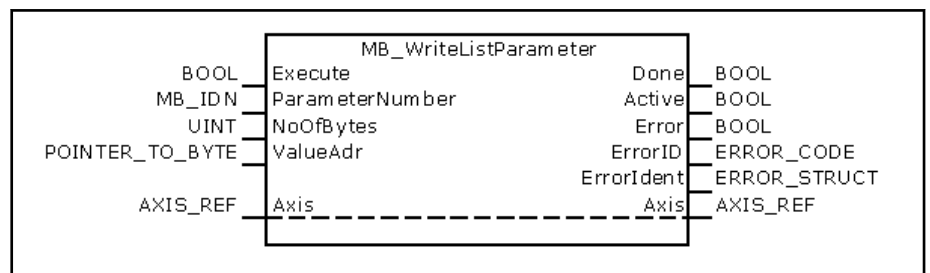


Fig. 7-51: MB_WriteListParameter FB

	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.

RMB_PLCopenFieldBus.library

	Name	Type	Comment
	Parameter number	MB_IDN	Number of the parameter desired (e.g. FP_S_0_0000, also refer to Parameter Constants page 379)
	NoOfBytes	UINT	Number of bytes to be written
	ValueAdr	POINT-ER_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 has been written successfully to the control/drive.
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	It 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_WriteListParameter FB

Signal-time diagram

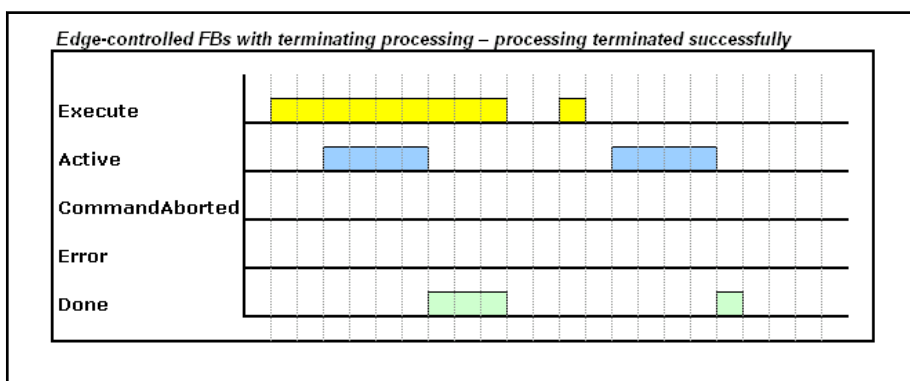


Fig. 7-52: Default signal-time diagram

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

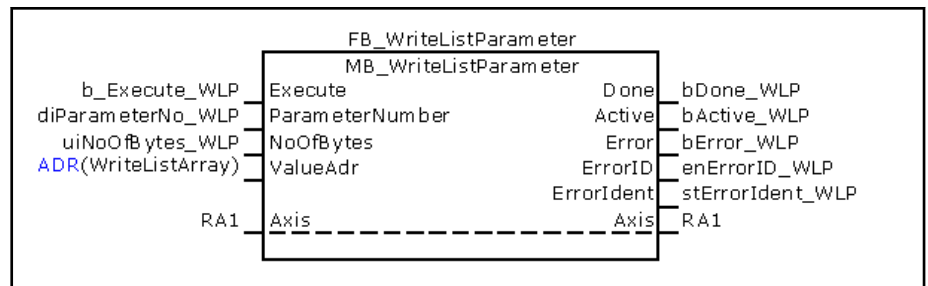


Fig. 7-53: FBD program example: WriteListParameter

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

```

7.5.25 MB_WriteListParameter, Special Features of the Field Bus Drives



The MB_WriteListParameter function block only runs together with the IndraDrive drives.

Error handling

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.

7.5.26 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.

RMB_PLCopenFieldBus.library



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	Area
RMB_PLCopenFieldBus	Parameter_FBs

Tab. 7-49: Library assignment

Interface description

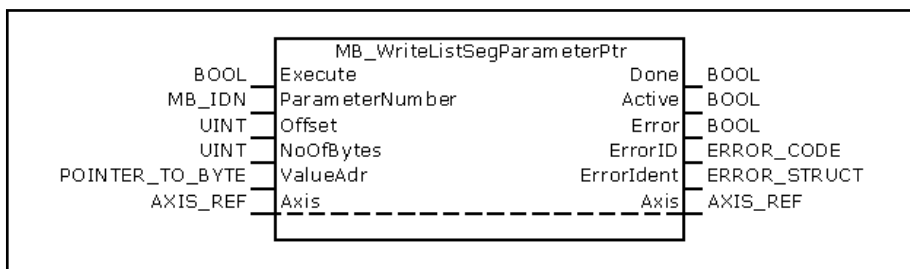


Fig. 7-54: MB_WriteListSegParameterPtr FB

	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 desired (e.g. FP_S_0_0000, also refer to Parameter Constants page 379)
	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.
VAR_OUTPUT	Done	BOOL	The value has been written successfully to the control/drive.
	Active	BOOL	Data processing runs when preprocessing is complete
	Error	BOOL	It 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_WriteListSegParameterPtr FB

Signal-time diagram

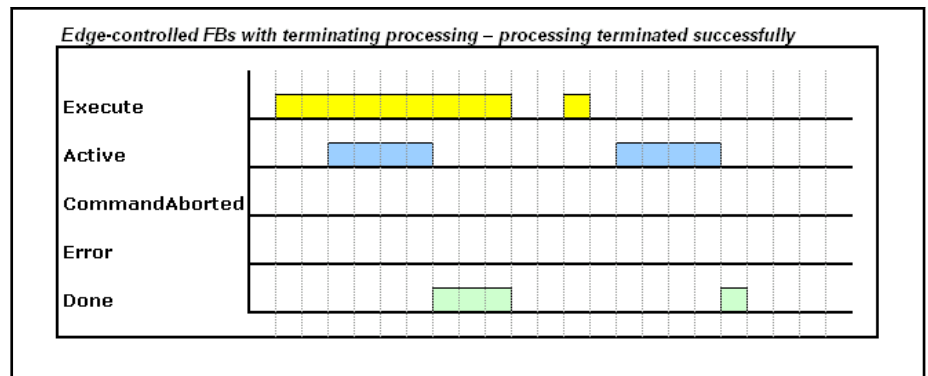


Fig. 7-55: Default signal-time diagram

Program example: MB_WriteList-SegParameterPtr

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

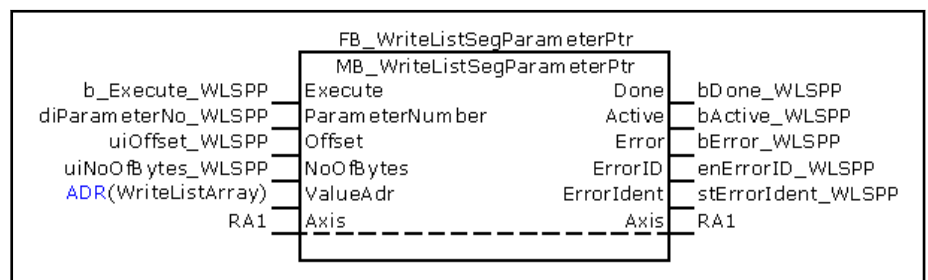


Fig. 7-56: FBD program example: WriteListSegParameterPtr

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
```

RMB_PLCOpenFieldBus.library

```
ErrorId=>          enErrorID_WLSPP,          // Error class
ErrorIdent=>       stErrorIdent_WLSPP);      // Error definition
```

7.5.27 MB_WriteListSegParameterPtr, Special Features of the Field Bus Drives

 The MB_WriteListSegParameterPtr function block only runs together with the IndraDrive drives.

Error handling 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.

7.5.28 MB_WriteParameter

Brief Description The function block writes a value in a parameter specified by the user.

Library	Area
RMB_PLCOpenFieldBus	Parameter_FBs

Tab. 7-51: Library assignment

Interface description

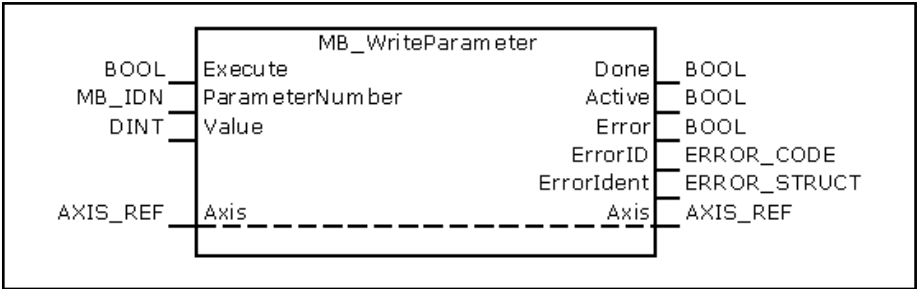


Fig. 7-57: MB_WriteParameter FB

	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 desired (e.g. FP_S_0_0000, also refer to Parameter Constants page 379)

	Name	Type	Comment
	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	Data processing runs when preprocessing is complete
	Error	BOOL	It 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_WriteParameter FB

Signal-time diagram

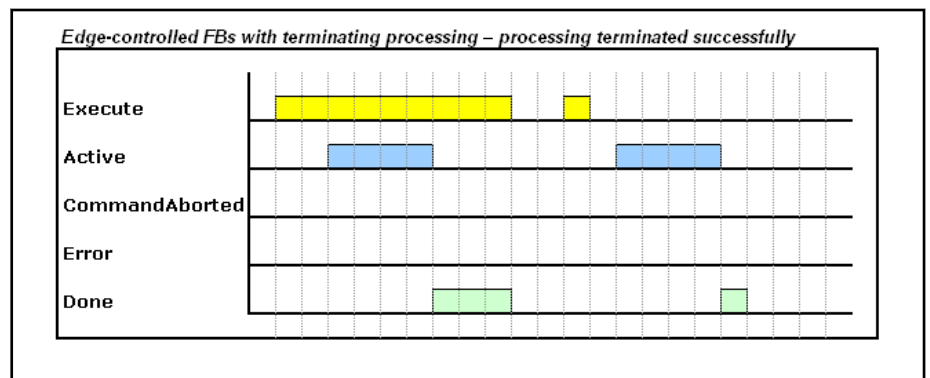


Fig. 7-58: Default signal-time diagram

7.5.29 MB_WriteParameter, Special Features of the Field Bus Drives



The MB_WriteParameter function block only runs together with the IndraDrive drives.

Error handling

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.

7.5.30 MB_WriteRealParameter

Brief Description

This function block writes a value to a user-specific parameter.

RMB_PLCopenFieldBus.library

Library	Area
RMB_PLCopenFieldBus	Parameter_FBs

Tab. 7-53: Library assignment

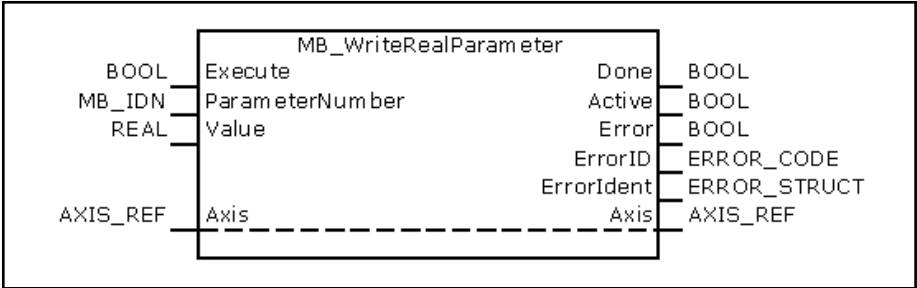


Fig. 7-59: MB_WriteRealParameter FB

	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 desired (e.g. FP_S_0_0000, also refer to Parameter Constants page 379)
	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	Data processing runs when preprocessing is complete
	Error	BOOL	It 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. 7-54: Interface of the MB_WriteRealParameter FB

Signal-time diagram

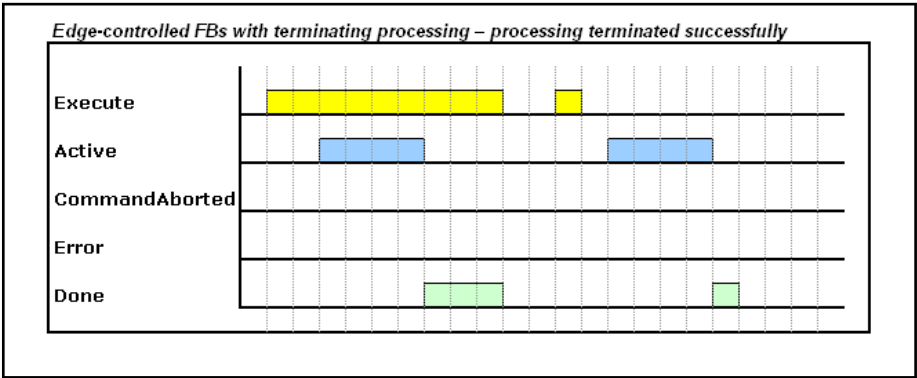


Fig. 7-60: Default signal-time diagram

7.5.31 MB_WriteRealParameter, Special Features of the Field Bus Drives



The MB_WriteRealParameter function block only runs together with the IndraDrive drives.

Error handling 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.

7.5.32 MB_WriteStringParameter

Brief Description The function block writes a character string with a maximum of 255 characters to a user-specific parameter.

