Kollmorgen Automation Suite

KAS Reference Guide - Motion Library

Document Edition: V, December 2023
Valid for KAS Software Revision 4.01
Part Number: 959716

For safe and proper use, follow these instructions. Keep for future use.

1 Table of Contents

1 Table of Contents **2**
2 Fieldbus Library **23**
2.1 EtherCAT Library **23**
2.2 EtherNet/IP (ODVA) **23**
2.3 Fieldbus - EtherCAT **23**
2.3.1 Remarks 24
2.3.1.1 Why use ECATReadSdo and ECATWriteSdo FBs? 24
2.3.1.2 Why use the DriveParam FBs? 24
2.3.2 EtherCAT Function Blocks that Work with Drive Parameters 24
2.3.2.1 Execution Time 25
2.3.3 EtherCAT Function Blocks that Work with SDOs 26
2.3.4 DriveParamRead 26
2.3.4.1 Inputs 26
2.3.4.2 Outputs 27
2.3.4.3 Remarks 27
2.3.4.4 FBD Language Example 30
2.3.4.5 FFLD Language Example 30
2.3.4.6 IL Language Example 30
2.3.4.7 ST Language Example 30
2.3.5 DriveParamStrRead 31
2.3.5.1 Inputs 31
2.3.5.2 Outputs 31
2.3.5.3 Remarks 31
2.3.5.4 FBD Language Example 34
2.3.5.5 FFLD Language Example 34
2.3.5.6 IL Language Example 34
2.3.5.7 ST Language Example 34
2.3.6 DriveParamWrite 34
2.3.6.1 Inputs 34
2.3.6.2 Outputs 35
2.3.6.3 Remarks 35
2.3.6.4 FBD Language Example 38
2.3.6.5 FFLD Language Example 38
2.3.6.6 IL Language Example 38
2.3.6.7 ST Language Example 38
2.3.7 ECATCommErrors 38
2.3.7.1 Inputs 38
2.3.7.2 Outputs 39
2.3.7.3 Remarks 39
2.3.7.4 FBD Language Example 40
2.3.7.5 FFLD Language Example 40
2.3.7.6 IL Language Example 40
2.3.7.7 ST Language Example 40
2.3.8 ECATDeviceAction 40
2.3.8.1 Inputs 40

2.3.8.2 Outputs 41
2.3.8.3 Remarks 41
2.3.8.4 Error Codes 42
2.3.8.5 FBD Language Example 42
2.3.8.6 FFLD Language Example 43
2.3.8.7 IL Language Example 43
2.3.8.8 ST Language Example 43
2.3.9 ECATDeviceStatus 43
2.3.9.1 Inputs 43
2.3.9.2 Outputs 44
2.3.9.3 Remarks 45
2.3.9.4 FBD Language Example 46
2.3.9.5 FFLD Language Example 46
2.3.9.6 IL Language Example 46
2.3.9.7 ST Language Example 46
2.3.10 ECATDevReadParam 46
2.3.10.1 Inputs 46
2.3.10.2 Outputs 47
2.3.10.3 Remarks 47
2.3.10.4 FBD Language Example 49
2.3.10.5 FFLD Language Example 50
2.3.10.6 IL Language Example 50
2.3.10.7 ST Language Example 50
2.3.11 ECATGetObjVal 50
2.3.12 ECATMasterStatus 51
2.3.12.1 Inputs 51
2.3.12.2 Outputs 51
2.3.12.3 Remarks 51
2.3.12.4 FBD Language Example 52
2.3.12.5 FFLD Language Example 52
2.3.12.6 IL Language Example 52
2.3.12.7 ST Language Example 52
2.3.13 ECATReadData 52
2.3.13.1 Inputs 53
2.3.13.2 Outputs 53
2.3.13.3 Remarks 53
2.3.13.4 FBD Language Example 53
2.3.13.5 FFLD Language Example 54
2.3.13.6 IL Language Example 54
2.3.13.7 ST Language Example 54
2.3.14 ECATReadSDO 54
2.3.14.1 Inputs 54
2.3.14.2 Outputs 55
2.3.14.3 Remarks 56
2.3.14.4 FBD Language Example 57
2.3.14.5 FFLD Language Example 57
2.3.14.6 IL Language Example 58
2.3.14.7 ST Language Example 58

2.3.15 ECATWCStatus ... 58
2.3.15.1 Inputs ... 58
2.3.15.2 Outputs ... 58
2.3.15.3 Remarks ... 58
2.3.15.4 FBD Language Example ... 59
2.3.15.5 FFLD Language Example ... 59
2.3.15.6 IL Language Example ... 59
2.3.15.7 ST Language Example ... 59
2.3.16 ECATWriteData ... 59
2.3.16.1 Inputs ... 59
2.3.16.2 Outputs ... 60
2.3.16.3 Remarks ... 60
2.3.16.4 FBD Language Example ... 60
2.3.16.5 FFLD Language Example ... 60
2.3.16.6 IL Language Example ... 61
2.3.16.7 ST Language Example ... 61
2.3.17 ECATWriteSDO ... 61
2.3.17.1 Inputs ... 61
2.3.17.2 Outputs ... 63
2.3.17.3 Remarks ... 63
2.3.17.4 FBD Language Example ... 64
2.3.17.5 FFLD Language Example ... 65
2.3.17.6 IL Language Example ... 65
2.3.17.7 ST Language Example ... 65
2.3.18 FSoEParamsInit ... 65
2.3.18.1 Inputs ... 65
2.3.18.2 Outputs ... 66
2.3.18.3 Remarks ... 66
2.3.18.4 FBD Language Example ... 68
2.3.18.5 FFLD Language Example ... 68
2.3.18.6 IL Language Example ... 68
2.3.18.7 ST Language Example ... 68
2.4 Fieldbus - EtherNet/IP (ODVA) ... 69
2.4.1 eipAdapter ... 69
2.4.1.1 Inputs ... 69
2.4.1.2 Outputs ... 69
2.4.1.3 Remarks ... 69
2.4.2 eipReadAttr ... 70
2.4.2.1 Inputs ... 70
2.4.2.2 Outputs ... 70
2.4.2.3 Remarks ... 71
2.4.2.4 FBD Language Example ... 71
2.4.2.5 FFLD Language Example ... 71
2.4.2.6 IL Language Example ... 71
2.4.2.7 ST Language Example ... 71
2.4.3 eipWriteAttr ... 72
2.4.3.1 Inputs ... 72
2.4.3.2 Outputs ... 72

2.4.3.3 Remarks ... 72
2.4.3.4 FBD Language Example ... 73
2.4.3.5 FFLD Language Example ... 73
2.4.3.6 IL Language Example ... 73
2.4.3.7 ST Language Example ... 73
3 Kollmorgen UDFBs ... 75
3.1 Common ... 75
3.2 Pipe Network ... 75
3.3 PLCopen ... 76
3.4 PLC ... 76
3.5 Create or Edit UDFBs ... 77
3.5.1 Create a UDFB Instance ... 77
3.5.2 Edit UDFBs ... 77
3.6 UDFB - Motion / Common ... 78
3.6.1 FB_AKDFltRpt ... 78
3.6.1.1 Inputs ... 78
3.6.1.2 Outputs ... 79
3.6.1.3 Remarks ... 79
3.6.1.4 FBD Language Example ... 80
3.6.1.5 FFLD Language Example ... 80
3.6.1.6 IL Language Example ... 80
3.6.1.7 ST Language Example ... 80
3.6.2 FB_AxisPlsPosModulo ... 81
3.6.2.1 Inputs ... 81
3.6.2.2 Outputs ... 81
3.6.2.3 Remarks ... 81
3.6.2.4 FBD Language Example ... 82
3.6.2.5 FFLD Language Example ... 82
3.6.2.6 IL Language Example ... 83
3.6.2.7 ST Language Example ... 83
3.6.3 FB_AxisPlsPosNoModulo ... 83
3.6.3.1 Inputs ... 83
3.6.3.2 Outputs ... 83
3.6.3.3 Remarks ... 84
3.6.3.4 FBD Language Example ... 84
3.6.3.5 FFLD Language Example ... 84
3.6.3.6 IL Language Example ... 85
3.6.3.7 ST Language Example ... 85
3.6.4 FB_Cylinder ... 85
3.6.4.1 Inputs ... 85
3.6.4.2 Outputs ... 85
3.6.4.3 Remarks ... 86
3.6.4.4 FBD Language Example ... 86
3.6.4.5 FFLD Language Example ... 86
3.6.4.6 IL Language Example ... 87
3.6.4.7 ST Language Example ... 87
3.6.5 FB_S700FltRpt ... 87
3.6.5.1 Inputs ... 87

3.6.5.2 Outputs ... 87
3.6.5.3 Remarks ... 88
3.6.5.4 FBD Language Example ... 88
3.6.5.5 FFLD Language Example ... 89
3.6.5.6 IL Language Example ... 89
3.6.5.7 ST Language Example ... 89
3.7 UDFB - Motion / Pipe Network ... 89
3.7.1 MLFB_DriveFault ... 90
3.7.1.1 Inputs ... 90
3.7.1.2 Outputs ... 90
3.7.1.3 Remarks ... 91
3.7.1.4 FBD Language Example ... 92
3.7.1.5 FFLD Language Example ... 92
3.7.1.6 IL Language Example ... 92
3.7.1.7 ST Language Example ... 92
3.7.2 MLFB_ECATRestart ... 93
3.7.2.1 Inputs ... 93
3.7.2.2 Outputs ... 93
3.7.2.3 Remarks ... 93
3.7.2.4 FBD Language Example ... 93
3.7.2.5 FFLD Language Example ... 94
3.7.2.6 IL Language Example ... 94
3.7.2.7 ST Language Example ... 94
3.7.3 Fast Homing - UDFB - Motion/Pipe Network ... 94
3.7.3.1 MLFB_HomeFindHomeFastInput ... 94
3.7.3.2 MLFB_HomeFindHomeFastInputModulo ... 98
3.7.3.3 MLFB_HomeFindLimitFastInput ... 103
3.7.3.4 MLFB_HomeFindLimitFastInputModulo ... 106
3.7.4 Homing - UDFB - Motion/Pipe Network ... 110
3.7.4.1 MLFB_HomeFindHomeInput ... 110
3.7.4.2 MLFB_HomeFindHomeInputThenZeroAngle ... 113
3.7.4.3 MLFB_HomeFindLimitInput ... 115
3.7.4.4 MLFB_HomeFindLimitInputThenZeroAngle ... 117
3.7.4.5 MLFB_HomeFindZeroAngle ... 119
3.7.4.6 MLFB_HomeMoveUntilPosErrExceeded ... 121
3.7.4.7 MLFB_HomeMoveUntilPosErrExceededThenZeroAngle ... 123
3.7.4.8 MLFB_HomeUsingCurrentPosition ... 126
3.7.5 MLFB_Jog ... 127
3.7.5.1 Inputs ... 127
3.7.5.2 Outputs ... 128
3.7.5.3 Remarks ... 128
3.7.5.4 FBD Language Example ... 128
3.7.5.5 FFLD Language Example ... 128
3.7.5.6 IL Language Example ... 128
3.7.5.7 ST Language Example ... 129
3.7.6 PLS - UDFB - Motion/Pipe Network ... 129
3.7.6.1 MLFB_PlsPosFw ... 129
3.7.6.2 MLFB_PlsPosFwBw ... 130