Library	Area
RMB_PLCOpenFieldBus	Parameter_FBs

Tab. 7-55: Library assignment

Interface description

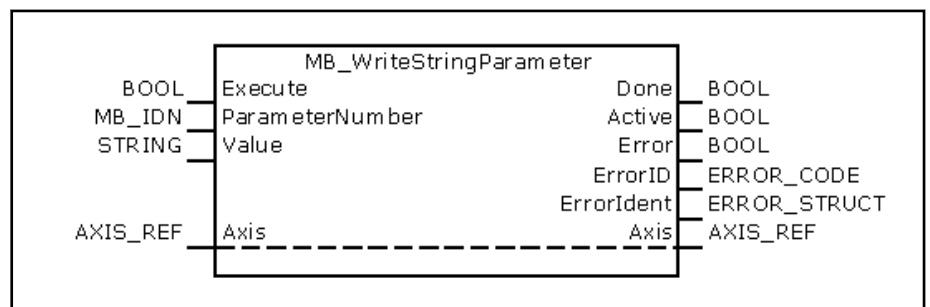


Fig. 7-61: MB_WriteStringParameter FB

	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 desired (e.g. FP_S_0_0000, also refer to Parameter Constants page 379)
	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	Data processing runs when preprocessing is complete
	Error	BOOL	It indicates that an error occurred when the function block instance was called

RMB_PLCopenFieldBus.library

	Name	Type	Comment
	ErrorID	ERROR_CODE	Brief note on error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 7-56: Interface of the MB_WriteStringParameter FB

Signal-time diagram

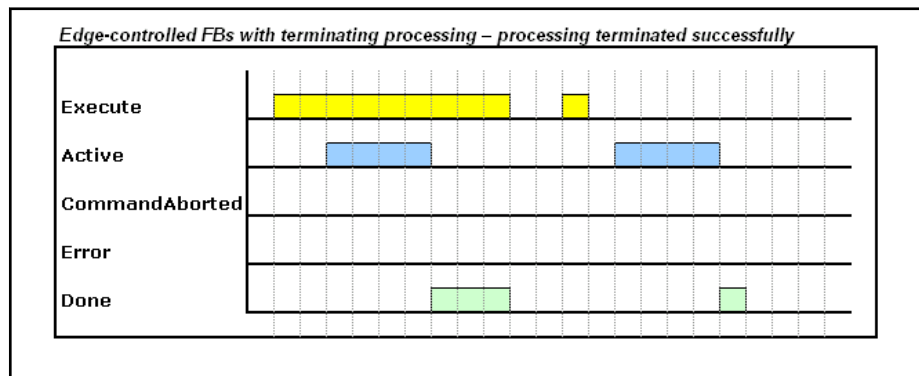


Fig. 7-62: Default signal-time diagram

7.5.33 MB_WriteStringParameter, Special Features of the Field Bus Drives



The MB_WriteStringParameter function block only runs together with the IndraDrive drives.

Error handling

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.

7.5.34 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	Area
RMB_PLCopenFieldBus	Parameter_FBs

Tab. 7-57: Library assignment

Interface description

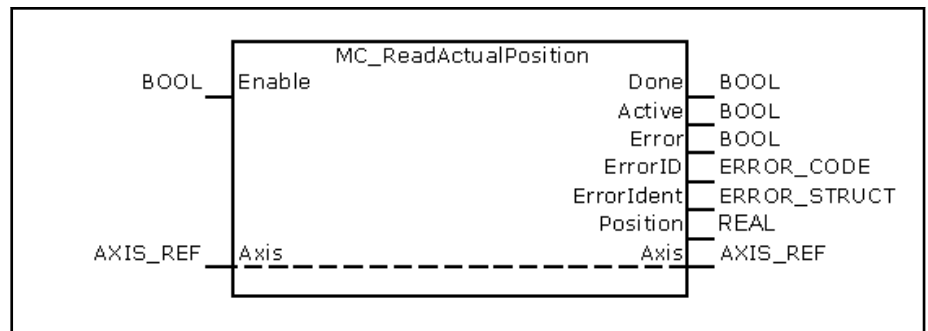


Fig. 7-63: MC_ReadActualPosition FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It 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	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
	Position	REAL	Current absolute position of the drive ⁷⁾

Tab. 7-58: Interface of the MC_ReadActualPosition FB

Signal-time diagram

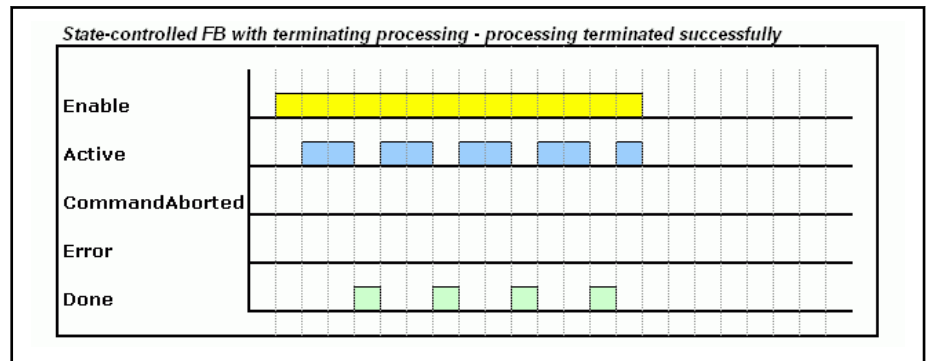


Fig. 7-64: Default signal-time diagram

7.5.35 MC_ReadActualPosition, Special Features of the Field Bus Drives



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.

⁷⁾ The measuring units of time, position, velocity, acceleration and deceleration depend on the effective scaling.

RMB_PLCopenFieldBus.library

Error handling 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 appears

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.

7.5.36 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	Area
RMB_PLCopenFieldBus	Parameter_FBs

Tab. 7-59: Library assignment

Interface description

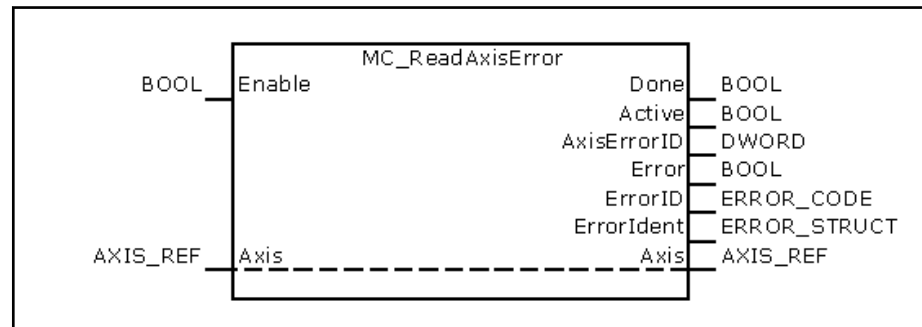


Fig. 7-65: MC_ReadAxisError FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It 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. 7-60: Interface of the MC_ReadActualPosition FB

Signal-time diagram

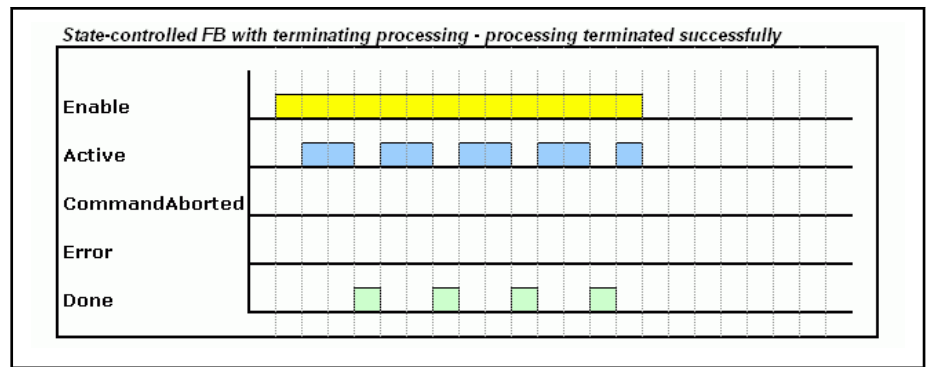


Fig. 7-66: Default signal-time diagram

7.5.37 MC_ReadAxisError, Special Features of the Field Bus Drives



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).

Error handling

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 appears

F_RELATED_TABLE

```
Error := TRUE;
ErrorID := RESOURCE_ERROR;
ErrorIdent.Table := F_RELATED_TABLE;
ErrorIdent.Additional1 := 16#000A; (*Configuration error*)
ErrorIdent.Additional2 := 0;
```

7.5.38 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	Area
RMB_PLCOpenFieldBus	Parameter_FBs

Tab. 7-61: Library assignment

RMB_PLCopenFieldBus.library

Interface description

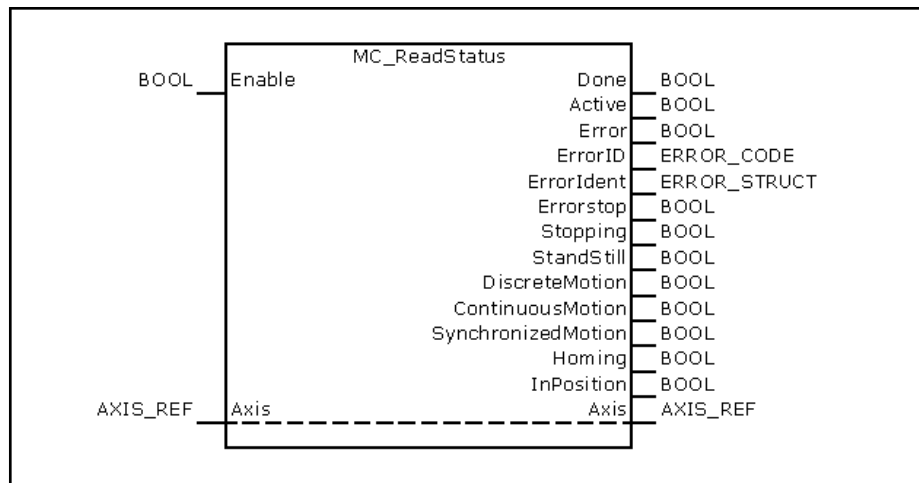


Fig. 7-67: MC_ReadStatus FB

	Name	Type	Comment
VAR_IN_OUT	Axis	AXIS_REF	It 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. 7-62: Interface of the MC_ReadStatus FB

Signal-time diagram

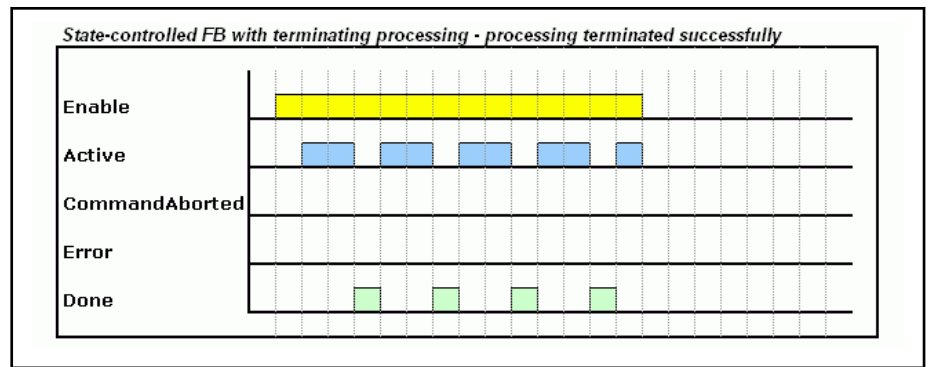


Fig. 7-68: Default signal-time diagram

7.5.39 MC_ReadStatus, Special Features of the Field Bus Drives



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.

Error handling

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 a Field Bus Connection (Field Bus Axes)

7.7.1 General Information

Drives (IndraDrive from MPx2) can be controlled by a Rexroth IndraLogic XLC or IndraMotion MLC depending on their interfaces via the respective field bus.

RMB_PLCopenFieldBus.library



The drives are connected to the control via a PROFIBUS.

The coupling via a sercos II or sercos III bus is not included in 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 PROFIBUS,

- an IndraWorks project has to be created,
- the desired PLC has to be applied to the project from the device library,
- a PROFIBUS interface (onboard or function module) has to be activated as PROFIBUS DP master,
- [Making field bus axes of the PLC known, page 415](#), and
- [communication-required drive settings, page 416](#) have to be made.

The PLC programs use MotionLogic function blocks, functions and data types summarized in several [Libraries, page 419](#).

These libraries are integrated together with the PLC program(s) in the IndraWorks project.

Additionally to the control of the field bus drive via the PLC program, it is required to exchange values with the drive saved in its parameters.

For **IndraDrive**, this can be executed via the PLCopen function blocks (MB_ReadParameter, MB_ReadRealParameter, MB_WriteParameter, MB_WriteRealParameter...) of the **RMB_PLCopenFieldBus.library**.

7.7.2 Related Documentation for IndraDrive

Title Rexroth IndraDrive ...	Kind of documentation	Document typecode ¹⁾ DOK-INDRV*-...	Part number R911...
MPx-17 Functions	Application Manual	MP*-17VRS**-APxx-EN-P	331236
MPx-16 Functions	Application Manual	MP*-16VRS**-APxx-EN-P	326767
Firmware for Drive Controllers MPH-08, MPB-08, MPD-08, MPC-08	Functional Description	MP*-08VRS**-APxx-EN-P	332643
Firmware for Drive Controllers MPH-07, MPB-07, MPD-07, MPC-07	Functional Description	MP*-07VRS**-FKxx-EN-P	328670
Firmware for Drive Controllers MPH-06, MPB-06, MPD-06, MPC-06	Functional Description	MP*-06VRS**-FKxx-EN-P	326766
Firmware for Drive Controllers MPH-05, MPB-05, MPD-05	Functional Description	MP*-05VRS**-FKxx-EN-P	320182
Firmware for Drive Controllers MPH-04, MPB-04, MPD-04	Functional Description	MP*-04VRS**-FKxx-EN-P	315485
Firmware for Drive Controllers MPH-03, MPB-03, MPD-03	Functional Description	MP*-03VRS**-FKxx-EN-P	308329
Firmware for Drive Controllers MPH-02, MPB-02, MPD-02	Functional Description	MP*-02VRS**-FKxx-EN-P	299223

Title Rexroth IndraDrive ...	Kind of documentation	Document typecode ¹⁾ DOK-INDRV*-...	Part number R911...
MPx-16 and MPx-17 Parameters	Reference Book	GEN1-PARA**-RExx-EN-P	328651
Drive Controllers MPx-02 to MPx-08	Parameter Description	GEN-**VRS**-PAXx-EN-P	297317
MPx-16 and MPx-17 Diagnostic Messages	Reference Book	GEN1-DIAG**-RExx-EN-P	326738
MPx-02 to MPx-08 and HMV	Troubleshooting Guide	GEN-**VRS**-WAXx-EN-P	297319

1) In the document typecodes, "xx" is a wild card for the current edition of the documentation (example: RE02 is the second edition of a Reference Book)

Tab. 7-63: Documentations – Firmware

7.7.3 Creating Field Bus Axes in the Project

System and drive settings are required in the IndraWorks project at the PLC in order to ensure a correct communication between the drives and the PLC.

The field bus axes are ordinary field bus participants at the PROFIBUS.

The description files for possible IndraDrives are included in the IndraWorks device library:

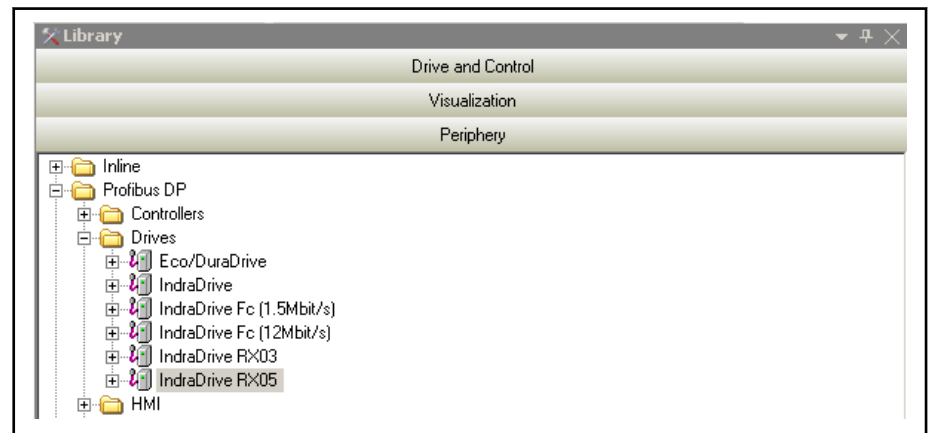


Fig. 7-69: Description files of the drive in the IndraWorks library

Drag the required drives from the library onto the PROFIBUS DP master object.

In the Project Explorer, drives can thus also be inserted between existing drives.

Optionally, slaves can be added in the context menu via **Add ► Slave ► Drives** of the PROFIBUS DP master object.

In this case, the new drive is added as last drive below the PROFIBUS DP master object.



Fig. 7-70: Entering IndraDrive as PROFIBUS participant

RMB_PLCopenFieldBus.library

The drive can be renamed:

Context menu **Rename F2**, e.g. "Drive_X" instead of "IndraDrive_RX05".

The drive can receive a new station address:

Double-clicking on the object; changing the station address for example to "11" instead of "3".

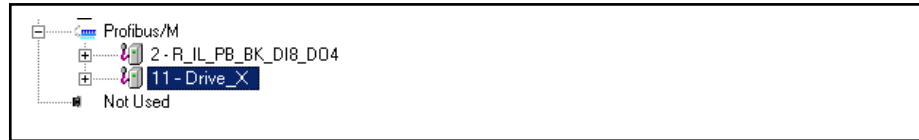


Fig. 7-71: Renaming IndraDrive as PROFIBUS participant

The PROFIBUS device is provided with several subdevices whose content is specified by the device description file.



At IndraDrive, the following settings are required:

- ParamCh_5_Words,
- F-Module_not_used, (if the module is not used)
- Input_8_Words,
- Output_10_Words.

The drives (device files) in original state are not provided with the required settings to communicate together with the PLCopen function blocks from the RMB_PLCopenFieldBus.Library via the PROFIBUS.

First, the settings given in the note have to be made by "Setting device" of the subdevices.

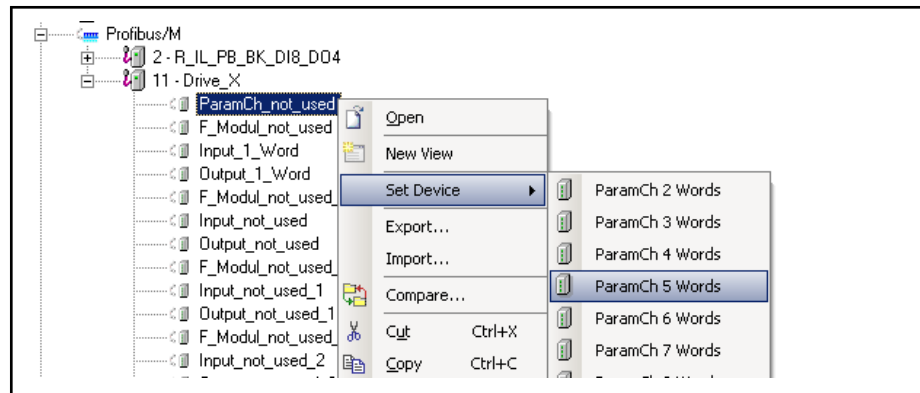


Fig. 7-72: IndraDrive, setting the subdevices

Even the drives are not configured in their original state for such a communication with a PLC via a PROFIBUS.

Therefore, drive settings have to be made for the PLC communication.

7.7.4 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 made via the "IndraWorks Engineering" desktop or via "IndraWorks DS".



Currently, it **cannot** be parameterized via the PROFIBUS connection of the PLC.

From the PC, on which the IndraWorks Engineering desktop is started, the communication has to be established via RS232 serial or via the existing PROFIBUS DP master for the parameterization.

In the following, the parameterization is described via serial communication.

Establishing the Communication for Drive Parameterization

Using the function **Project ► Search for Devices** in the IndraWorks Engineering Desktop, the parameterization communication to the drive can be established.

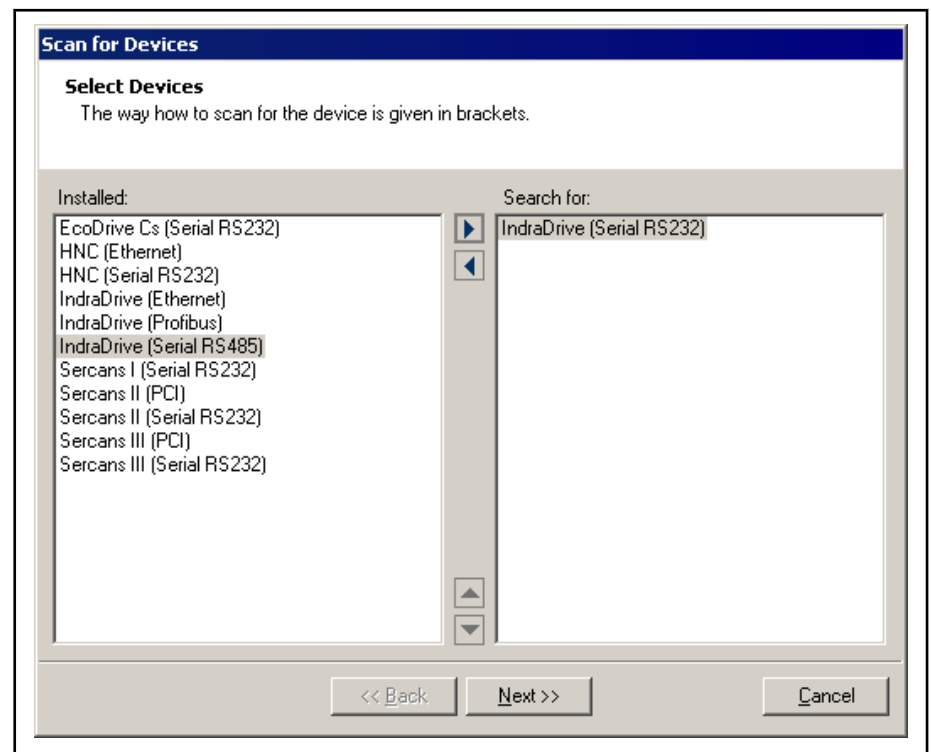


Fig. 7-73: Establishing an IndraWorks parameterization communication to the drive

If the connection to the drive has been established, the settings can be made.

RMB_PLCoenFieldBus.library

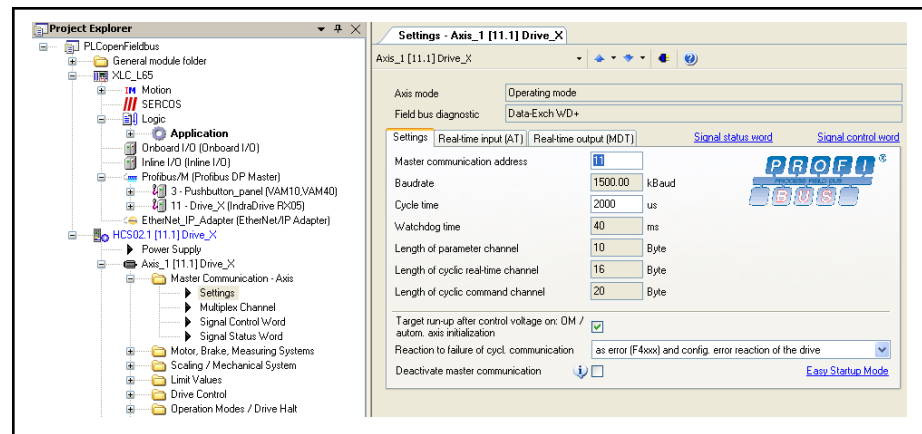


Fig. 7-74: Example: IndraWorks dialog to parameterize an IndraDrive up to MPx05

Communication Settings for IndraDrive

General information

In the following, the settings required to communicate with an IndraDrive are described.

Setting the bus device address

The bus device address of the drive has to be selected in the node "Master communication", depending on the set station address.

Length of the parameter channel

Via direct input of the parameter P-0-4083 with the parameter editor, the length of the parameter channel is set to 10 bytes.

A newer IndraDrive firmware (from MPH-03VRS) automatically sets the option 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 here:

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 main operation mode, the operating mode "Drive-controlled positioning" must be set.

The first secondary operation mode assigned must be "Velocity control".

The second and following secondary operation mode is 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.

7.7.5 Including Required Libraries

The PLCopen concept was implemented for IndraWorks Engineering for two 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_xxx_yyy.library⁸⁾, supplementing data types to the ML* system,
- supplementing libraries, e.g. RIL_Uilities.library.

PROFIBUS DP

- RMB_PLCOpenFieldBus.library, function blocks and data types of the PLCopen for PROFIBUS drives.
- RIL_CommonTypes.library, error management.
- supplementing libraries, e.g. RIL_Uilities.library.

It was tried to maintain a uniform concept for both field bus types. This also affects identical names for functionally identical function blocks and data types.

Except the RMB_PLCOpenFieldBus.library, all libraries are part of the default installation of IndraWorks Engineering. They are also initialized during the installation and **do not** require namespace specifications.

RMB_PLCOpenFieldBus.Library

The RMB_PLCOpenFieldBus.library is not initialized by default.

If it is required, it has to be postinstalled via the library manager **Add library...**

The menu item opens a selection window in which the library can be found under **IndraLogic ► Base**.

⁸⁾ xxx: MLC or XLC; yyy: L25, L45; L65, VEP

RMB_PLCopenFieldBus.library

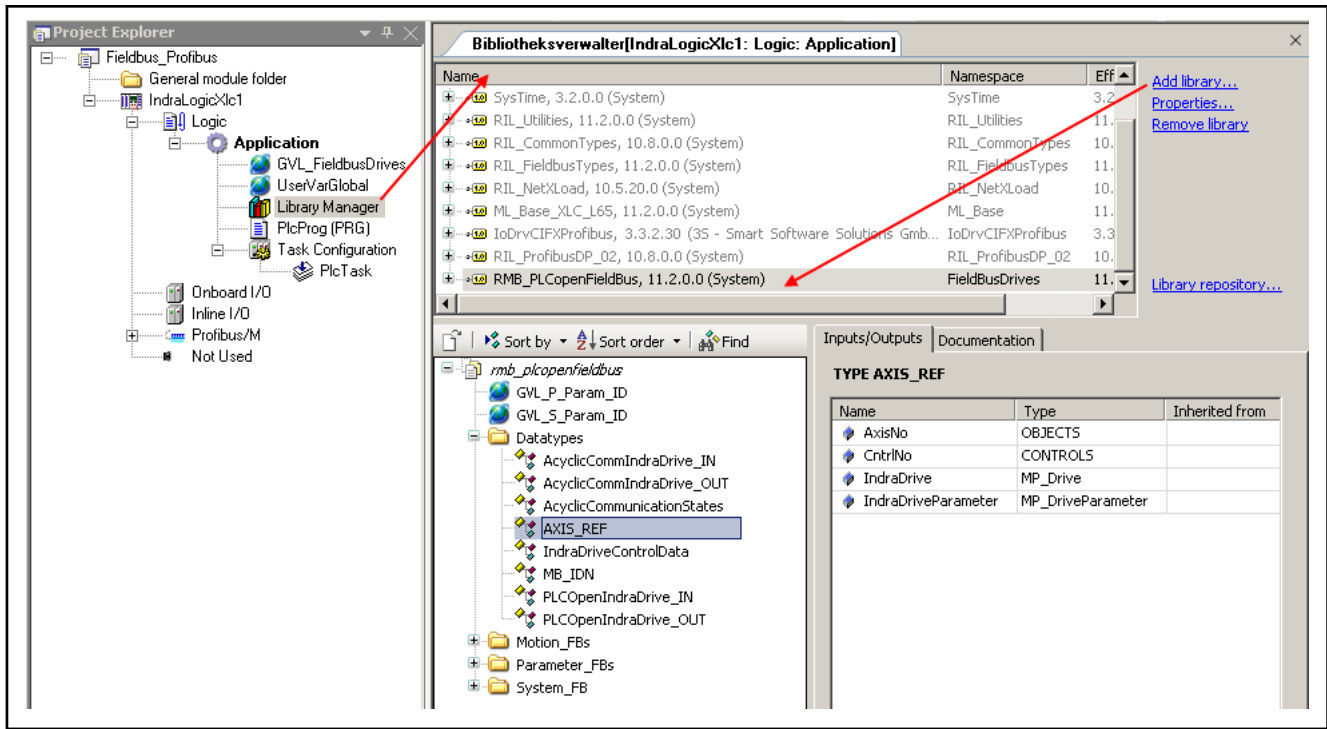


Fig. 7-75: Example: Library Manager

The RMB_PLCopenFieldBus.library can run on all IndraWorks target systems.