3.7.6.3 MLFB_PlsTimeFw ... 132
3.8 UDFB - Motion / PLCopen ... 134
3.8.1 Drive Fault Reporting - UDFB - Motion/PLCopen ... 134
3.8.1.1 MCFB_AKDFault ... 135
3.8.1.2 MCFB_AKDFaultLookup ... 136
3.8.1.3 MCFB_DriveFault ... 137
3.8.2 MCFB_ECATRestart ... 140
3.8.2.1 Inputs ... 140
3.8.2.2 Outputs ... 140
3.8.2.3 Remarks ... 140
3.8.2.4 FBD Language Example ... 141
3.8.2.5 FFLD Language Example ... 141
3.8.2.6 IL Language Example ... 141
3.8.2.7 ST Language Example ... 141
3.8.3 MCFB_StepAbsSwitchFastInput ... 141
3.8.3.1 Inputs ... 142
3.8.3.2 Outputs ... 143
3.8.3.3 Remarks ... 144
3.8.3.4 FBD Language Example ... 145
3.8.3.5 FFLD Language Example ... 146
3.8.3.6 IL Language Example ... 146
3.8.3.7 ST Language Example ... 146
3.8.4 MCFB_StepLimitSwitchFastInput ... 146
3.8.4.1 Inputs ... 147
3.8.4.2 Outputs ... 148
3.8.4.3 Remarks ... 148
3.8.4.4 FBD Language Example ... 149
3.8.4.5 FFLD Language Example ... 150
3.8.4.6 IL Language Example ... 150
3.8.4.7 ST Language Example ... 150
3.8.5 Homing - UDFB - Motion/PLCopen ... 151
3.8.5.1 MCFB_StepAbsolutes ... 151
3.8.5.2 MCFB_StepAbsSwitch ... 153
3.8.5.3 MCFB_StepBlock ... 158
3.8.5.4 MCFB_StepLimitSwitch ... 162
3.8.5.5 MCFB_StepRefPulse ... 166
3.8.6 MCFB_Jog ... 170
3.8.6.1 Inputs ... 170
3.8.6.2 Outputs ... 171
3.8.6.3 Remarks ... 171
3.8.6.4 FBD Language Example ... 171
3.8.6.5 FFLD Language Example ... 172
3.8.6.6 IL Language Example ... 172
3.8.6.7 ST Language Example ... 172
3.8.7 MCFB_GearedWebTension ... 172
3.8.7.1 Inputs ... 172
3.8.7.2 Outputs ... 173
3.8.7.3 Remarks ... 174

3.8.7.4 FBD Language Example ... 177
3.8.7.5 FFLD Language Example ... 177
3.8.7.6 IL Language Example ... 178
3.8.7.7 ST Language Example ... 178
3.9 UDFB - PLC ... 178
3.9.1 FB_ElapseTime ... 179
3.9.1.1 Inputs ... 179
3.9.1.2 Outputs ... 179
3.9.1.3 Remarks ... 179
3.9.1.4 FBD Language Example ... 180
3.9.1.5 FFLD Language Example ... 180
3.9.1.6 IL Language Example ... 180
3.9.1.7 ST Language Example ... 180
3.9.2 FB_FirstOrderDigitalFilter ... 180
3.9.2.1 Inputs ... 181
3.9.2.2 Outputs ... 181
3.9.2.3 Remarks ... 181
3.9.2.4 FBD Language Example ... 185
3.9.2.5 FFLD Language Example ... 185
3.9.2.6 IL Language Example ... 185
3.9.2.7 ST Language Example ... 185
3.9.3 FB_PWDutyOutput ... 185
3.9.3.1 Inputs ... 185
3.9.3.2 Outputs ... 186
3.9.3.3 Remarks ... 186
3.9.3.4 IL Language Example ... 188
3.9.3.5 ST Language Example ... 188
3.9.4 FB_ScaleInput - Scaling Analog IO ... 188
3.9.4.1 Inputs ... 188
3.9.4.2 Outputs ... 188
3.9.4.3 Remarks ... 189
3.9.4.4 FBD Language Example ... 189
3.9.4.5 FFLD Language Example ... 189
3.9.4.6 IL Language Example ... 189
3.9.4.7 ST Language Example ... 189
3.9.5 FB_ScaleOutput - Scaling Analog IO ... 190
3.9.5.1 Inputs ... 190
3.9.5.2 Outputs ... 190
3.9.5.3 Remarks ... 190
3.9.5.4 FBD Language Example ... 191
3.9.5.5 FFLD Language Example ... 191
3.9.5.6 IL Language Example ... 191
3.9.5.7 ST Language Example ... 191
3.9.6 FB_TemperaturePID ... 192
3.9.6.1 Inputs ... 192
3.9.6.2 Outputs ... 192
3.9.6.3 Remarks ... 192
3.9.6.4 FBD Language Example ... 194

3.9.6.5 FFLD Language Example ... 195
3.9.6.6 IL Language Example ... 195
3.9.6.7 ST Language Example ... 195
3.9.7 PipeNetwork_FFLD - Special FFLD Function ... 195
3.9.7.1 Inputs ... 195
3.9.7.2 Outputs ... 196
3.9.7.3 Remarks ... 196
3.9.7.4 FBD Language Example ... 196
3.9.7.5 FFLD Language Example ... 196
3.9.7.6 IL Language Example ... 197
3.9.7.7 ST Language Example ... 197
3.9.8 ProfilesCode_FFLD - Special FFLD Function ... 197
3.9.8.1 Inputs ... 197
3.9.8.2 Outputs ... 197
3.9.8.3 Remarks ... 197
3.9.8.4 FBD Language Example ... 198
3.9.8.5 FFLD Language Example ... 198
3.9.8.6 IL Language Example ... 198
3.9.8.7 ST Language Example ... 198
4 Motion Library ... 199
4.1 Motion / Pipe Network ... 199
4.1.1 Adder ... 199
4.1.1.1 MLAddInit ... 200
4.1.1.2 MLAddReadOff1 ... 201
4.1.1.3 MLAddReadOff2 ... 202
4.1.1.4 MLAddReadRatio1 ... 204
4.1.1.5 MLAddReadRatio2 ... 205
4.1.1.6 MLAddWriteInput ... 206
4.1.1.7 MLAddWriteOff1 ... 207
4.1.1.8 MLAddWriteOff2 ... 208
4.1.1.9 MLAddWriteRat1 ... 210
4.1.1.10 MLAddWriteRat2 ... 211
4.1.2 Block ... 212
4.1.2.1 MLBlkCreate ... 212
4.1.2.2 MLBlkIsReady ... 213
4.1.2.3 MLBlkReadModPos ... 214
4.1.2.4 MLBlkReadOutVal ... 215
4.1.2.5 MLBlkWriteModPos ... 216
4.1.3 Pipe ... 217
4.1.3.1 MLPipeAct ... 217
4.1.3.2 MLPipeAddBlock ... 219
4.1.3.3 MLPipeCreate ... 220
4.1.3.4 MLPipeDeact ... 221
4.1.4 Axis ... 223
4.1.4.1 Function by Types ... 223
4.1.4.2 Functions in Alphabetical Order ... 223
4.1.4.3 MLAxisAbs ... 226
4.1.4.4 Forcing the Direction of Rotation ... 227

4.1.4.5 MLAxisAdd 229
4.1.4.6 MLAxisAddress 231
4.1.4.7 MLAxisAddTq 232
4.1.4.8 MLAxisCfgFastIn 233
4.1.4.9 MLAxisCmdPos 234
4.1.4.10 MLAxisDriveNumber 235
4.1.4.11 MLAxisFBackPos 236
4.1.4.12 MLAxisGenEN 237
4.1.4.13 MLAxisGenIsEN 238
4.1.4.14 MLAxisGenIsRdy 238
4.1.4.15 MLAxisGenPos 239
4.1.4.16 MLAxisGenReadAcc 240
4.1.4.17 MLAxisGenReadDec 241
4.1.4.18 MLAxisGenReadSpd 242
4.1.4.19 MLAxisGenWriteAcc 243
4.1.4.20 MLAxisGenWriteDec 244
4.1.4.21 MLAxisGenWriteSpd 245
4.1.4.22 MLAxisInit 246
4.1.4.23 MLAxisIsCnctd 248
4.1.4.24 MLAxisIsTrigged 249
4.1.4.25 MLAxisMoveVel 250
4.1.4.26 MLAxisPipePos 251
4.1.4.27 MLAxisPower 252
4.1.4.28 MLAxisPowerDOff 253
4.1.4.29 MLAxisRatedTq 253
4.1.4.30 MLAxisReadActPos 255
4.1.4.31 MLAxisReadFBUnit 256
4.1.4.32 MLAxisReadFEUU 257
4.1.4.33 MLAxisReadGenStatus 257
4.1.4.34 MLAxisReadModPos 259
4.1.4.35 MLAxisReadTq 259
4.1.4.36 MLAxisReadUUnits 260
4.1.4.37 MLAxisReadVel 261
4.1.4.38 MLAxisReAlgnRdy 262
4.1.4.39 MLAxisReAlign 263
4.1.4.40 MLAxisRel 264
4.1.4.41 MLAxisResetErrors 266
4.1.4.42 MLAxisRstFastIn 266
4.1.4.43 MLAxisStatus 267
4.1.4.44 MLAxisStop 269
4.1.4.45 MLAxisTimeStamp 271
4.1.4.46 MLAxisWriteModPos 272
4.1.4.47 MLAxisWritePipPos 273
4.1.4.48 MLAxisWritePos 274
4.1.4.49 MLAxisWriteUUnits 276
4.1.4.50 MLPNAxisCreate 277
4.1.4.51 Examples of Axis Functions 278
4.1.5 CAM Profile 279

4.1.5.1 MLCamInit 280
4.1.5.2 MLCamSwitch 281
4.1.5.3 MLPrfReadIOffset 282
4.1.5.4 MLPrfReadIScale 284
4.1.5.5 MLPrfReadOOffset 285
4.1.5.6 MLPrfReadOScale 286
4.1.5.7 MLPrfWriteIOffset 287
4.1.5.8 MLPrfWriteIScale 289
4.1.5.9 MLPrfWriteOOffset 290
4.1.5.10 MLPrfWriteOScale 291
4.1.6 Comparator 293
4.1.6.1 Examples: Comparator Functions 293
4.1.6.2 MLCompCheck 295
4.1.6.3 MLCompInit 296
4.1.6.4 MLCompReadRef 298
4.1.6.5 MLCompReset 299
4.1.6.6 MLCompWriteRef 300
4.1.7 Convertor 301
4.1.7.1 MLCNVConECAT 302
4.1.7.2 MLCNVConnect 303
4.1.7.3 MLCNVConnectEx 305
4.1.7.4 MLCNVDisconnect 306
4.1.7.5 MLCNVInit 307
4.1.8 Delay 309
4.1.8.1 MLDelayInit 309
4.1.9 Derivator 310
4.1.9.1 MLDerInit 310
4.1.9.2 MLDerReadInModPos 311
4.1.9.3 MLDerWriteInModPos 313
4.1.10 Gear 314
4.1.10.1 Examples of Gear Functions 315
4.1.10.2 MLGearInit 317
4.1.10.3 Avoid Modulo Calculation Problems 319
4.1.10.4 MLGearReadOffset 320
4.1.10.5 MLGearReadOffSlp 321
4.1.10.6 MLGearReadRatio 321
4.1.10.7 MLGearReadRatSlp 322
4.1.10.8 MLGearWriteOff 323
4.1.10.9 MLGearWriteOSlp 324
4.1.10.10 MLGearWriteRatio 325
4.1.10.11 MLGearWriteRatSlp 326
4.1.10.12 With Offset 327
4.1.10.13 Without Offset 327
4.1.11 Integrator 328
4.1.11.1 MLIntWriteOutVal 328
4.1.11.2 MLIntInit 329
4.1.12 Master 331
4.1.12.1 Examples of Master Functions 331

4.1.12.2 MLMstAbs ... 332
4.1.12.3 Forcing the Direction of Rotation ... 334
4.1.12.4 MLMstAdd ... 336
4.1.12.5 MLMstForcePos ... 337
4.1.12.6 MLMstInit ... 338
4.1.12.7 MLMstReadAccel ... 341
4.1.12.8 MLMstReadDecel ... 342
4.1.12.9 MLMstReadInitPos ... 343
4.1.12.10 MLMstReadSpeed ... 344
4.1.12.11 MLMstRel ... 345
4.1.12.12 MLMstRun ... 346
4.1.12.13 MLMstStatus ... 347
4.1.12.14 MLMstWriteAccel ... 348
4.1.12.15 MLMstWriteDecel ... 349
4.1.12.16 MLMstWriteInitPos ... 350
4.1.12.17 MLMstWriteSpeed ... 351
4.1.13 Phaser ... 352
4.1.13.1 Example: Phaser Functions ... 353
4.1.13.2 MLPhaInit ... 354
4.1.13.3 MLPhaReadActPhase ... 355
4.1.13.4 MLPhaReadPhase ... 356
4.1.13.5 MLPhaReadSlope ... 357
4.1.13.6 MLPhaWritePhase ... 358
4.1.13.7 MLPhaWriteSlope ... 359
4.1.14 PMP ... 360
4.1.14.1 MLPmpAbs ... 361
4.1.14.2 MLPmpForcePos ... 362
4.1.14.3 MLPmpInit ... 363
4.1.14.4 MLPmpReadAccel ... 365
4.1.14.5 MLPmpReadFstSpd ... 366
4.1.14.6 MLPmpReadInitPos ... 367
4.1.14.7 MLPmpReadJerk ... 368
4.1.14.8 MLPmpReadLstSpd ... 369
4.1.14.9 MLPmpRel ... 370
4.1.14.10 MLPmpRun ... 372
4.1.14.11 MLPmpStatus ... 373
4.1.14.12 MLPmpWriteAccel ... 374
4.1.14.13 MLPmpWriteFstSpd ... 375
4.1.14.14 MLPmpWriteJerk ... 376
4.1.14.15 MLPmpWriteLstSpd ... 377
4.1.15 Sampler ... 378
4.1.15.1 MLSmpConECAT ... 379
4.1.15.2 MLSmpConnect ... 381
4.1.15.3 MLSmpConPLCAxis ... 382
4.1.15.4 MLSmpConPNAxis ... 383
4.1.15.5 MLSmpInit ... 384
4.1.16 Synchronizer ... 387
4.1.16.1 Example: Synchronizer Functions ... 387

4.1.16.2 MLSyncInit ... 388
4.1.16.3 MLSyncReadDeltaS ... 390
4.1.16.4 MLSyncStart ... 391
4.1.16.5 MLSyncStop ... 392
4.1.16.6 MLSyncWriteDeltaS ... 393
4.1.17 Trigger ... 395
4.1.17.1 Example: Trigger Functions ... 396
4.1.17.2 MLTrigClearFlag ... 397
4.1.17.3 MLTrigInit ... 398
4.1.17.4 MLTrigIsTrigged ... 400
4.1.17.5 MLTrigReadDelay ... 402
4.1.17.6 MLTrigReadPos ... 403
4.1.17.7 MLTrigReadTime ... 405
4.1.17.8 MLTrigSetEdge ... 407
4.1.17.9 MLTrigWriteDelay ... 408
4.2 Motion / PLCopen Library ... 409
4.2.1 Control ... 409
4.2.2 Information ... 409
4.2.3 I/O ... 409
4.2.4 PLCopenMotion ... 410
4.2.5 Profile ... 410
4.2.6 Reference ... 411
4.2.7 Registration ... 411
4.2.8 Control ... 411
4.2.8.1 MC_ClearFaults ... 411
4.2.8.2 MC_CreatePLCAxis ... 413
4.2.8.3 MC_EStop ... 416
4.2.8.4 MC_InitAxis ... 417
4.2.8.5 MC_InitAxisFeedback ... 419
4.2.8.6 MC_Power ... 420
4.2.8.7 MC_ResetError ... 422
4.2.8.8 MC_Stop ... 423
4.2.9 I/O ... 426
4.2.9.1 MC_AbortTrigger ... 426
4.2.9.2 MC_TouchProbe ... 428
4.2.10 Information ... 435
4.2.10.1 MC_ReadActPos ... 436
4.2.10.2 MC_ReadActVel ... 437
4.2.10.3 MC_ReadAxisErr ... 438
4.2.10.4 MC_ReadBoolPar ... 440
4.2.10.5 MC_ReadParam ... 441
4.2.10.6 MC_ReadStatus ... 442
4.2.10.7 MC_WriteBoolPar ... 444
4.2.10.8 MC_WriteParam ... 445
4.2.11 PLCopenMotion ... 447
4.2.11.1 MC_AddSuperAxis ... 447
4.2.11.2 MC_Halt ... 449
4.2.11.3 MC_MoveAbsolute ... 451

4.2.11.4 MC_MoveAdditive ... 456
4.2.11.5 MC_MoveContVel ... 459
4.2.11.6 MC_MoveRelative ... 464
4.2.11.7 MC_MoveSuperimp ... 467
4.2.11.8 MC_MoveVelocity ... 470
4.2.11.9 MC_RemSuperAxis ... 473
4.2.11.10 MC_SetOverride ... 475
4.2.12 Profile ... 476
4.2.12.1 MC_CamIn ... 476
4.2.12.2 MC_CamOut ... 484
4.2.12.3 MC_CamResumePos ... 486
4.2.12.4 MC_CamStartPos ... 488
4.2.12.5 MC_CamTblSelect ... 490
4.2.12.6 MC_GearIn ... 492
4.2.12.7 MC_GearInPos ... 495
4.2.12.8 MC_GearOut ... 503
4.2.12.9 MC_Phasing ... 505
4.2.12.10 MC_SyncSlaves ... 507
4.2.13 Reference ... 509
4.2.13.1 MC_Reference ... 509
4.2.13.2 MC_SetPos ... 514
4.2.13.3 MC_SetPosition ... 516
4.2.14 Registration ... 516
4.2.14.1 MC_MachRegist ... 516
4.2.14.2 Position Inputs ... 522
4.2.14.3 Option Inputs ... 523
4.2.14.4 MC_MarkRegist ... 525
4.2.14.5 MC_StopRegist ... 531
4.3 Motion / Common ... 532
4.3.1 Coordinated Motion ... 532
4.3.2 Info ... 532
4.3.3 Motion ... 532
4.3.4 Profiles ... 533
4.3.5 Motion / Common - Info ... 533
4.3.5.1 MC_ErrorDescription ... 533
4.3.6 Motion / Common - Profiles ... 534
4.3.6.1 MLProfileBuild ... 534
4.3.6.2 MLProfileCreate ... 542
4.3.6.3 MLProfileInit ... 544
4.3.6.4 MLProfileRelease ... 545
4.3.7 Motion / Common - Motion ... 548
4.3.7.1 State Machine ... 548
4.3.7.2 MLMotionCycleTime ... 548
4.3.7.3 MLMotionInit ... 549
4.3.7.4 MLMotionRstErr ... 551
4.3.7.5 MLMotionStart ... 552
4.3.7.6 MLMotionStatus ... 554
4.3.7.7 MLMotionStop ... 556