The elements of the RMB_PLCopenFieldBus.library are always to be used with a specification of the namespace "FieldBusDrives."

If field bus axes and servos axes should be operated together, both libraries (RMB_PLCopenFieldBus.library and ML_PLCopen.library) have to be contained in the project. With the namespace specification, the function blocks and the axis reference (AXIS_REF) are distinguished.

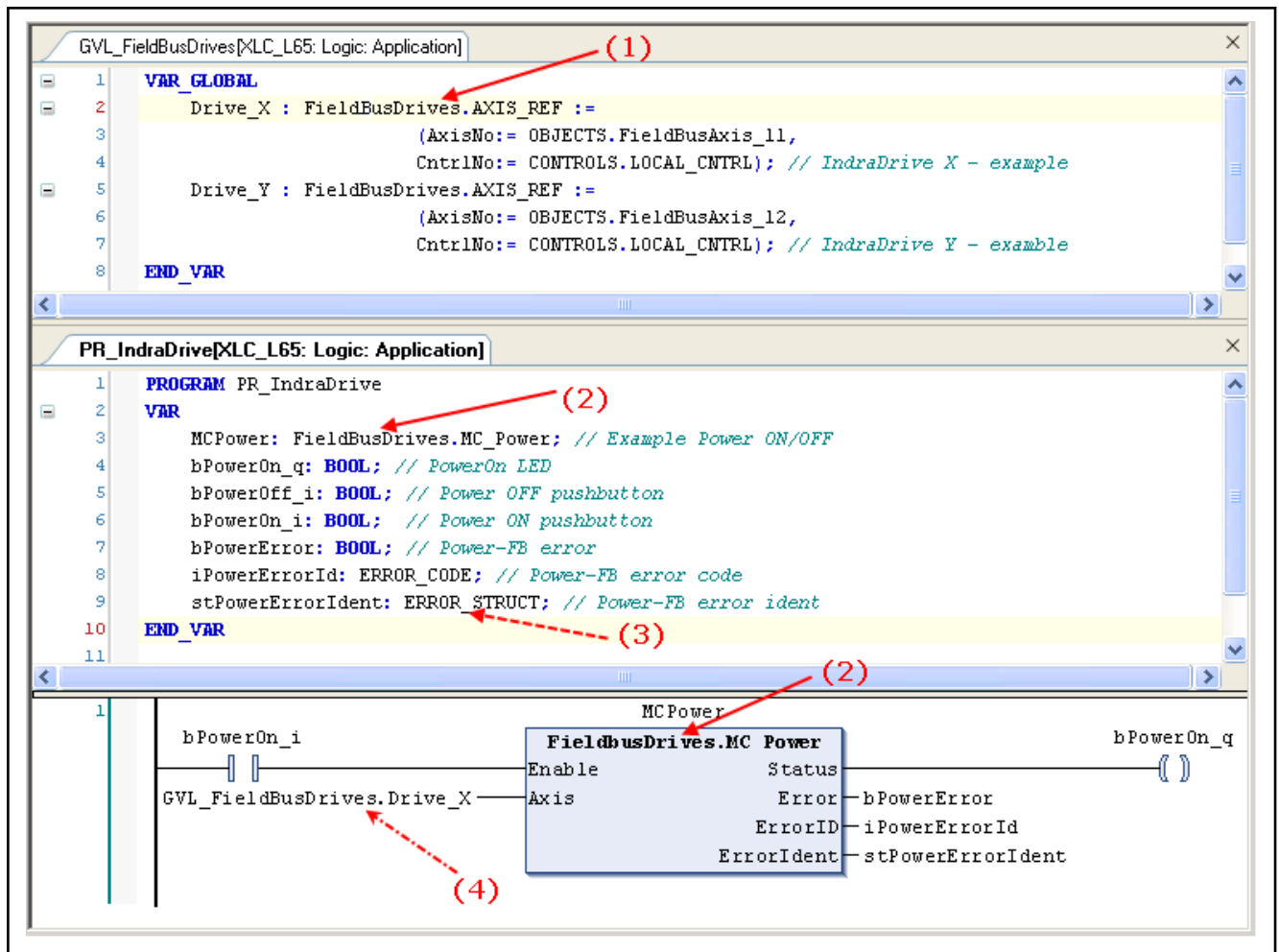
Namespace specification "Field-BusDrives"

The namespace specification "FieldBusDrives." for a field bus drive has to be in front.

The following figure shows the usage of the namespace.

The declaration of an axis reference (AXIS_REF) is always required by specifying the namespace.

The specification of the namespace during the declaration of a PLCopen function blocks is only required if the library (ML_PLCopen) is available with PLCopen function blocks in the project.



- (1) AXIS_REF originates from the RMB_PLCOpenFieldBus.library not from the RIL_CommonTypes.library
- (2) MC_Power originates from the RMB_PLCOpenFieldBus.library not from the ML_PLCOpen.library (declaration and implementation)
- (3) ERROR_STRUCT originates from the RIL_CommonTypes.library
- (4) Drive_X originates from the Global Variable List "GVL_FieldBusDrives"

Fig. 7-76: Examples for the namespace specifications

7.7.6 Addressing Field Bus Drives in the PLC Project

AXIS_REF, Axis Addresses of the 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.

RMB_PLCopenFieldBus.library

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 "CntlNo" 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

```
VAR GLOBAL
Drive_X: FieldBusDrives.AXIS_REF :=
    (CntlNo:=LOCAL_CNTRL,
     AxisNo:=FieldBusAxis_11); // IndraDrive X - example
Drive_Y: FieldBusDrives.AXIS_REF :=
    (CntlNo:=LOCAL_CNTRL,
     AxisNo:=FieldBusAxis_12); // IndraDrive Y - example
Drive_Z: FieldBusDrives.AXIS_REF :=
    (CntlNo:=LOCAL_CNTRL,
     AxisNo:=FieldBusAxis_13); // IndraDrive Z - example
Drive_n: FieldBusDrives.AXIS_REF :=
    (CntlNo:=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: CntlNo

```
(CntlNo: = LOCAL_CNTRL
```

The enumeration value LOCAL_CNTRL is in the RIL_CommonTypes.lib, folder "Datatypes 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 "Datatypes 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 reasonable to compare the PROFIBUS station number and the axis number.

AxisData[], Axis Data

The AxisData array, known from the sercos axes, is not supported for field bus axes.

7.7.7 I/O Addressing of the Drives in the PLC Project

General information

On the PROFIBUS, the field bus drives appear as ordinary I/O modules. An I/O address has to be assigned.

In order 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 IndraDrive are always individual submodules of the participant.

The communication "Drive to the PLC" consists of the following:

- Subdevice (ParamCh_5_Words), 5 words for acyclic 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.

ParamCh_5_Words

Variable	Mapping	Channel	Address	Type
Application.GVL_FieldBusDrives.Drive_X.IndraDriveParameters.ParamCh_5_Words_In		Input0	%I1W2	
Application.GVL_FieldBusDrives.Drive_X.IndraDriveParameters.ParamCh_5_Words_Out		Output0	%Q1W2	

Input_8_Words

Variable	Mapping	Channel	Address	Type	Current Value
Application.GVL_FieldBusDrives.Drive_X.IndraDrive._Input_8_Words		Input0	%I1W12		

Output_10_Words

Variable	Mapping	Channel	Address	Type	Current Value
Application.GVL_FieldBusDrives.Drive_X.IndraDrive._Output_10_Words		Output0	%Q1W12		

Fig. 7-77: Example: I/O mapping for an IndraDrive field bus drive

RMB_PLCOpenFieldBus.library

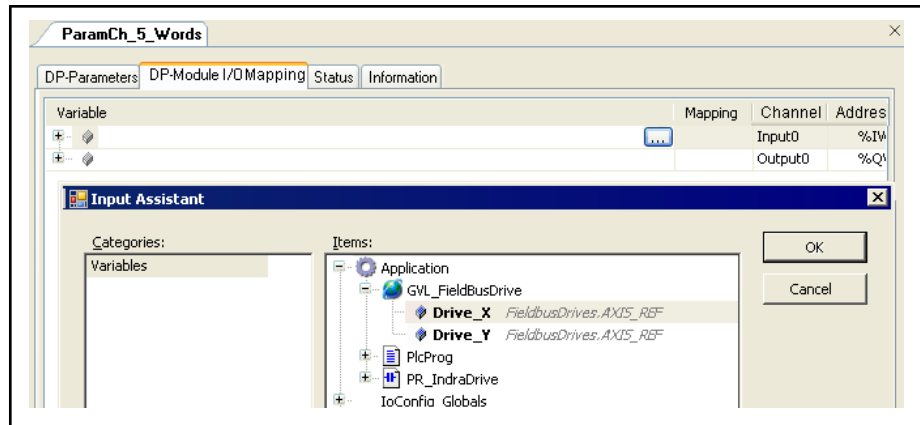


Fig. 7-78: Example: Input help for the mapping to select the drive

In AXIS_REF, the field bus drives are distinguished for the cyclic parameter exchange and "IndraDriveParameter" for the acyclic parameter exchange according to "IndraDrive"

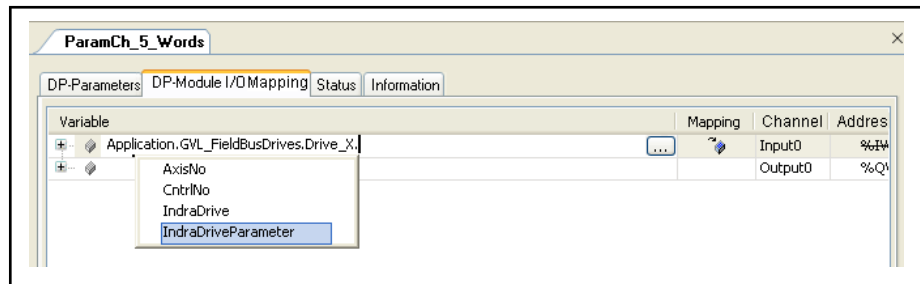


Fig. 7-79: Example: Mapping for the parameter exchange

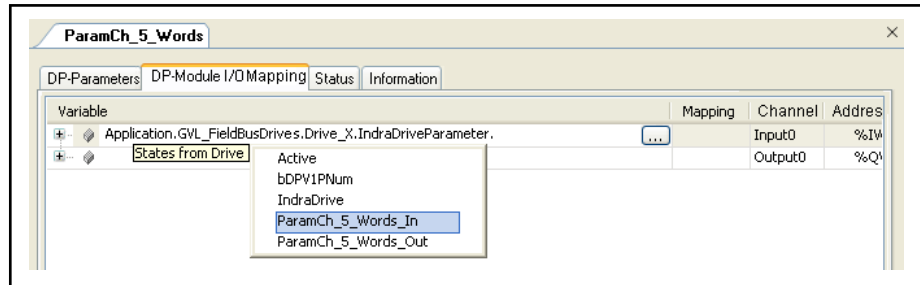


Fig. 7-80: Example: Example for the inputs of the parameter exchange

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 ML_ControllerBase.library

8.1 Introduction

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 separated in two parts.

- [Basic class of the controller, page 425](#), Functionality of the controller,
- [Controller Management, page 428](#), controller behavior.

8.2 Basic class of the controller

8.2.1 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
IActualValue	Actual value detection
IModuleControl	Controller initialization Phase switching
IOperationModeHandler	For operation modes of a controller, see: Controller management, page 428
IOutputData	Command value generation of a controller

Tab. 8-1: 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

ML_ControllerBase.library

getIOutputData	IOutputData
getIRetainHandler	IRetainHandler
getIDigitalIO	IDigitalIO
getIAdditionalSetValue	IAdditionalSetValue

Tab. 8-2: 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^;

```

8.2.2 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
lrActualPos	LREAL	Actual position of the controller
lrActualVel	LREAL	Actual velocity of the controller
lrActualAcceleration	LREAL	Actual acceleration of the controller
lrActualForce	LREAL	Actual force of the controller
lrTargetPos	LREAL	Cyclic position command value of the controller
lrTargetVel	LREAL	Cyclic velocity command value of the controller
lrTargetAcceleration	LREAL	Cyclic acceleration command value of the controller
lrTargetForce	LREAL	Cyclic force command value of the controller
fbConverterCAxis	ML_Convert	Conversion functions to convert between internal and external units (4 bytes)

fbConverterCAxis_DoublePrecision	ML_Convert_DoublePrecision	Conversion functions to convert between internal and external units (8 bytes)
IrCycleTime	LREAL	Cycle time of the Motion kernel in μ s. New command values are generated during this period.

Tab. 8-3: 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. 8-4: Data units

8.2.3 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. 8-5: 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.

ML_ControllerBase.library

8.2.4 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).

8.2.5 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`

8.2.6 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.

8.2.7 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.

8.3 Controller management

8.3.1 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 `aritifModuleInterface` 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 := (aritmModuleInterface := myController);
END_VAR

myAdministration(Axis := ControlAxis);
```

8.3.2 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

The [State machine of the operating state, page 429](#), is run through.

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, Torque/actual force value
wCStatusWord	A-0-0019, Controller status word

Tab. 8-6: Synchronized status information

8.3.3 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 430

Tab. 8-7: State machine: operating state

ML_ControllerBase.library

8.3.4 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 430](#).
- Control value output** It is executed via the `runOutputData` method of the `IOutputData` interface of the controller.

8.3.5 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.

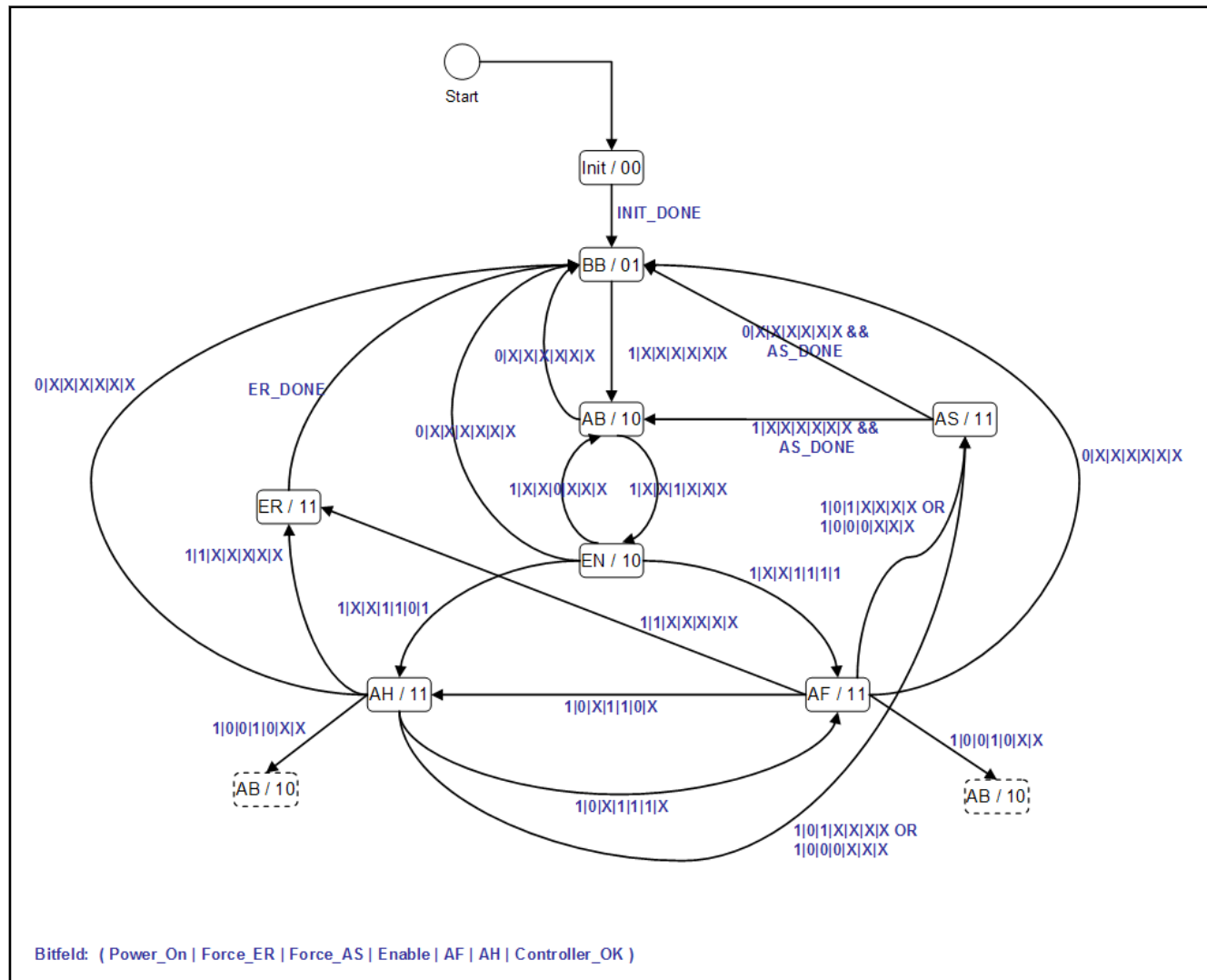


Fig. 8-1: State machines of the operation modes

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 x 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 connect 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. 8-8: 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. 8-9: Operation mode call in the state...

ML_ControllerBase.library

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 operation mode: The control loop is closed. The controller is in a cyclic velocity control.
K_OPCARD_SCND_2	2. Secondary operation mode: The control loop is closed. The controller is in cyclic force control.
K_OPCARD_SCND_3	3. Secondary operation mode: The control loop is closed. The controller is in a cyclic analog value control.

Tab. 8-10: Operation mode of a controller

8.3.6 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 [ML_OORV_CONTROLLERDATA](#), page 426).

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> .

8.3.7 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 for the handling in the data structure of the controller `ML_OORV_CONTROLLERDATA` (see [ML_OORV_CONTROLLERDATA](#), page 426).

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
Axxxxxxx	Diagnostics is a message
Exxxxxxx	Diagnostics is a warning
Fxxxxxxx	Diagnostics is an error

Tab. 8-11: Coding the diagnostics

The system prioritizes 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.

9 ML_IPOImpactControl.library

9.1 Introduction and Overview

Goal To implement the replacing closed-loop 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 be able to adjust a behavior desired by the PLC controller.

Technical concept Normally, a new command value is provided when a motion command is executed by the interpolator. This command value specification is carried out in each Motion cycle. The concept for manipulating this task intends to put a call in an additional state machine in front of the actual 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.

For explanatory purposes, refer to the following simplified example:

To keep the command value generation on an existing value, the state machine in front can request the interpolator not to carry out further calculations. Then, 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

Interference	Comment
Direction-dependent stopping of the interpolation	This interference is also called "Break".
Direction-dependent jumping in the interpolation end value.	This interference is also called "Jump".
Setting individual position and velocity values in the "Break" state or at the end of a motion command.	This functionality is required to write value 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,
- MC_MoveAdditive,
- MC_MoveRelative and
- MC_MoveVelocity.

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.

ML_IPOImpactControl.library

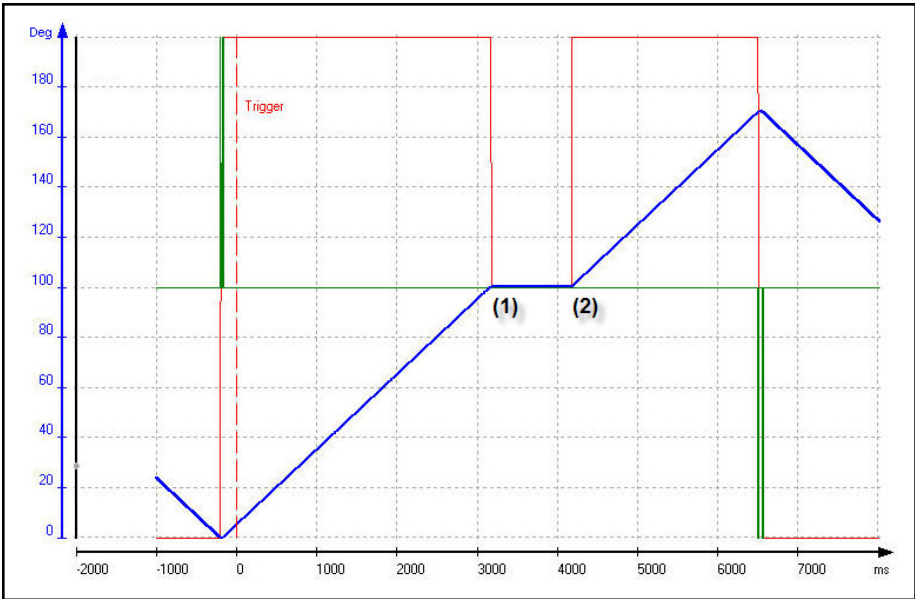


Fig. 9-1: blue = interpolated position, red = interpolated velocity, green = interpolated acceleration

9.2 ML_IPOImpactControl

Brief Description The ML_IPOImpactControl function block and its methods are used to manipulate the interpolation.

The function block must be called cyclically since it additionally monitors the error state of the axis. If an axis reports an error, all manipulations set are automatically reset.

Library	Range
ML_IPOImpactControl	Motion

Tab. 9-1: Library assignment

Interface

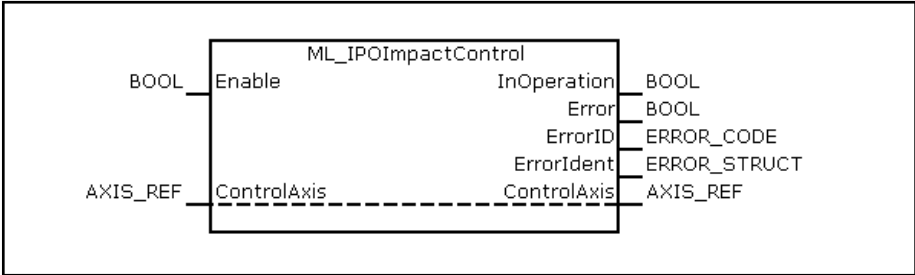


Fig. 9-2: ML_IPOImpactControl function block

	Name	Type	Comment
VAR_IN_OUT	ControlAxis	AXIS_REF	Contains information on the concrete axis.
VAR_INPUT	Enable	BOOL	As long as "Enable" is TRUE, the interpolation can be influenced
VAR_OUTPUT	InOperation	BOOL	Function block is enabled, interpolation can be influenced
	Error	BOOL	Indicates that an error occurred in the function block instance
	ErrorID	ERROR_CODE	Brief note on the error cause
	ErrorIdent	ERROR_STRUCT	Detailed error information

Tab. 9-2: Interface of the ML_IPOImpactControl function block

9.3 ML_IPOImpactControl Methods

9.3.1 mSetNewState

Brief Description This method sets a new state for influencing 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.

9.3.2 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.

9.3.3 mSetDirection

Brief Description This method sets the direction in which the interpolation should be influenced.

Signature

```
METHOD mSetDirection : DINT
VAR_INPUT
    eDirection: eIPOCONTROL_DIRECTION;
END_VAR
```

ML_IPOImpactControl.library

Possible values for eDirection:

Value	Comment
IPOCONTROL_DIRECTION_NONE	Interpolation is not influenced in any direction.
IPOCONTROL_DIRECTION_FORWARD	Interpolation should be influenced in forward direction.
IPOCONTROL_DIRECTION_BACKWARD	Interpolation should be influenced in backward direction.
IPOCONTROL_DIRECTION_BACKWARD OR IPOCONTROL_DIRECTION_FORWARD	Interpolation should be influenced in both directions.

9.3.4 mGetDirection

Brief Description This method provides the direction in which the interpolation should be influenced.

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 influenced in any direction.
IPOCONTROL_DIRECTION_FORWARD	Interpolation should be influenced in forward direction.
IPOCONTROL_DIRECTION_BACKWARD	Interpolation should be influenced in backward direction.
IPOCONTROL_DIRECTION_BACKWARD OR IPOCONTROL_DIRECTION_FORWARD	Interpolation should be influenced in both directions.

9.3.5 mSetAcceleration

Brief Description This method sets the individual acceleration that should be considered by the interpolation at the state transition

"neutral" ↔ "break".

Signature

```

METHOD mSetAcceleration : DINT
VAR_INPUT
    rAcceleration: REAL ;
END_VAR

```

9.3.6 mGetAcceleration

Brief Description This method provides the individual acceleration that should be considered by the interpolation at the state transition.

Signature

```

METHOD mGetAcceleration : DINT
VAR_INPUT

```

```
prAcceleration: POINTER TO REAL;  
END_VAR
```

9.3.7 mSetBreakBehaviour

Brief Description This method sets the Break behavior that should 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 performed directly.
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.

9.3.8 mGetBreakBehaviour

Brief Description This method provides the Break behavior that was 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 performed directly.
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.

9.3.9 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
```

ML_IPOImpactControl.library

9.3.10 mGetPosition

Brief Description This method provides the new position to be considered by the interpolation.

Signature

```
METHOD mSetPosition : DINT
VAR_INPUT
    lrPosition: LREAL;
END_VAR
```

9.3.11 mSetValidValues

Brief Description This method makes the settings regarding the values of velocity and position 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.

9.3.12 mGetValidValues

Brief Description This method provides the settings regarding the values of velocity and position to be currently considered by the interpolation.

Signature

```
METHOD mGetValidValues : DINT
VAR_INPUT
    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.

9.3.13 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
```

9.3.14 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
```

9.3.15 mGetAxisControlledBy

Brief Description This method provides the information whether the axis interpolation is influenced by the PLC program.

Signature

```
METHOD mGetAxisControlledBy : DINT
VAR_INPUT
    peAxisControlledBy: POINTER TO eIPOCONTROL_CONTROLLEDBy;
END_VAR
```

Possible values for peAxisControlledBy:

Value	Comment
IPOCONTROL_CONTROLLEDBy_DEFAULT	The interpolation is not influenced.
IPOCONTROL_CONTROLLEDBy_PLC	The interpolation is influenced by the PLC program.

9.4 Applying the Methods

9.4.1 Introduction

Calling the methods is shown in the following two examples - States "Break" and "Jump".

9.4.2 Example 1: Activate "Break" State

Explanation In "Break" state, the interpolation is paused. That means that 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".

ML_IPOImpactControl.library

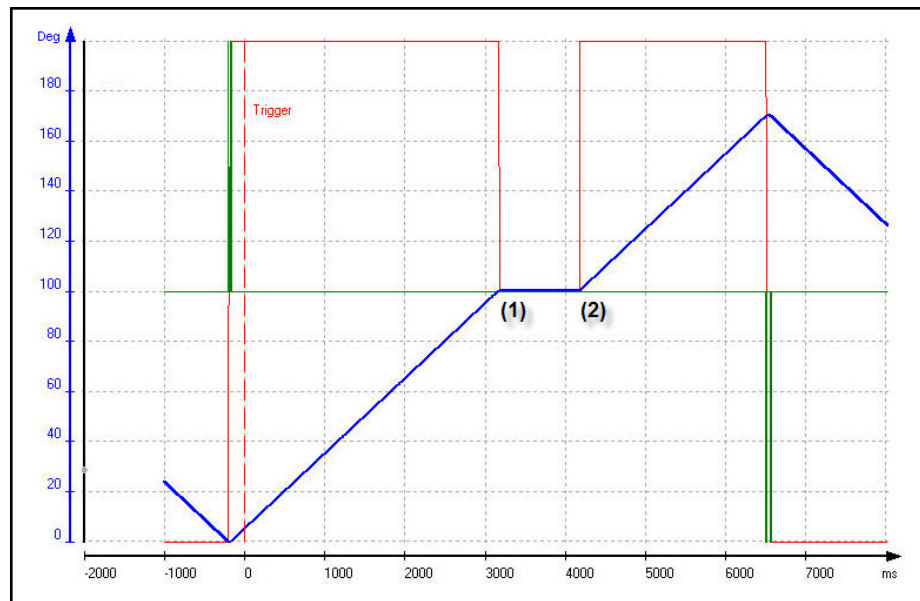


Fig. 9-3: blue = interpolated position, red = interpolated velocity, green = interpolated acceleration

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);
```

9.4.3 Example 2: Activate "Jump" State

Explanation 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".