4.3.8 Motion / Common - Coordinated Motion 557
4.3.8.1 Group Control 557
4.3.8.2 Info 558
4.3.8.3 Motion 558
4.3.8.4 Reference 559
4.3.8.5 Coordinated Motion - Group Control Library 559
4.3.9 Example: Valid Setup 563
4.3.10 Example: Invalid Setup 563
4.3.11 Related Functions 574
4.3.12 MC_TRANSFORM_TYPE_GANTRY 586
4.3.13 MC_TRANSFORM_TYPE_GANTRY_WITH_SKEW 586
4.3.14 MC_TRANSFORM_TYPE_HBOT 586
4.3.15 MC_TRANSFORM_TYPE_TBOT 587
4.3.16 MC_TRANSFORM_TYPE_DELTA (*Experimental*) 587
4.3.17 MC_TRANSFORM_TYPE_SCARA_ELBOW_POS (*Experimental*) 589
4.3.18 MC_TRANSFORM_TYPE_SCARA_ELBOW_NEG (*Experimental*) 590
4.3.18.1 Motion / Common - Coordinated Motion - Info 591
4.3.18.2 Motion / Common - Coordinated Motion - Motion 609
4.3.18.3 Coordinated Motion - Reference Library 647
5 System Library 651
5.1 Controller 651
5.2 File 651
5.3 TCP/IP Sockets 651
5.4 UDP 652
5.5 Controller Library 652
5.5.1 ClearCtrlErrors 652
5.5.1.1 Inputs 652
5.5.1.2 Outputs 653
5.5.1.3 Remarks 653
5.5.1.4 FBD Language Example 653
5.5.1.5 FFLD Language Example 653
5.5.1.6 IL Language Example 653
5.5.1.7 ST Language Example 653
5.5.2 GetCtrlErrors 653
5.5.2.1 Inputs 653
5.5.2.2 Outputs 654
5.5.2.3 Remarks 654
5.5.2.4 FBD Language Example 654
5.5.2.5 FFLD Language Example 654
5.5.2.6 IL Language Example 655
5.5.2.7 ST Language Example 655
5.5.3 GetCtrlInfo 655
5.5.3.1 Inputs 655
5.5.3.2 Outputs 656
5.5.3.3 Remarks 657
5.5.3.4 FBD Language Example 657
5.5.3.5 FFLD Language Example 658
5.5.3.6 IL Language Example 658

5.5.3.7 ST Language Example 658
5.5.4 GetCtrlPerf 658
5.5.4.1 Inputs 658
5.5.4.2 Outputs 658
5.5.4.3 Remarks 660
5.5.4.4 FBD Language Example 661
5.5.4.5 FFLD Language Example 661
5.5.4.6 IL Language Example 662
5.5.4.7 ST Language Example 662
5.6 File Tools Function Blocks 662
5.6.1 FileClose 663
5.6.1.1 Inputs 663
5.6.1.2 Outputs 663
5.6.1.3 Remarks 663
5.6.1.4 FBD Language Example 664
5.6.1.5 FFLD Language Example 664
5.6.1.6 IL Language Example 664
5.6.1.7 ST Language Example 664
5.6.2 FileCopy 664
5.6.2.1 Inputs 664
5.6.2.2 Outputs 665
5.6.2.3 Remarks 665
5.6.2.4 FBD Language Example 665
5.6.2.5 FFLD Language Example 665
5.6.2.6 IL Language Example 665
5.6.2.7 ST Language Example 665
5.6.3 FileDelete 666
5.6.3.1 Inputs 666
5.6.3.2 Outputs 666
5.6.3.3 Remarks 666
5.6.3.4 FBD Language Example 667
5.6.3.5 FFLD Language Example 667
5.6.3.6 IL Language Example 667
5.6.3.7 ST Language Example 667
5.6.4 FileEOF 668
5.6.4.1 Inputs 668
5.6.4.2 Outputs 668
5.6.4.3 Remarks 668
5.6.4.4 FBD Language Example 669
5.6.4.5 FFLD Language Example 669
5.6.4.6 IL Language Example 669
5.6.4.7 ST Language Example 669
5.6.5 FileExists 669
5.6.5.1 Inputs 669
5.6.5.2 Outputs 670
5.6.5.3 Remarks 670
5.6.5.4 FBD Language Example 670
5.6.5.5 FFLD Language Example 670

5.6.5.6 IL Language Example 670
5.6.5.7 ST Language Example 670
5.6.6 FileOpenA 671
5.6.6.1 Inputs 671
5.6.6.2 Outputs 671
5.6.6.3 Remarks 671
5.6.6.4 FBD Language Example 672
5.6.6.5 FFLD Language Example 672
5.6.6.6 IL Language Example 672
5.6.6.7 ST Language Example 672
5.6.7 FileOpenR 673
5.6.7.1 Inputs 673
5.6.7.2 Outputs 673
5.6.7.3 Remarks 673
5.6.7.4 FBD Language Example 674
5.6.7.5 FFLD Language Example 674
5.6.7.6 IL Language Example 674
5.6.7.7 ST Language Example 674
5.6.8 FileOpenW 675
5.6.8.1 Inputs 675
5.6.8.2 Outputs 675
5.6.8.3 Remarks 675
5.6.8.4 FBD Language Example 676
5.6.8.5 FFLD Language Example 676
5.6.8.6 IL Language Example 676
5.6.8.7 ST Language Example 676
5.6.9 FileReadBinData 677
5.6.9.1 Inputs 677
5.6.9.2 Outputs 677
5.6.9.3 Remarks 677
5.6.9.4 FBD Language Example 677
5.6.9.5 FFLD Language Example 677
5.6.9.6 IL Language Example 677
5.6.9.7 ST Language Example 677
5.6.10 FileReadLine 678
5.6.10.1 Inputs 678
5.6.10.2 Outputs 678
5.6.10.3 Remarks 679
5.6.10.4 FBD Language Example 679
5.6.10.5 FFLD Language Example 679
5.6.10.6 IL Language Example 679
5.6.10.7 ST Language Example 679
5.6.11 FileRename 680
5.6.11.1 Inputs 680
5.6.11.2 Outputs 680
5.6.11.3 Remarks 680
5.6.11.4 FBD Language Example 681
5.6.11.5 FFLD Language Example 681

5.6.11.6 IL Language Example ... 681
5.6.11.7 ST Language Example ... 681
5.6.12 FileSeek ... 682
5.6.12.1 Inputs ... 682
5.6.12.2 Outputs ... 682
5.6.12.3 Remarks ... 682
5.6.12.4 FBD Language Example ... 682
5.6.12.5 FFLD Language Example ... 682
5.6.12.6 IL Language Example ... 683
5.6.12.7 ST Language Example ... 683
5.6.13 FileSize ... 683
5.6.13.1 Inputs ... 683
5.6.13.2 Outputs ... 683
5.6.13.3 Remarks ... 684
5.6.13.4 FBD Language Example ... 684
5.6.13.5 FFLD Language Example ... 684
5.6.13.6 IL Language Example ... 684
5.6.13.7 ST Language Example ... 684
5.6.14 FileWriteBinData ... 685
5.6.14.1 Inputs ... 685
5.6.14.2 Outputs ... 685
5.6.14.3 Remarks ... 685
5.6.14.4 FBD Language Example ... 686
5.6.14.5 FFLD Language Example ... 686
5.6.14.6 IL Language Example ... 686
5.6.14.7 ST Language Example ... 686
5.6.15 FileWriteLine ... 686
5.6.15.1 Inputs ... 686
5.6.15.2 Outputs ... 687
5.6.15.3 Remarks ... 687
5.6.15.4 FBD Language Example ... 688
5.6.15.5 FFLD Language Example ... 688
5.6.15.6 IL Language Example ... 688
5.6.15.7 ST Language Example ... 688
5.7 TCP/IP Function Blocks ... 688
5.7.1 TcpAccept ... 689
5.7.1.1 Inputs ... 689
5.7.1.2 Outputs ... 689
5.7.1.3 Remarks ... 689
5.7.1.4 FBD Language Example ... 690
5.7.1.5 FFLD Language Example ... 690
5.7.1.6 IL Language Example ... 690
5.7.1.7 ST Language Example ... 690
5.7.2 TcpBinReceive ... 690
5.7.2.1 Inputs ... 691
5.7.2.2 Outputs ... 691
5.7.2.3 Remarks ... 691
5.7.2.4 FBD Language Example ... 691

5.7.2.5 FFLD Language Example 692
5.7.2.6 IL Language Example 692
5.7.2.7 ST Language Example 692
5.7.3 TcpBinSend 692
5.7.3.1 Inputs 692
5.7.3.2 Outputs 692
5.7.3.3 Remarks 693
5.7.3.4 FBD Language Example 693
5.7.3.5 FFLD Language Example 693
5.7.3.6 IL Language Example 693
5.7.3.7 ST Language Example 693
5.7.4 TcpClose 694
5.7.4.1 Inputs 694
5.7.4.2 Outputs 694
5.7.4.3 Remarks 694
5.7.4.4 FBD Language Example 695
5.7.4.5 FFLD Language Example 695
5.7.4.6 IL Language Example 695
5.7.4.7 ST Language Example 695
5.7.5 TcpConnect 695
5.7.5.1 Inputs 696
5.7.5.2 Outputs 696
5.7.5.3 Remarks 696
5.7.5.4 FBD Language Example 696
5.7.5.5 FFLD Language Example 696
5.7.5.6 IL Language Example 696
5.7.5.7 ST Language Example 697
5.7.6 TcpIsConnected 697
5.7.6.1 Inputs 697
5.7.6.2 Outputs 697
5.7.6.3 Remarks 698
5.7.6.4 FBD Language Example 698
5.7.6.5 FFLD Language Example 698
5.7.6.6 IL Language Example 698
5.7.6.7 ST Language Example 698
5.7.7 TcpIsValid 699
5.7.7.1 Inputs 699
5.7.7.2 Outputs 699
5.7.7.3 Remarks 699
5.7.7.4 FBD Language Example 699
5.7.7.5 FFLD Language Example 699
5.7.7.6 IL Language Example 699
5.7.7.7 ST Language Example 700
5.7.8 TcpListen 700
5.7.8.1 Inputs 700
5.7.8.2 Outputs 700
5.7.8.3 Remarks 701
5.7.8.4 FBD Language Example 701