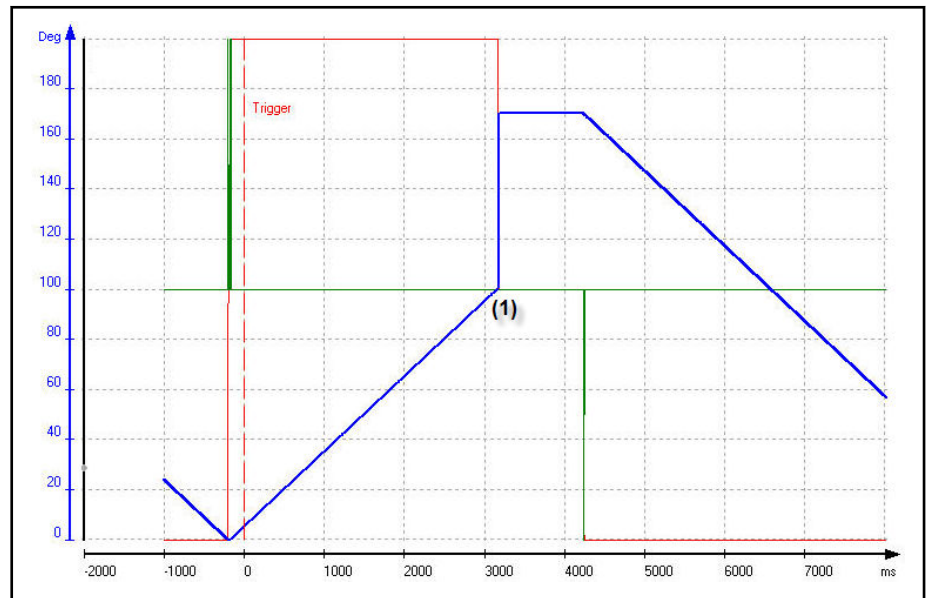


Fig. 9-4: blue = interpolated position, red = interpolated velocity, green = interpolated acceleration

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 invalid 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
MyIPOImpactControl.mSetVelocity(rVelocity:= 1.0);
//Set a new position
MyIPOImpactControl.mSetPosition(lrPosition:= 180.0);
//Activates the values
```

ML_IPOImpactControl.library

```
MyIPOImpactControl.mSetValidValues(eValidValues:=IPOCONTROL_RESPECT_POSITION  
OR IPOCONTROL_RESPECT_VELOCITY);
```

10 ML_ParamBase.library

10.1 Usage

Using this library, the controller axis can be equipped with extended parameterization function, 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.

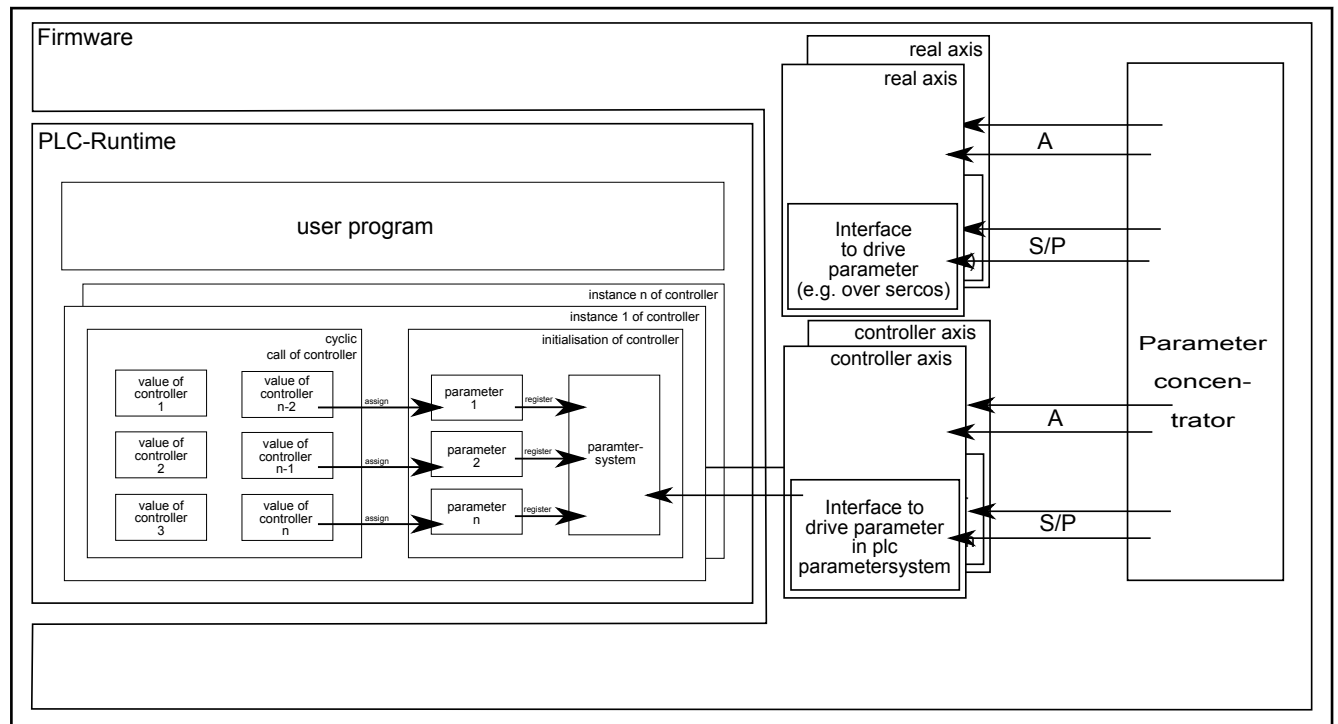


Fig. 10-1: Using a controller axis and a PLC

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 contained 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 [Parameter system, page 446](#) and two different parameter definitions:

- [PARAM_SercosParameter, page 450](#),
- [PARAM_ExtSercosParameter, page 455](#).

A function block to process command sequences is added to the library ([PARAM_CmdExecuter, page 457](#)).

ML_ParamBase.library

10.2 Function blocks

10.2.1 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, an axis reference has to be specified during the initialization.

Signature

```
METHOD Init
VAR_INPUT
    Axis : AXIS_REF;
END_VAR
```

Registering Parameters

Brief description The system can only know which parameters are assigned to the controller axis if these are registered in the parameter system.

Therefore, the **addParameter** method has to be called with its address. Each parameter to be used has to be registered that way.

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 450](#)).

The respective parameter number is specified in the description of the function. 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 specified functions are to be seen as low level functions.

Language Switching

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 459](#)) can be used.

More information can be found in the section "Data Types" ([Data Types](#), [page 459](#)).

The language can be switched in the PLC via the Property Language. Additionally, the parameter system reacts to the EIDN S-0-0265.

Reading Parameters

Brief description

In the PLC, parameters can be read via the method **ParamRead**. 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
```

CAUTION

Overwriting memory

Reserve sufficient memory. Otherwise, memory can be overwritten by accident.

Reading Parameters

Brief description

As for the **ParamRead** function, **ParamRead** can be used to write on parameter data.

Depending on the attribute, the values cannot be written at any point in time. This way, 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
```

Saving 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 function.

If an empty string is transferred, a file name in the form "ControllerCF<Axis-Number>.bin" is created.

The following elements are backed up:

- EIDN
- High word of the attribute
- Data

ML_ParamBase.library

- CRC16 check sum

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

Using the command **S-0-0262**, the default values are restored. The configuration is executed 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 454](#)).



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 Saved 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.

ML_ParamBase.library

Synchronize 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 function. This function has to be called again, e.g. after a switching of the scaling.

The transfer parameter **iType** determines which parameters have 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 function sets the parameter property cyclic command values to get. The parameter is identified via the parameter number.

For more information, please see [chapter "Administration" on page 454](#).

The **eiEidn** input identifies the parameter.

Signature

```
METHOD dwSetCyclicConfigured : DWORD
VAR_INPUT
    eiEidn : DWORD;
    boValue : BOOL;
END_VAR
```

10.2.2 PARAM_SercosParameter**General information**

This class is the basic implementation of a parameter. The use of the basic implementation 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 function can be executed.

In case of a command parameter, only the command interface is accessed.

The two remaining interfaces are used for normal parameter usage for read-only and writing call.

The respective pointer of the parameter is also transferred so that the function is provided 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 **dwWriteData** 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, ...)

Signature dwCmdInit

```

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** function.

Signature dwCmdCyclic

```

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...)

Signature dwCmdAbort

```

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

```

ML_ParamBase.library

Attributes

Overview The PARAM_SercosParameter function block is provided with the following public attributes.

Class variables: PARAM_SercosParameter

Variable name	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 value references to a global array placing the pointers on the translated names ([Language, page 459](#)). 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, it is referenced on it via a pointer. 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 data 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
    data : POINTER TO PARAM_uData; // Pointer to data to write (dwData)
END_VAR
```

Read IDN

Brief description The IDN of a parameter can be read via the following method.

Signature

```
METHOD dwGetIDN : SERCOS_ERROR
VAR_INPUT
    data : PARAM_uData; //Object contains idn (PARAM_uData.eiData)
END_VAR
```

Read 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
```

Read 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
```

Read 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>)
    puData : POINTER TO PARAM_uData;
    // pointer to data to set written size
    puiLength : POINTER TO UINT;
END_VAR
```

Read 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
```

ML_ParamBase.library

Read Date**Brief description**

To read the date, the outer interface 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
```

Write Date**Brief description**

The following method is provided to write a date.

This function 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 have extreme values, an extreme value check cannot mandatorily be executed.

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 function mentioned below. This method only operates if it is 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 by means of the following function. If **boValue** is TRUE, the parameter is marked as cyclically configured. The error code K_STS_DATA_CONFCYCLIC ("[SERCOS_ERROR](#)" on [page 461](#)) is returned when writing the parameter in phase 4.

Signature

```
METHOD dwSetCyclicConfigured : DWORD
VAR_INPUT
    boValue : BOOL := FALSE;
END_VAR
```

10.2.3 PARAM_ExtSercosParameter

General information

The function block PARAM_ExtSercosParameter inherits the properties of [PARAM_SercosParameter](#), [page 450](#). Thus, only the differences and extensions are shown.

Attributes

Overview The function block PARAM_ExtSercosParameter is provided with the following inputs in addition to the function block PARAM_SercosParameter.

Class variables of the PARAM_ExtSercosParameter

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_plrScaling-Factor	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.

A validity check is only carried out 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	Description
PARAM_ESP_CHECK-EXTREM	Extreme values should be checked
PARAM_ESP_CHECK-COMB	Combination check should be performed

ML_ParamBase.library

PAR-AM_ESP_VAR_MIN	The parameter is provided with a minimum value whose value corresponds to the date of another parameter.
PAR-AM_ESP_VAR_MAX	The parameter is provided with a maximum value whose value corresponds to the data of another parameter.
PARAM_ESP_MEMO-RY_CF	The parameter should be saved on the CF card.
PARAM_ESP_WRITE-PROTECTION_P2	Parameter can be written in P2
PARAM_ESP_WRITE-PROTECTION_P4	Parameter can be written in P4

c_upParamData, c_upPropDataAs for attribute **c_upActData**, only the pointer is saved.

The value IL_NULL can be assigned to both the pointers (e.g. as for 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, this 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 function 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](#), page 455.

Signature

```

METHOD iGetParamType : INT
VAR_INPUT
END_VAR

```

10.2.4 PARAM_CmdExecuter**Brief description**

This function block realizes the processing of command sequences. A state machine is used.

The commands are processed subsequently, depending on their state. The respective method of **I_PARAM_CmdAccess** is called.

The return value of the last called method and the user inputs control the state machine during the command processing (fig. 10-2 "State machine of the command handler" on page 459).

If a command is in the "K_STATE_NOT_ACTIVE" state it is deleted from the memory.

Methods**Adding command**

The **dwAddCmd** function is used to add a command.

This function 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

```

ML_ParamBase.library

```

    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. 10-1: Return values

Clearing command memory

This method clears the internal memory. The method is automatically executed by the function block during the initializing process.

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.

10.3 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. For more information, see [PARAM_CmdExecuter](#), page 457.

The concept intends that each function block used for command processing has to implement the **I_PARAM_CmdAccess** interface. The individual methods 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.

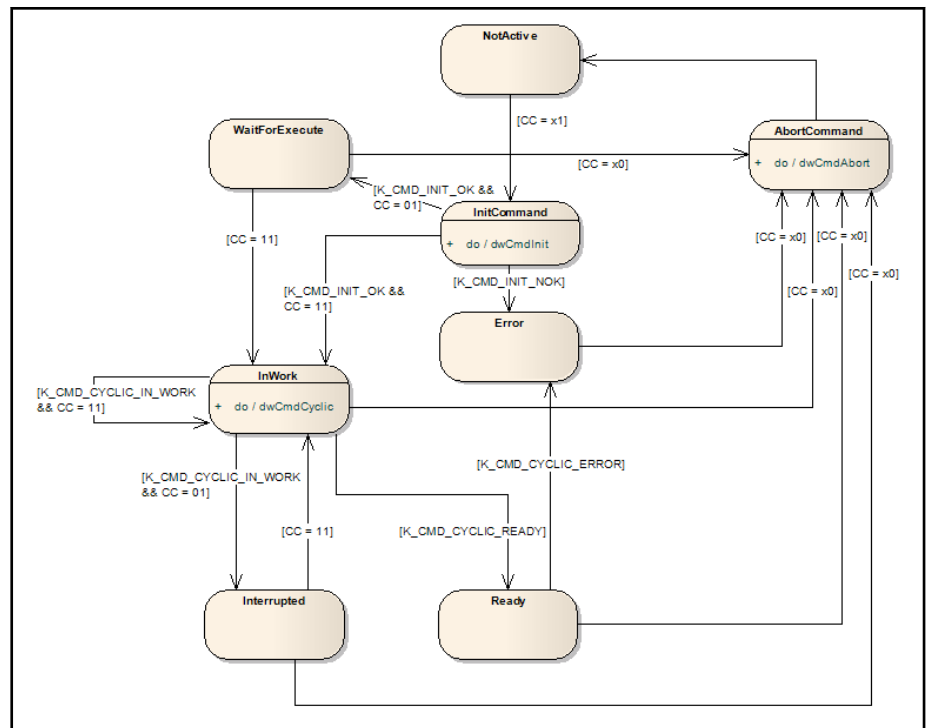


Fig. 10-2: State machine of the command handler

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.

10.4 Data Types

10.4.1 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.

ML_ParamBase.library

10.4.2 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, this variable definition can be used.

According to the sercos standard, the data contains a header with the current and maximum length. Both values are identical in 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
```

The required data type can now be assigned to the respective pointer.

PARAM_uArray 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 the data types of the 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_TYPE
```



When declaring a variable of this type, approximately 65,532 bytes of memory are reserved.

10.5 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.

10.6 Constants

usiSIZE_OF_SERCOSHEADER

Constant for the size of the sercos-specific header ([Data Access, page 460](#)).

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
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

ML_ParamBase.library

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

ML_ParamBase.library

K_STS_DATA_CONFCYCLIC	16#700A	Parameter is cyclically configured
K_STS_DATA_INVALIDINDEX	16#700B	Index is invalid
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_CMDNOTEXEC-TIME	16#7012	Command can currently not be executed
K_STS_DATA_CMDNOTEXEC-PARAM	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

Tab. 10-2: SERCOS_ERROR

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

Tab. 10-3: SERCOS_CMD_STATES

SERCOS_CMD_RETURN

Return value	Description
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

Tab. 10-4: SERCOS_CMD_RETURN

11 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.....	13
Validity of the documentation.....	13
Access to data of a field bus drive.....	353
Access to data of an axis	
Profibus.....	379
Access to data of an axis, control or another object.....	133
Access to data of an axis, control, kinematics (axis group) or another object.....	275
Access to oscilloscope data via function variable access.....	116
Active.....	24
ANSI Z535.6-2006.....	13
APP_INFO.....	163
Axis data via function variable access.....	95
Axis data via shared memory and function variable access, general.....	91
Axis data via synchronous function variable access in C2C link.....	98
Axis types and function blocks supported by these axis types.....	338
AXIS_GROUP_REF.....	41
AXIS_REF.....	39, 355
Application of field bus drive.....	421
AxisData[] - Axis data in shared memory.....	93
AxisData[],	
Application of the field bus drive.....	423

B

Basic class	
ML_ControllerBase.....	425

C

Command handling.....	458
CommandAborted.....	24
Control data in shared memory.....	101
Control Data via Function Variable Access.....	103
Control data via shared memory and function variable access.....	100
Controller data.....	426
Controller management.....	428
Connect controller.....	428
Cyclic procedure of the controller.....	430
Cyclic procedure of the controller management.....	429
Operation mode handling.....	430
Predefined generators of the operation modes.....	432
State machine: operating state.....	429
CONTROLS.....	39

D

Data types for fault diagnosis with IndraMotion blocks.....	71

Data types for referencing objects of IndraMotion blocks.....	38
Data types for synchronization in the position-controlled synchronous operation modes....	67
Data types for the indication of the date and absolute time designations for IndraMotion blocks.....	73
Data types for the support of cams, Motion-Profiles and FlexProfiles.....	43
Data types for the support of touch probe functions.....	75
Data types to support MotionProfile and FlexProfile functions.....	84
Diagnostics of MB_ReadxxxParameter / MB_WritexxxParameter / direct access.....	349
Diagnostics on the kinematic state.....	351
Done.....	24
DV_Axis[].....	95
DV_Control[].....	103
DV_Osci[].....	116
DV_Plis[].....	108
DV_Probe[].....	113
dwGetEidnP.....	461
dwGetEidnP_SESI.....	461
dwGetEidnS.....	461
dwGetEidnS_SESI.....	461

E

E_LOG_CLASS.....	128
E_LOG_FACILITY.....	129
E_LOG_SEVERITY.....	129
E_SCALING_TYPE.....	163
Enable.....	24
Error tables	
MLC_TABLE.....	343
ERROR_CODE.....	71
ERROR_STRUCT.....	72
ERROR_TABLE.....	72
Error_Tables.....	340
MLC_TABLE.....	343
SERCOS_TABLE.....	340
Execute.....	24
ExecuteLock.....	24
EXT_SYS_TIME_DATE.....	74

F

Field bus drive	
Acyclic parameter communication.....	423
AXIS_REF.....	421
AxisData[].....	423
Bus device address.....	418
Communication settings for IndraDrive.....	418
Communication-required settings.....	416
Create in the project.....	415
Cyclic data exchange.....	423

Index

Drive in original state.....	416	H	
I/O addressing.....	423	Hazard warnings.....	13
Include libraries.....	419	Helpdesk.....	465
Length of the parameter channel.....	418	HNC100-3X.....	252
Operation mode settings.....	418	Hotline.....	465
PLC program, required declaration.....	421		
Process data and profile type.....	418	I	
Reference drive.....	419	I_PARAM_CmdAccess	
Signal control word.....	418	dwCmdAbort.....	451
Signal status word.....	418	dwCmdCyclic.....	451
FlexProfile		dwCmdInit.....	451
Event control.....	237	I_PARAM_ReadAccess	
Processing types.....	233	dwReadData.....	450
Select coordinate system.....	230	dwWriteData.....	451
Switch.....	230	I/O address assignment of the field bus drives.	423
Synchronize.....	230	IActualValue.....	425
Function block prefixes.....	353	getIActualValue.....	425
Function blocks for controlled motion.....	262	IAdditionalSetValue	
Function blocks for multi-axis control.....	198	getIAdditionalSetValue.....	426
Function blocks for parameter access		IDigitalIO	
Profibus.....	377	getIDigitalIO.....	426
Function blocks for single-axis control.....	176	IL_NULL.....	38
Function blocks for the axis group control.....	248	IL_SercosAttribute.....	138
Function blocks for the parameter access		IL_SercosElementsToParaNo.....	141
sercos.....	275	IL_SercosStringToParaNo.....	142
Function blocks for the use of the cyclic data		IL_SetDateTime.....	152
channel.....	250	IModuleControl.....	425
Function blocks for the use of the touch		getIModuleControl.....	425
probe function.....	320	IndraDrive	
Function variable access		Bus device address.....	418
DV_Axis[].	95	Length of the parameter channel.....	418
DV_Osci[].	116	Setting process data and profile type.....	418
DV_Pls[].	108	Signal control word.....	418
DV_Probe[].	113	Signal status word.....	418
Function Variable Access		IndraWorks DS	
DV_Control[].	103	Settings of field bus drives.....	417
Function variable access in C2C link		IndraWorks Engineering desktop	
GV_LinkAxis[].	98	Settings of field bus drives.....	417
G		INFO_APPLICATIONS.....	163
General information on controlled motion.....	262	InGear.....	24
General information on using the cyclic data		InOperation.....	24
channel.....	250	Input description of the field bus process data.....	409, 413
GV_LinkAxis[].	98	InSync.....	24
GVL_A_GlobalVarIdx.....	132	IntelliSense	
GVL_A_Param_ID.....	135	Axis data.....	93, 95
GVL_Base.....	130	Control data.....	101
GVL_BaseConstant.....	131	Kinematic Data v.....	103
GVL_C_Param_ID.....	135	Oscilloscope data.....	116
GVL_FieldBusDrives.....	356, 422	PLS data.....	106, 108
GVL_M_Param_ID.....	136	Touch probe data.....	111, 113
GVL_N_Param_ID.....	136	IntelliSense of a function variable synchro-	
GVL_O_Param_ID.....	137	nously in the C2C link.....	98
GVL_P_Param_ID.....	137, 354	Interface IAcutalValue.....	428
GVL_S_Param_ID.....	138, 353	runActualValue.....	428
		Interface of the IModuleControl.....	428
		InitModule.....	428

SwitchToOperationMode.....	428	MB_Home, special features of Sercos-based axes.....	197
SwitchToParamMode.....	428	MB_IDN.....	83, 355
Interface of the IOperationModeHandler.....	428	MB_INTERPOLATION_MODE.....	77
bCyclic.....	428	MB_MASTER_TYPE.....	87
bInitOperationCard.....	428	MB_MotionProfile.....	210
Interface of the IOutputData.....	428	MB_Phasing.....	201
runOutputData.....	428	MB_PhasingSlave.....	203
InTorque.....	24	MB_POSITION_EVAL.....	77
InVelocity.....	24	MB_PROBE_CONFIG.....	79
IOperationModeHandler.....	425	MB_PROBE_CONFIG, special features of SERCOS-based axes.....	79
getIOperationModeHandler.....	425	MB_PROBE_DATA.....	78
IOutputData.....	425	MB_PROBE_INPUT.....	78
getIOutputData.....	426	MB_PROBE_MODULE.....	77
IRetainHandler.....		MB_PROFILE_START_POINT.....	87
getIRetainHandler.....	426	MB_PROFILE_STEP.....	89
K		MB_ReadLintParameter.....	281
KIN_OBJECTS.....	41	MB_ReadLintParameter, special features of SERCOS-based axes.....	282
L		MB_ReadListParameter.....	283, 381
Length of the parameter channel, IndraDrive... Library	418	MB_ReadListParameter, special features of SERCOS-based axes.....	285
ML_Base.library.....	81	MB_ReadListParameter, special features of the field bus drives.....	383
ML_ControllerBase.....	425	MB_ReadListSegParameterPtr.....	384
ML_IPOImpactControl.library.....	435	MB_ReadListSegParameterPtr, special features of the field bus drives.....	386
ML_ParamBase.library.....	445	MB_ReadLrealListParameter.....	285
ML_PLCopen.library.....	169	MB_ReadLrealListParameter, special features of SERCOS-based axes.....	288
RIL_CommonTypes.library.....	37	MB_ReadLrealParameter.....	288
RMB_PLCopenFieldBus.library.....	353	MB_ReadLrealParameter, special features of SERCOS-based axes.....	289
Logbook access.....	124	MB_ReadMaxRealValue.....	290, 387
M		MB_ReadMaxRealValue, special features of SERCOS-based axes.....	291
MB_ChangeCamData.....	223	MB_ReadMaxRealValue, special features of the field bus drives.....	388
MB_ChangeCamData, special features of SERCOS-based axes.....	224	MB_ReadMaxValue.....	291, 388
MB_ChangeFlexEventSet.....	239	MB_ReadMaxValue, special features of SERCOS-based axes.....	292
MB_ChangeFlexProfileSet.....	236	MB_ReadMaxValue, special features of the field bus drives.....	389
MB_ChangeProfileSet.....	216	MB_ReadMinRealValue.....	293, 390
MB_ChangeProfileSet, special features of SERCOS-based axes.....	218	MB_ReadMinRealValue, special features of SERCOS-based axes.....	294
MB_ChangeProfileStep.....	219	MB_ReadMinRealValue, special features of the field bus drives.....	391
MB_ChangeProfileStep, special features of SERCOS-based axes.....	222	MB_ReadMinValue.....	294, 391
MB_CHANGESTEP_MODE.....	84	MB_ReadMinValue, special features of SERCOS-based axes.....	295
MB_Command.....	277	MB_ReadMinValue, special features of the field bus drives.....	392
MB_DIFF_EVAL.....	75	MB_ReadName.....	296
MB_EVENT_ACTION.....	84	MB_ReadName, special features of SERCOS-based axes.....	297
MB_EVENT_TRIGGER.....	86	MB_ReadParameter.....	297, 393
MB_EXECUTION_MODE.....	86		
MB_EXPECT_WINDOW_MODE.....	76		
MB_FLEX_EVENT.....	87		
MB_GearInPos.....	245		
MB_GearInPos, special features of Sercos-based axes.....	246		
MB_Home.....	196, 365		
MB_Home, special features of field bus drives.	366		

Index

MB_ReadParameter, special features of SERCOS-based axes.....	298	MB_WriteStringParameter, special features of the field bus drives.....	408
MB_ReadParameter, special features of the field bus drives.....	394	MC_AddAxisToGroup.....	248
MB_ReadRealParameter.....	299, 394	MC_CAM_ID	43
MB_ReadRealParameter, special features of SERCOS-based axes.....	300	MC_CamIn.....	205
MB_ReadRealParameter, special features of the field bus drives.....	395	MC_CamIn, special features of sercos- based axes.....	207
MB_ReadSercosAttribute.....	300, 396	MC_CamOut.....	209
MB_ReadSercosAttribute, Special features of SERCOS-based axes.....	301	MC_CamOut, Special features of SERCOS- based axes.....	209
MB_ReadSercosAttribute, special features of the field bus drives.....	397	MC_Command, special features of SERCOS-based axes.....	280
MB_ReadSercosDataStatus.....	302	MC_GearIn.....	240
MB_ReadSercosDataStatus, Special fea- tures of SERCOS-based axes.....	303	MC_GearIn, special features of Sercos- based axes.....	242
MB_ReadStringParameter.....	304, 397	MC_GearOut.....	243