5.7.8.5 FFLD Language Example ... 701
5.7.8.6 IL Language Example ... 701
5.7.8.7 ST Language Example ... 701
5.7.9 TcpSend ... 702
5.7.9.1 Inputs ... 702
5.7.9.2 Outputs ... 702
5.7.9.3 Remarks ... 702
5.7.9.4 FBD Language Example ... 703
5.7.9.5 FFLD Language Example ... 703
5.7.9.6 IL Language Example ... 703
5.7.9.7 ST Language Example ... 703
5.8 UDP Functions for Controller and Simulator ... 703
5.8.1 UDP Functions ... 704
5.8.2 udpAddrMake ... 704
5.8.2.1 Inputs ... 704
5.8.2.2 Outputs ... 704
5.8.2.3 Remarks ... 705
5.8.2.4 FBD Language Example ... 705
5.8.2.5 FFLD Language Example ... 705
5.8.2.6 IL Language Example ... 705
5.8.2.7 ST Language Example ... 705
5.8.3 udpClose ... 705
5.8.3.1 Inputs ... 705
5.8.3.2 Outputs ... 705
5.8.3.3 Remarks ... 705
5.8.3.4 FBD Language Example ... 705
5.8.3.5 FFLD Language Example ... 706
5.8.3.6 IL Language Example ... 706
5.8.3.7 ST Language Example ... 706
5.8.4 udpCreate ... 706
5.8.4.1 Inputs ... 706
5.8.4.2 Outputs ... 706
5.8.4.3 Remarks ... 706
5.8.4.4 FBD Language Example ... 706
5.8.4.5 FFLD Language Example ... 706
5.8.4.6 IL Language Example ... 707
5.8.4.7 ST Language Example ... 707
5.8.5 udpIsValid ... 707
5.8.5.1 Inputs ... 707
5.8.5.2 Outputs ... 707
5.8.5.3 Remarks ... 707
5.8.5.4 FBD Language Example ... 707
5.8.5.5 FFLD Language Example ... 707
5.8.5.6 IL Language Example ... 707
5.8.5.7 ST Language Example ... 707
5.8.6 udpRcvFrom ... 708
5.8.6.1 Inputs ... 708
5.8.6.2 Outputs ... 708

5.8.6.3 Remarks ... 708
5.8.6.4 FBD Language Example ... 708
5.8.6.5 FFLD Language Example ... 708
5.8.6.6 IL Language Example ... 708
5.8.6.7 ST Language Example ... 709
5.8.7 udpRcvFromArray ... 709
5.8.7.1 Inputs ... 709
5.8.7.2 Outputs ... 709
5.8.7.3 Remarks ... 709
5.8.7.4 FBD Language Example ... 709
5.8.7.5 FFLD Language Example ... 709
5.8.7.6 IL Language Example ... 710
5.8.7.7 ST Language Example ... 710
5.8.8 udpRcvFromVar ... 710
5.8.8.1 Inputs ... 710
5.8.8.2 Outputs ... 710
5.8.8.3 Remarks ... 710
5.8.8.4 FBD Language Example ... 711
5.8.8.5 FFLD Language Example ... 711
5.8.8.6 IL Language Example ... 711
5.8.8.7 ST Language Example ... 712
5.8.9 udpSendTo ... 712
5.8.9.1 Inputs ... 712
5.8.9.2 Outputs ... 712
5.8.9.3 Remarks ... 712
5.8.9.4 FBD Language Example ... 712
5.8.9.5 FFLD Language Example ... 712
5.8.9.6 IL Language Example ... 712
5.8.9.7 ST Language Example ... 713
5.8.10 udpSendToArray ... 713
5.8.10.1 Inputs ... 713
5.8.10.2 Outputs ... 713
5.8.10.3 Remarks ... 713
5.8.10.4 FBD Language Example ... 713
5.8.10.5 FFLD Language Example ... 713
5.8.10.6 IL Language Example ... 713
5.8.10.7 ST Language Example ... 714
5.8.11 udpSendToVar ... 714
5.8.11.1 Inputs ... 714
5.8.11.2 Outputs ... 714
5.8.11.3 Remarks ... 714
5.8.11.4 FBD Language Example ... 715
5.8.11.5 FFLD Language Example ... 715
5.8.11.6 IL Language Example ... 715
5.8.11.7 ST Language Example ... 715
5.9 PrintMessage ... 715
5.9.1 Inputs ... 716
5.9.2 Outputs ... 716

5.9.3 Remarks ... 716
5.9.3.1 Source ... 716
5.9.3.2 Level ... 716
5.9.3.3 Usage ... 716
5.9.4 FBD Language Example ... 717
5.9.5 FFLD Language Example ... 717
5.9.6 IL Language Example ... 717
5.9.7 ST Language Example ... 717
5.10 File and TCP/IP Function Block ErrorIDs ... 717
5.10.1 File ErrorIDs ... 718
5.10.2 TCP/IP ErrorIDs ... 718
Appendix A: Index ... 719
6 Copyrights, Licenses, and Trademarks ... 726
6.1 Copyrights ... 726
6.2 Trademarks ... 726
6.3 PCMM2G ... 727
6.4 Disclaimer ... 727
7 Support and Services ... 728

2 Fieldbus Library

These tables list the Fieldbus functions and function blocks:

- "EtherCAT Library" (→ p. 23)
- "EtherNet/IP (ODVA)" (→ p. 23)

2.1 EtherCAT Library

Name	Description
DriveParamRead	Reads a drive parameter by sending an ASCII command to a drive.
DriveParamStrRead	Reads a single drive parameter by sending an ASCII command to a drive.
DriveParamWrite	Writes a drive parameter by sending an ASCII command to a drive.
ECATCommErrors	Returns a list of bad EtherCAT connections.
ECATDeviceAction	Performs an action on an EtherCAT device.
ECATDeviceStatus	Provides the EtherCAT state and the port link status information for the EtherCAT device.
ECATDevReadParam	Returns the EtherCAT device-specific information.
ECATGetObjVal	Deprecated.
ECATMasterStatus	Reads the EtherCAT master state and the lost frame counter to determine if EtherCAT is running normally.
ECATReadData	Reads cyclic parameter (byte offset format).
ECATReadSDO	Reads a 32-bit word from I/O nodes using a CANopen SDO read command.
ECATWCStatus	Returns the current number of working counter errors for the Sync unit.
ECATWriteData	Writes cyclic parameter (byte offset format).
ECATWriteSDO	Writes a 32-bit word to I/O nodes using a CANopen SDO write command.
FSoEParamsInit	Transfers safety parameters from the safety master to safety slave devices to initialize the safety network.

2.2 EtherNet/IP (ODVA)

Name	Description
eipAdapter	Provides information about the current state of the Scanner connection.
eipReadAttr	Sends an explicit message (UCMM) to an EtherNet/IP adapter for reading a single CIP attribute.
eipWriteAttr	Sends an explicit message (UCMM) to an EtherNet/IP adapter for reading a single CIP attribute.

2.3 Fieldbus - EtherCAT

Name	Description
DriveParamRead	Reads a drive parameter by sending an ASCII command to a drive.

Name	Description
DriveParamStrRead	Reads a single drive parameter by sending an ASCII command to a drive.
DriveParamWrite	Writes a drive parameter by sending an ASCII command to a drive.
ECATCommErrors	Returns a list of bad EtherCAT connections.
ECATDeviceAction	Performs an action on an EtherCAT device.
ECATDeviceStatus	Provides the EtherCAT state and the port link status information for the EtherCAT device.
ECATDevReadParam	Returns the EtherCAT device-specific information.
ECATGetObjVal	Deprecated.
ECATMasterStatus	Reads the EtherCAT master state and the lost frame counter to determine if EtherCAT is running normally.
ECATReadData	Reads cyclic parameter (byte offset format).
ECATReadSDO	Reads a 32-bit word from I/O nodes using a CANopen SDO read command.
ECATWCStatus	Returns the current number of working counter errors for the Sync unit.
ECATWriteData	Writes cyclic parameter (byte offset format).
ECATWriteSDO	Writes a 32-bit word to I/O nodes using a CANopen SDO write command.
FSoEParamsInit	Transfers safety parameters from the safety master to safety slave devices to initialize the safety network.

2.3.1 Remarks

The EtherCAT SDO function blocks are activated by the CANopen over EtherCAT (CoE) protocol in a client/server mode.

- The client (aka EtherCAT master) is the KAS Runtime application.
- The servers (aka EtherCAT slaves) are the drives and I/O nodes where data can be retrieved.
- The SDO function blocks only support the reading and writing of 32-bit values.
  - It is the fundamental size of CANopen SDO calls.

2.3.1.1 Why use ECATReadSdo and ECATWriteSdo FBs?

The "ECATReadSDO" (→ p. 54) and "ECATWriteSDO" (→ p. 61) response time is faster.

They are preferred over "DriveParamRead" (→ p. 26) and "DriveParamWrite" (→ p. 34).

2.3.1.2 Why use the DriveParam FBs?

Reasons to prefer the DriveParam FBs are:

- They allow direct use of the parameter name.
  - Example: IL.LIMITP instead of the SDO index: 356Eh
- They can be used to setup a drive terminal in the HMI application.
  - This is similar to the Terminal view available in the AKD widget embedded in the KAS-IDE.

2.3.2 EtherCAT Function Blocks that Work with Drive Parameters

These function blocks are used to work with drive parameters not supported by ML and MC function blocks.

- They support reading and writing drive parameters using the non-cyclic SDO channel in the EtherCAT network.
- The ASCII name for the parameter is used as an input.

2.3.2.1 Execution Time

These function blocks typically take a longer time to execute (up to ten cycles to finish executing).
It takes the same amount of time to Read or Write a parameter.

NOTE

It takes more than one cycle to execute these function blocks (but less than 100 ms).

2.3.2.1.1 Reason

It is not only linked to the ASCII SDO communication. Because these FBs are waiting for the AKD drive to responds, the execution time can also increase due to the load of the AKD firmware at the time you call them.

2.3.2.1.2 Result

The PLC code is overrunning the cycle duration. as explained in Tasking Model / Scheduling.
As a consequence, you can see this message in the Controller Log window:

> The Virtual Machine missed 1 cycle(s) of PLC execution.

2.3.2.1.3 Solution

When this happens we recommend:

- Use these function blocks sparingly in programs
- Rely on the EtherCAT read/write SDO function blocks whenever possible
- Smooth the load of the PLC code by executing these function blocks at the required update rate.

Stats about the FB Execution Time

- There is a small difference in timing when running EtherCAT at 2ms compared to other frequencies.

	0.25, 0.5, 1ms	2ms
Mean	9ms	14ms
Min	3ms	8ms
Max	16ms	24ms

- **Max** time to consider when executing a single SDO command, (i.e., before the Done output becomes TRUE): **24ms**.
- **Max** time to consider when executing a single Drive Parameter command (i.e. before the Done output becomes TRUE): **60 ms**.

	4 kHz	1 kHz
Mean	**20 ms**	**11 ms**
Min	15 ms	9 ms
Max	45 ms	58 ms

- When sending multiple commands to a single drive, only one command can be sent at a time.
- The time to execute multiple commands is:
  Number of commands x Execution time of a single command
- When commands are sent to different AKD drives at the same time, the requests do not interfere with each other.
- The function finishes execution in the same maximum time as one drive.

2.3.3 EtherCAT Function Blocks that Work with SDOs

"ECATReadSDO" (→ p. 54) and "ECATWriteSDO" (→ p. 61) are used to work with drive or remote I/O parameters that are not supported by ML and MC function blocks.

Drive or remote I/O parameters that have an associated SDO number can be read and written using these function blocks.

NOTE

It takes more than one cycle to execute these function blocks (but less than 100 ms).

Stats about the FB Execution Time

- There is a small difference in timing when running EtherCAT at 2ms compared to other frequencies.

	0.25, 0.5, 1ms	2ms
Mean	9ms	14ms
Min	3ms	8ms
Max	16ms	24ms

- **Max** time to consider when executing a single SDO command, (i.e., before the Done output becomes TRUE): **24ms**.
- **Max** time to consider when executing a single Drive Parameter command (i.e. before the Done output becomes TRUE): **60 ms**.

	4 kHz	1 kHz
Mean	**20 ms**	**11 ms**
Min	15 ms	9 ms
Max	45 ms	58 ms

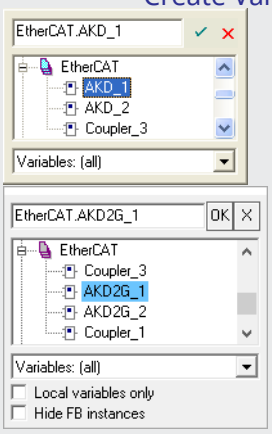
- When sending multiple commands to a single drive, only one command can be sent at a time.
- The time to execute multiple commands is:
  Number of commands x Execution time of a single command
  - When commands are sent to different AKD drives at the same time, the requests do not interfere with each other.
- The function finishes execution in the same maximum time as one drive.

2.3.4 DriveParamRead

PLCopen ✓ Pipe Network ✓

Function Block - Reads a drive parameter by sending an ASCII command to a drive.

2.3.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge of Execute, a drive parameter is read. • The function block processes only one request at a time. • If Execute is toggled quickly so that another rising edge occurs before the function block has completed, the function block does not issue a second read command.
Drive	INT	No range	N/A	No default	The address of the drive where data is read. • The first node usually has the value 1001. • The second node usually has the value 1002. • Alternately, use the members of the EtherCAT structure to specify a device's address to Create Variables. 
Param	STRING	No range	N/A	No default	The parameter to read.

2.3.4.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates whether this function block has completed **without** error.
Error	BOOL	No range	N/A	Indicates whether this function block has completed **with** error.
ErrorID	DINT	No range	N/A	The **DriveParamRead** error result if Error is TRUE. • See the "EtherCAT Error Codes" (→ p. 28) for more information. • Upon success, Error is set to 0 (zero).
Value	LREAL	No range	N/A	The value of the drive parameter. Value is only set when the function block has successfully completed.
Units	STRING	No range	N/A	The value of the drive parameter. Value is only set when the function block has successfully completed.

2.3.4.3 Remarks

NOTE

This function block **uses and reserves** the EtherCAT SDO Channel.
The SDO Channel remains reserved until the done output is TRUE.
This FB should be called at each cycle until the done output is TRUE.
If it is not called at each cycle, the rest of SDO communication (e.g., the AKD GUI Views) is blocked.
Using this FB in SFC P0 or P1 steps is not recommended because these steps are executed only once.
If this FB is used in P0 or P1, then it must be used in an SFC N step to ensure the FB completes.

2.3.4.3.1 Usage

Use this function block to read drive parameters that are not supported by other function blocks.

Examples include:

- Drive bus voltage
- Drive display
- Fault history
- Motor temperature
- Present drive limit settings
- Present regeneration loading

- It takes multiple cycles to complete this function block.
- Typically only **one DriveParamRead** or **DriveParamWrite** function should be active for **each axis** at one time.
- If executing this function block continuously or if multiple times is required, add code that waits for this function block to complete (e.g., Done bit = 1) before executing it again.
  - See **stats** about the "Execution Time" (→ p. 25).
- See "EtherCAT Function Blocks that Work with Drive Parameters" (→ p. 24) for information about function blocks not supported by ML and MC function blocks.

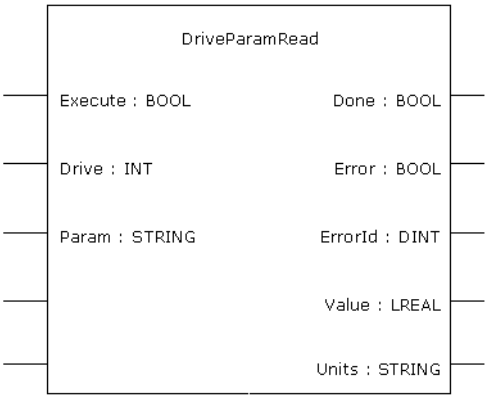


Figure 2-1: DriveParamRead

2.3.4.3.2 EtherCAT Error Codes

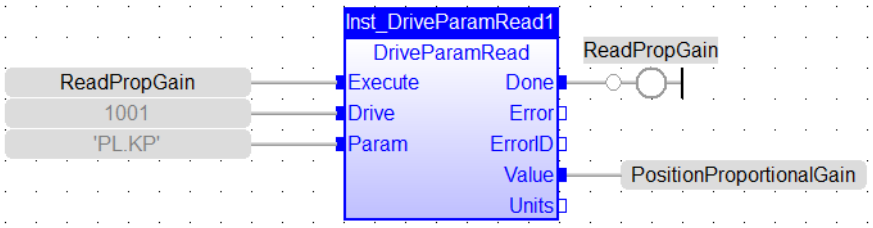
Error Codes List

Value Dec (hex)	Error Code	Description
0	ECERR_OK	The SDO call succeeded.
1792 (0x700)	ECERR_DEVICE_ERROR	EtherCAT device is not accessible.
1794 (0x702)	ECERR_DEVICE_INVALIDCMD	Invalid mailbox command.
1795 (0x703)	ECERR_DEVICE_INVALIDINDEX	An invalid value for the Index input was specified.

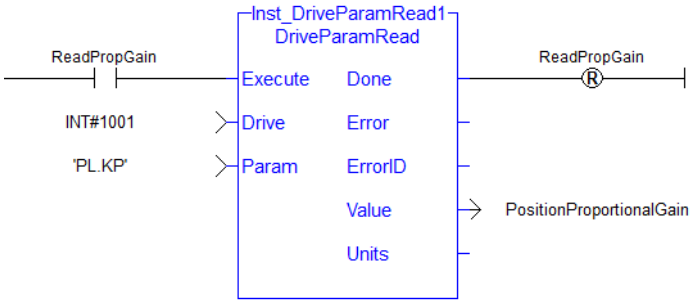
Value Dec (hex)	Error Code	Description
1796 (0x704)	ECERR_DEVICE_ INVALIDACCESS	Reading of the variable is not permitted.
1797 (0x705)	ECERR_DEVICE_INVALIDSIZE	An invalid size for the parameter was specified.
1798 (0x706)	ECERR_DEVICE_INVALIDDATA	Invalid parameter value(s) in SDO index and/or sub-index.
1799 (0x707)	ECERR_DEVICE_NOTREADY	Device is not in a ready state. The network is not in operational.
1800 (0x708)	ECERR_DEVICE_BUSY	Device is not available to respond.
1801 (0x709)	ECERR_DEVICE_ INVALIDCONTEXT	Device responded with an illegal error code, indicating the command is not allowed under the present conditions.
1802 (0x70A)	ECERR_DEVICE_NOMEMORY	EtherCAT mailbox is out of memory or device is out of disk space.
1803 (0x70B)	ECERR_DEVICE_INVALIDPARM	EtherCAT mailbox request was not valid.
1804 (0x70C)	ECERR_DEVICE_NOTFOUND	EtherCAT device not found.
1805 (0x70D)	ECERR_DEVICE_SYNTAX	An unexpected error occurred.
1810 (0x712)	ECERR_DEVICE_ INVALIDSTATE	The EtherCAT device is in an invalid state.
1817 (0x719)	ECERR_DEVICE_TIMEOUT	The EtherCAT device failed to respond, timing out.
1826 (0x722)	ECERR_DEVICE_ INSERTMAILBOX	Error while inserting the mailbox command into internal FIFO.
1827 (0x723)	ECERR_DEVICE_ INVALIDOFFSET	An invalid value for the SubIndex input was specified.
1828 (0x724)	ECERR_DEVICE_ UNKNOWNMAILBOXCMD	The master sent an unknown mailbox command to the slave.
1829 (0x725)	ECERR_DEVICE_ ACCESSDENIED	Device responded with an invalid access error code, indicating the command is not allowed.
1832 (0x728)	ECERR_DEVICE_INVALIDADDR	The specified EtherCAT node address is invalid.
1836 (0x72c)	ECERR_DEVICE_NOT_A_ FSOE_MASTER	Device is not a FSoE master.
1920 (0x780)	ECERR_DEVICE_PARAM_ ACCESS_ERROR	Unknown error occurred while accessing parameter.
1921 (0x781)	ECERR_DEVICE_PARAM_NOT_ FOUND	Parameter was not found.
1922 (0x782)	ECERR_DEVICE_PARAM_NOT_ INTEGER	Parameter is a floating-point value. Integer value required.
1923 (0x783)	ECERR_DEVICE_VALUE_IS_ NEGATIVE	No negative values allowed. Value specified was negative.