MB_ReadStringParameter, special features of SERCOS-based axes.....	305	MC_GearOut, Special features of SERCOS- based axes.....	244
MB_ReadStringParameter, special features of the field bus drives.....	398	MC_MotionProfile, special features of Sercos-based axes.....	212
MB_ReadUnit.....	305	MC_MoveAbsolute.....	180, 366
MB_ReadUnit, special features of SERCOS- based axes.....	306	MC_MoveAbsolute, special features of field bus drives.....	368
MB_SERCOS_DATA_STATUS.....	90	MC_MoveAbsolute, special features of SERCOS-based axes.....	182
MB_SetPositionControlMode.....	195	MC_MoveAdditive.....	183, 368
MB_STEP_TYPE.....	89	MC_MoveAdditive, special features of field bus drives.....	370
MB_Stop.....	363	MC_MoveAdditive, special features of SERCOS-based axes.....	185
MB_Stop, special features of field bus drives...	365	MC_MoveRelative.....	185, 371
MB_SYNC_TYPE.....	90	MC_MoveRelative, special features of field bus drives.....	373
MB_WriteLintParameter.....	306	MC_MoveRelative, special features of SERCOS-based axes.....	188
MB_WriteLintParameter, special features of SERCOS-based axes.....	307	MC_MoveVelocity.....	189, 374
MB_WriteListParameter.....	308, 399	MC_MoveVelocity, special features of field bus drives.....	377
MB_WriteListParameter, special features of SERCOS-based axes.....	310	MC_MoveVelocity, special features of SERCOS-based axes.....	192
MB_WriteListParameter, special features of the field bus drives.....	401	MC_Power.....	176, 359
MB_WriteListSegParameterPtr.....	401	MC_Power, special features of field bus drives	360
MB_WriteListSegParameterPtr, special fea- tures of the field bus drives.....	404	MC_Power, Special features of Sercos- based axes.....	177
MB_WriteLrealParameter.....	310	MC_ReadActualPosition.....	316, 408
MB_WriteLrealParameter, special features of SERCOS-based axes.....	311	MC_ReadActualPosition, special features of SERCOS-based axes.....	317
MB_WriteParameter.....	312, 404	MC_ReadActualPosition, special features of the field bus drives.....	409
MB_WriteParameter, special features of SERCOS-based axes.....	313	MC_ReadAxisError.....	317, 410
MB_WriteParameter, special features of the field bus drives.....	405	MC_ReadAxisError, special features of SERCOS-based axes.....	318
MB_WriteRealParameter.....	313, 405	MC_ReadAxisError, special features of the field bus drives.....	411
MB_WriteRealParameter, special features of SERCOS-based axes.....	314	MC_ReadStatus.....	318, 411
MB_WriteRealParameter, special features of the field bus drives.....	406	MC_ReadStatus, Special features of sercos- based axes.....	320
MB_WriteStringParameter.....	315, 407		
MB_WriteStringParameter, special features of SERCOS-based axes.....	316		

MC_ReadStatus, special features of the field bus drives.....	413	GenerateCopyActualTrq.....	432
MC_RemoveAxisFromGroup.....	249	GenerateCopyActualVel.....	432
MC_Reset.....	177, 360	GenerateCopyCyclicAcceleration.....	433
MC_Reset, special features of field bus drives.....	361	GenerateCopyCyclicAnalog.....	433
MC_Reset, Special features of SERCOS-based axes.....	178	GenerateCopyCyclicPosition.....	433
MC_START_MODE.....	68	GenerateCopyCyclicTorque.....	433
MC_STEP_MODE.....	44	GenerateCopyCyclicVelocity.....	433
MC_Stop.....	178, 362	ML_GetCpuLoad.....	156
MC_Stop, special features of field bus drives...	363	ML_GetDiagnosisLog.....	157
MC_Stop, Special features of SERCOS-based axes.....	180	ML_GetFirmwareString.....	157
MC_SYNC_DIRECTION.....	70	ML_GetGateway.....	144
MC_TorqueControl.....	192	ML_GetHardwareInfoCtrl.....	158
MC_TorqueControl, special features of Sercos-based axes.....	194	ML_GetInfoApplication.....	159
ML_AbortTrigger.....	321	ML_GetIpAddress.....	145
ML_AXISDATA_SM.....	118	ML_GetMacAddress.....	146
ML_Base, data types at the interface of the IndraMotion function blocks.....	81	ML_GetMaxTemperature.....	160
ML_Base.library.....	81	ML_GetNewestDiagnosisIndex.....	160
ML_CONTROLDATA_SM.....	120	ML_GetOhcFan.....	160
ML_ControllerBase.....	425	ML_GetScalingType.....	161
ML_ControlOff.....	264	ML_GetSubnetMask.....	147
ML_ControlOn.....	264	ML_GetSystemTimeUTC.....	153
ML_ControlWriteActualData.....	265	ML_GetTargetInfo.....	161
ML_Convert.....	427	ML_GetTemperature.....	161
convFromAxis_Acceleration.....	427	ML_GetWarningCpuLoad.....	162
convFromAxis_Analog.....	427	ML_InitTouchProbe.....	322
convFromAxis_Force.....	427	ML_IPOImpactControl.....	436
convFromAxis_Position.....	427	mGetAcceleration.....	438
convFromAxis_Velocity.....	427	mGetAxisControlledBy.....	441
convToAxis_Acceleration.....	427	mGetBreakBehaviour.....	439
convToAxis_Analog.....	427	mGetDirection.....	438
convToAxis_Force.....	427	mGetNewState.....	437
convToAxis_Position.....	427	mGetPosition.....	440
convToAxis_Velocity.....	427	mGetValidValues.....	440
ML_Convert_DoublePrecision.....	427	mGetVelocity.....	441
convFromAxis_Acceleration.....	427	mSetAcceleration.....	438
convFromAxis_Analog.....	427	mSetBreakBehaviour.....	439
convFromAxis_Force.....	427	mSetDirection.....	437
convFromAxis_Position.....	427	mSetNewState.....	437
convFromAxis_Velocity.....	427	mSetPosition.....	439
convToAxis_Acceleration.....	427	mSetValidValues.....	440
convToAxis_Analog.....	427	mSetVelocity.....	441
convToAxis_Force.....	427	ML_IPOImpactControl methods.....	437
convToAxis_Position.....	427	ML_IPOImpactControl.library.....	435
convToAxis_Velocity.....	427	ML_LINK.....	121
ML_DAYSOFWEEK.....	156	ML_LOGBOOK.....	129
ML_DIAGNOSIS_LOG.....	163	ML_MessageHandler.....	433
ML_FlexProfile.....	224	run_MSGH_ClearErrors.....	433
ML_FlexProfile, special features of Sercos-based axes.....	227	run_MSGH_ClearWarning.....	433
ML_Generator.....	432	run_MSGH_SetMessagecode.....	433
GenerateCopyActualAcc.....	432	ML_OORV.....	428
GenerateCopyActualAna.....	432	ML_OORV_CONTROLLERDATA.....	426
GenerateCopyActualPos.....	432	ML_OpenCyclicAnalogChannel.....	252
		ML_OpenCyclicAnalogChannelCtrl.....	266
		ML_OpenCyclicPositionChannel.....	254
		ML_OpenCyclicPositionChannelCtrl.....	268
		ML_OpenCyclicTorqueChannel.....	258
		ML_OpenCyclicTorqueChannelCtrl.....	270
		ML_OpenCyclicVelocityChannel.....	260

Index

ML_OpenCyclicVelocityChannelCtrl.....	272	Access methods.....	456
ML_ParamBase		Administration.....	457
Auxiliary functions.....	461	Attributes.....	455
Constants.....	461	dwGetOperationData.....	456
Data access.....	460	dwLoadDefaultData.....	457
Data types.....	459	dwSetOperationData.....	457
Language.....	459	iGetParamType.....	457
Language switching.....	446	Loading default values.....	457
PARAM_ExtSercosParameter, read date....	456	Parameter type.....	457
PARAM_Parametersystem.....	446	Write date.....	456
SERCOS_ERROR.....	461	PARAM_Language.....	459
ML_ParamBase.library.....	445	PARAM_Language_Array.....	459
ML_PLCopen.library.....	169	PARAM_Parametersystem.....	446
ML_PLSDATA_SM.....	120	addParameter.....	446
ML_PROBEDATA_SM.....	121	Cyclic write protection.....	450
ML_ReadDiagLog.....	124	dwCreateListAllCommands.....	449
ML_SERCOS.....	122	dwCreateListAllParam.....	449
ML_SetAxisDiagEntry.....	125	dwCreateListSaveOnCF.....	449
ML_SetDiagEntry.....	127	dwLoadDefaultParam.....	448
ML_SetGateway.....	148	dwLoadFromCF.....	448
ML_SetIpAddress.....	150	dwReadWriteParamType.....	450
ML_SetSubnetMask.....	151	dwSaveToCF.....	448
ML_SetSystemTimeUTC.....	154	dwSetCyclicConfigured.....	450
ML_SetWarningCpuLoad.....	162	Functionalities.....	446
ML_SYNC.....	122	Init.....	446
ML_TouchProbe.....	329	Initialization.....	446
ML_TouchProbeContinuous.....	331	Lists of all commands.....	449
ML_WriteCyclicAnalog.....	253	Lists of all parameters.....	449
ML_WriteCyclicAnalogCtrl.....	267	Lists of all parameters saved on the CF	
ML_WriteCyclicPosition.....	255	card.....	449
ML_WriteCyclicPositionCtrl.....	269	LoadDefaultParam.....	448
ML_WriteCyclicTorque.....	259	Loading default values.....	448
ML_WriteCyclicTorqueCtrl.....	271	ParamRead.....	447
ML_WriteCyclicVelocity.....	261	ParamWrite.....	447
ML_WriteCyclicVelocityCtrl.....	274	Reading parameters.....	447
ML_WriteExpectWindow.....	334	Registering parameters.....	446
MLC_TABLE.....	343	Restoring parameters.....	448
MotionControl action chain.....	198	Saving parameters.....	447
MP_Drive.....	413	Synchronize parameter values.....	450
MP_DriveParameter.....	413	PARAM_SercosParameter.....	450
MP_FB_IndraDrv_AcyclicCommunication.....	413	Access methods.....	452
Multi-axis modules		Attributes.....	452
Profibus.....	380	Command interface.....	451
		Cyclic write protection.....	455
N		Interfaces.....	450
Namespace.....	420	Loading default values.....	454
		Parameter type.....	454
O		Read attribute.....	453
OBJECTS.....	40	Read data status.....	452
Oscilloscope data via function variable access.	116	Read date.....	454
		Read IDN.....	453
P		Read interface.....	450
PARAM_CmdExecuter.....	457	Read maximum value.....	453
dwAddCmd.....	457	Read minimum value.....	453
dwClearStorage.....	458	Read name.....	453
dwSetControlCmd.....	458	Write date.....	454
PARAM_ExtSercosParameter.....	455	Wwrite interface.....	451
		PARAM_SercosParameter,	

Index

Administration.....	454
Parameter access of the PLC program.....	133, 275
PLC access to parameter values	
via Shared Memory.....	94
With function variable access.....	96
PLC access to the parameter values	
With function variable access synchro- nously in the C2C link.....	98
PLC controls drives via field bus connection....	413
PLS data via function variable access.....	108
PLsData[] - PLS data in shared memory.....	106
PROBECONFIG.....	172
ProbeSelect.....	172
Programmable limit switch data via shared memory and function variable access.....	105

R

REFERENCE_TO	
Axis data.....	93, 95
Control data.....	101
Control Data.....	103
Oscilloscope data.....	116
PLS data.....	106, 108
Touch probe data.....	111, 113
REFERENCE_TO to a function variable syn- chronously in the C2C link.....	98
RIL_CommonTypes, data types on the inter- face of IndraMotion function blocks.....	37
RIL_CommonTypes.library.....	37
RMB_PLCOpenFieldBus.library.....	353
Application.....	419

S

Safety alert symbol.....	13
Safety instructions.....	13
sercos III, C2C link	
MotionControl action chain.....	200
SERCOS_TABLE.....	340
Service hotline.....	465
Shared memory	
ControlData[].....	101
ProbeData[].....	111
Shared Memory	
AxisData[].....	93
PLsData[].....	106
Shutdown.....	24
Signal words.....	13
SM_EXT_TYPES.....	123
SM_TYPES.....	123
Standard interfaces at function blocks.....	23
State diagram.....	357
State diagram of a real axis.....	172
Status during function variable access....	97,
104, 109, 115, 118	
Status during synchronous function variable access in the C2C link.....	99
Support.....	465

Symbols.....	15
SYS_TIME_DATE.....	74
SYS_TIME64.....	73

T

TARGET_INFO.....	164
Time required for data transport with read and write parameter function blocks.....	277
Time required to process Motion function blocks in the IndraMotion MLC.....	337
Touch probe data in shared memory.....	111
Touch probe data via function variable access.	113
Touch probe data via shared memory and function variable access.....	110
TOUCHPROBE_REF.....	42
TP_OBJECTS.....	42
Twin axis modules	
Profibus.....	380

U

User diagnostics.....	124

W

Warnings.....	13

Notes

Notes

Bosch Rexroth AG

Electric Drives and Controls

P.O. Box 13 57

97803 Lohr, Germany

Bgm.-Dr.-Nebel-Str. 2

97816 Lohr, Germany

Tel. +49 9352 18 0

Fax +49 9352 18 8400

www.boschrexroth.com/electrics



R911336360