Value Dec (hex)	Error Code	Description
1924 (0x784)	ECERR_DEVICE_VALUE_OUT_OF_RANGE	Value is out of data-range.
1925 (0x785)	ECERR_DEVICE_VALUE_GREATER_THAN_MAX	Value bigger than maximum.
1926 (0x786)	ECERR_DEVICE_VALUE_LOWER_THAN_MIN	Value lower than minimum.
2048 (0x800)	ECERR_CLIENT_ERROR	Error in Mailbox response to a previously sent mailbox command.
2049 (0x801)	ECERR_CLIENT_TIMEOUT	The SDO command timed out.
2050 (0x802)	ECERR_CLIENT_ INVALIDPARM	An invalid value was specified.
2051 (0x803)	ECERR_CLIENT_ INVALIDSIZE	An invalid value for the size input was specified.

2.3.4.4 FBD Language Example



2.3.4.5 FFLD Language Example



2.3.4.6 IL Language Example

Not available.

2.3.4.7 ST Language Example

```
(* Read PL.KP on first AKD Drive on EtherCAT network *)

(* The code continually calls the FB (without re-executing it) until the
first execution is done, then reads the returned value from the drive and
reset the FB *)

IF ReadPropGain then
   Inst_DriveParamRead1( 1, 1001, 'PL.KP' );
End_If;

On Inst_DriveParamRead1.Done do
```

```
   Inst_DriveParamRead1( 0, 1001, 'PL.KP' );
   PositionProportionalGain := Inst_DriveParamRead1.Value; (* Reads the
returned value from the drive *)
   ReadPropGain := 0; (* Reset the FB *)
End_DO;
```

See Also

"DriveParamWrite" (→ p. 34)

2.3.5 DriveParamStrRead

PLCopen ✓ Pipe Network ✓

Function Block - Reads a single drive parameter by sending an ASCII command to a drive.

2.3.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	Executes the function block.
Drive	INT	No range	N/A	No default	The address of the drive from which data is read.
Param	STRING	No range	N/A	No default	The parameter to read.

2.3.5.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates whether this function block has completed **without** error.
Error	BOOL	No range	N/A	Indicates whether this function block has completed **with** error.
ErrorID	DINT	No range	N/A	The **DriveParamStrRead** error result if Error is TRUE. Upon success, Error is set to 0 (zero). See the "EtherCAT Error Codes" (→ p. 32) for more information.
Value	STRING	No range	N/A	The value of the drive parameter. Value is only set when the function block has successfully completed.

2.3.5.3 Remarks

NOTE

This function block **uses and reserves** the EtherCAT SDO Channel.
The SDO Channel remains reserved until the done output is TRUE.
This FB should be called at each cycle until the done output is TRUE.
If it is not called at each cycle, the rest of SDO communication (e.g., the AKD GUI Views) is blocked.
Using this FB in SFC P0 or P1 steps is not recommended because these steps are executed only once.
If this FB is used in P0 or P1, then it must be used in an SFC N step to ensure the FB completes.

- The value returned is the string response from the drive.
- See "EtherCAT Function Blocks that Work with Drive Parameters" (→ p. 24) for information about function blocks not supported by ML and MC function blocks.

NOTE

This differs from "DriveParamRead" (→ p. 26) because the drive response is not parsed.
DriveParamRead parses the drive response and returns the numeric value of the parameter and the units found that represent that parameter.
Since **DriveParamStrRead** returns the drive response directly, it can be used to read parameters that have string representations (e.g., the AKD2G's AXIS#.FAULTMSG# parameters).

2.3.5.3.1 Usage

Use this function block to read drive parameters that are not supported by other function blocks.

Examples include:

- Drive bus voltage
- Drive display
- Fault history
- Motor temperature
- Present drive limit settings
- Present regeneration loading

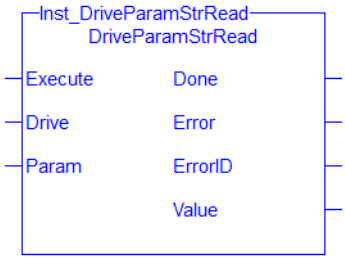


Figure 2-2: DriveParamStrRead

2.3.5.3.2 EtherCAT Error Codes

Error Codes List

Value Dec (hex)	Error Code	Description
0	ECERR_OK	The SDO call succeeded.
1792 (0x700)	ECERR_DEVICE_ERROR	EtherCAT device is not accessible.
1794 (0x702)	ECERR_DEVICE_INVALIDCMD	Invalid mailbox command.
1795 (0x703)	ECERR_DEVICE_INVALIDINDEX	An invalid value for the Index input was specified.
1796 (0x704)	ECERR_DEVICE_ INVALIDACCESS	Reading of the variable is not permitted.
1797 (0x705)	ECERR_DEVICE_INVALIDSIZE	An invalid size for the parameter was specified.
1798 (0x706)	ECERR_DEVICE_INVALIDDATA	Invalid parameter value(s) in SDO index and/or sub-index.
1799 (0x707)	ECERR_DEVICE_NOTREADY	Device is not in a ready state. The network is not in operational.
1800 (0x708)	ECERR_DEVICE_BUSY	Device is not available to respond.
1801 (0x709)	ECERR_DEVICE_ INVALIDCONTEXT	Device responded with an illegal error code, indicating the command is not allowed under the present conditions.

Value Dec (hex)	Error Code	Description
1802 (0x70A)	ECERR_DEVICE_NOMEMORY	EtherCAT mailbox is out of memory or device is out of disk space.
1803 (0x70B)	ECERR_DEVICE_INVALIDPARM	EtherCAT mailbox request was not valid.
1804 (0x70C)	ECERR_DEVICE_NOTFOUND	EtherCAT device not found.
1805 (0x70D)	ECERR_DEVICE_SYNTAX	An unexpected error occurred.
1810 (0x712)	ECERR_DEVICE_ INVALIDSTATE	The EtherCAT device is in an invalid state.
1817 (0x719)	ECERR_DEVICE_TIMEOUT	The EtherCAT device failed to respond, timing out.
1826 (0x722)	ECERR_DEVICE_ INSERTMAILBOX	Error while inserting the mailbox command into internal FIFO.
1827 (0x723)	ECERR_DEVICE_ INVALIDOFFSET	An invalid value for the SubIndex input was specified.
1828 (0x724)	ECERR_DEVICE_ UNKNOWNMAILBOXCMD	The master sent an unknown mailbox command to the slave.
1829 (0x725)	ECERR_DEVICE_ ACCESSDENIED	Device responded with an invalid access error code, indicating the command is not allowed.
1832 (0x728)	ECERR_DEVICE_INVALIDADDR	The specified EtherCAT node address is invalid.
1836 (0x72c)	ECERR_DEVICE_NOT_A_ FSOE_MASTER	Device is not a FSoE master.
1920 (0x780)	ECERR_DEVICE_PARAM_ ACCESS_ERROR	Unknown error occurred while accessing parameter.
1921 (0x781)	ECERR_DEVICE_PARAM_NOT_ FOUND	Parameter was not found.
1922 (0x782)	ECERR_DEVICE_PARAM_NOT_ INTEGER	Parameter is a floating-point value. Integer value required.
1923 (0x783)	ECERR_DEVICE_VALUE_IS_ NEGATIVE	No negative values allowed. Value specified was negative.
1924 (0x784)	ECERR_DEVICE_VALUE_OUT_ OF_RANGE	Value is out of data-range.
1925 (0x785)	ECERR_DEVICE_VALUE_ GREATER_THAN_MAX	Value bigger than maximum.
1926 (0x786)	ECERR_DEVICE_VALUE_ LOWER_THAN_MIN	Value lower than minimum.
2048 (0x800)	ECERR_CLIENT_ERROR	Error in Mailbox response to a previously sent mailbox command.
2049 (0x801)	ECERR_CLIENT_TIMEOUT	The SDO command timed out.
2050 (0x802)	ECERR_CLIENT_ INVALIDPARM	An invalid value was specified.
2051 (0x803)	ECERR_CLIENT_ INVALIDSIZE	An invalid value for the size input was specified.

2.3.5.4 FBD Language Example

Not available.

2.3.5.5 FFLD Language Example

Not available.

2.3.5.6 IL Language Example

Not available.

2.3.5.7 ST Language Example

```
            (* Read AXIS1.FAULTMSG1 on first AKD2G Drive on EtherCAT network
*)

(* The code continually calls the FB (without re-executing it) until the
first execution is done, then reads the returned value from the drive and
reset the FB *)

IF ReadFaultMsg Then
   Inst_DriveParamStrRead1(True, 1001, 'AXIS1.FAULTMSG1' );
End_If;

On Inst_DriveParamStrRead1 Do
   FaultMsg := Inst_DriveParamStrRead1.Value; (* Reads the returned value
from the drive *)
   Inst_DriveParamStrRead1(False, 1001, 'AXIS1.FAULTMSG1');
   ReadFaultMsg := False; (* Reset the FB *)
End_DO;
```

See Also

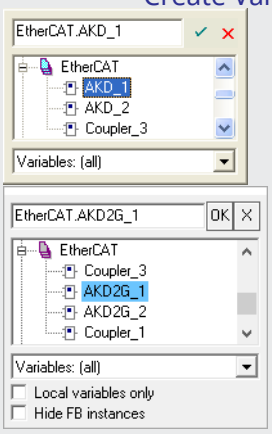
- "DriveParamRead" (→ p. 26)
- "DriveParamWrite" (→ p. 34)

2.3.6 DriveParamWrite

PLCopen ✓ Pipe Network ✓

Function Block - Writes a drive parameter by sending an ASCII command to a drive.

2.3.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge of Execute, a drive parameter is set. • The function block processes only one request at a time. • If Execute is toggled quickly so that another rising edge occurs before the function block has completed, the function block does not issue a second write command.
Drive	INT	No range	N/A	No default	The address of the drive where data is written. • The first node usually has the value 1001. • The second node usually has the value 1002. • Alternately, use the members of the EtherCAT structure to specify a device's address to Create Variables. 
Param	STRING	No range	N/A	No default	The parameter to write.
Value	LREAL	No range	N/A	No default	The value to set the drive parameter to.

2.3.6.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates whether this function block has completed **without** error.
Error	BOOL	No range	N/A	Indicates whether this function block has completed **with** error.
ErrorID	DINT	No range	N/A	The **DriveParamWrite** error result if Error is TRUE. Upon success, Error is set to 0 (zero). See the "EtherCAT Error Codes" (➔ p. 36) for more information.

2.3.6.3 Remarks

NOTE

This function block **uses and reserves** the EtherCAT SDO Channel.
The SDO Channel remains reserved until the done output is TRUE.
This FB should be called at each cycle until the done output is TRUE.
If it is not called at each cycle, the rest of SDO communication (e.g., the AKD GUI Views) is blocked.

Using this FB in SFC P0 or P1 steps is not recommended because these steps are executed only once. If this FB is used in P0 or P1, then it must be used in an SFC N step to ensure the FB completes.

- Use this function block to change drive parameters.
  Examples include:
  - Tuning parameters.
  - Changing drive limits (i.e., peak current).
- It takes multiple cycles to complete this function block.
- Typically only **one DriveParamRead** or **DriveParamWrite** function should be active for **each axis** at one time.
- If executing this function block continuously or if multiple times is required, add code that waits for this function block to complete (e.g., Done bit = 1) before executing it again.
  - See **stats** about the "Execution Time" (→ p. 25).
- See "EtherCAT Function Blocks that Work with Drive Parameters" (→ p. 24) for information about function blocks not supported by ML and MC function blocks.

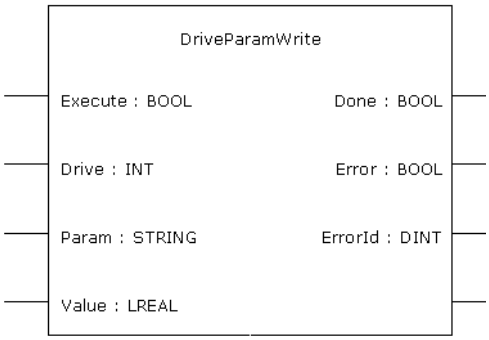


Figure 2-3: DriveParamWrite

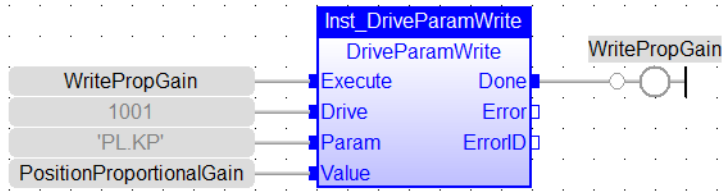
2.3.6.3.1 EtherCAT Error Codes

Error Codes List

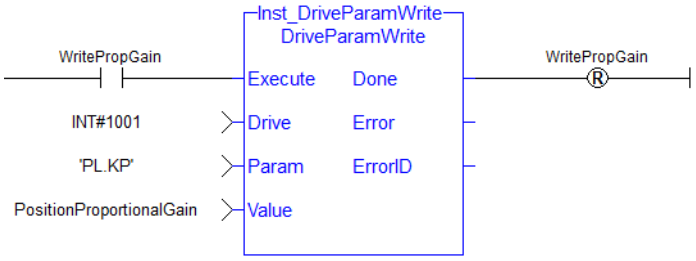
Value Dec (hex)	Error Code	Description
0	ECERR_OK	The SDO call succeeded.
1792 (0x700)	ECERR_DEVICE_ERROR	EtherCAT device is not accessible.
1794 (0x702)	ECERR_DEVICE_INVALIDCMD	Invalid mailbox command.
1795 (0x703)	ECERR_DEVICE_INVALIDINDEX	An invalid value for the Index input was specified.
1796 (0x704)	ECERR_DEVICE_ INVALIDACCESS	Reading of the variable is not permitted.
1797 (0x705)	ECERR_DEVICE_INVALIDSIZE	An invalid size for the parameter was specified.
1798 (0x706)	ECERR_DEVICE_INVALIDDATA	Invalid parameter value(s) in SDO index and/or sub-index.
1799 (0x707)	ECERR_DEVICE_NOTREADY	Device is not in a ready state. The network is not in operational.
1800 (0x708)	ECERR_DEVICE_BUSY	Device is not available to respond.
1801 (0x709)	ECERR_DEVICE_ INVALIDCONTEXT	Device responded with an illegal error code, indicating the command is not allowed under the present conditions.

Value Dec (hex)	Error Code	Description
1802 (0x70A)	ECERR_DEVICE_NOMEMORY	EtherCAT mailbox is out of memory or device is out of disk space.
1803 (0x70B)	ECERR_DEVICE_INVALIDPARM	EtherCAT mailbox request was not valid.
1804 (0x70C)	ECERR_DEVICE_NOTFOUND	EtherCAT device not found.
1805 (0x70D)	ECERR_DEVICE_SYNTAX	An unexpected error occurred.
1810 (0x712)	ECERR_DEVICE_ INVALIDSTATE	The EtherCAT device is in an invalid state.
1817 (0x719)	ECERR_DEVICE_TIMEOUT	The EtherCAT device failed to respond, timing out.
1826 (0x722)	ECERR_DEVICE_ INSERTMAILBOX	Error while inserting the mailbox command into internal FIFO.
1827 (0x723)	ECERR_DEVICE_ INVALIDOFFSET	An invalid value for the SubIndex input was specified.
1828 (0x724)	ECERR_DEVICE_ UNKNOWNMAILBOXCMD	The master sent an unknown mailbox command to the slave.
1829 (0x725)	ECERR_DEVICE_ ACCESSDENIED	Device responded with an invalid access error code, indicating the command is not allowed.
1832 (0x728)	ECERR_DEVICE_INVALIDADDR	The specified EtherCAT node address is invalid.
1836 (0x72c)	ECERR_DEVICE_NOT_A_ FSOE_MASTER	Device is not a FSoE master.
1920 (0x780)	ECERR_DEVICE_PARAM_ ACCESS_ERROR	Unknown error occurred while accessing parameter.
1921 (0x781)	ECERR_DEVICE_PARAM_NOT_ FOUND	Parameter was not found.
1922 (0x782)	ECERR_DEVICE_PARAM_NOT_ INTEGER	Parameter is a floating-point value. Integer value required.
1923 (0x783)	ECERR_DEVICE_VALUE_IS_ NEGATIVE	No negative values allowed. Value specified was negative.
1924 (0x784)	ECERR_DEVICE_VALUE_OUT_ OF_RANGE	Value is out of data-range.
1925 (0x785)	ECERR_DEVICE_VALUE_ GREATER_THAN_MAX	Value bigger than maximum.
1926 (0x786)	ECERR_DEVICE_VALUE_ LOWER_THAN_MIN	Value lower than minimum.
2048 (0x800)	ECERR_CLIENT_ERROR	Error in Mailbox response to a previously sent mailbox command.
2049 (0x801)	ECERR_CLIENT_TIMEOUT	The SDO command timed out.
2050 (0x802)	ECERR_CLIENT_ INVALIDPARM	An invalid value was specified.
2051 (0x803)	ECERR_CLIENT_ INVALIDSIZE	An invalid value for the size input was specified.

2.3.6.4 FBD Language Example



2.3.6.5 FFLD Language Example



2.3.6.6 IL Language Example

Not available.

2.3.6.7 ST Language Example

```
      (* Write 58.000 to PL.KP of first AKD Dri
      ve on Ethe
      rCAT network *)
Inst_DriveParamWrite( TRUE, 1001, 'PL.KP', 58 );
```

See Also

"DriveParamRead" (→ p. 26)

2.3.7 ECATCommErrors

PLCopen

Function Block - Returns a list of bad EtherCAT connections.

2.3.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	Read the communication errors on the rising edge.
Connection	ECATCommErr_ref See the "ECATCommErr_ref Structure" (→ p. 39) table.	N= 0 to 2 times the number of EtherCAT devices.	N/A	No default	Array of bad connections. The safe size for the list is: [2 * (number of devices) - 1].

2.3.7.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates whether this function block has completed **without** error.
Error	BOOL	No range	N/A	Indicates whether this function block has completed **with** error.
ErrorID	DINT	No range	N/A	The function call error result, if Error is TRUE. • See the "Possible Error Codes and Descriptions" (→ p. 40) table. • Upon success, Error is set to 0 (zero).
ConnectionCount	UINT	No range	N/A	The number of bad connections. Valid indices in the Connection array range from 0 (zero) to ConnectionCount -1 (assuming ConnectionCount != 0).

2.3.7.3 Remarks

If EtherCAT network communication is shutdown, the failed connections are based on information that was taken at the time the network was shutdown.

TIP

See Check the Connections for Errors for an example of implementing this function.

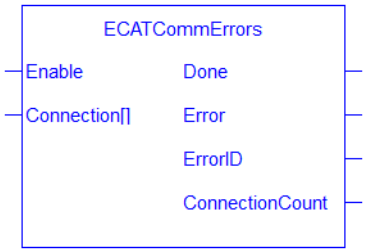


Figure 2-4: ECATCommErrors Function Block

2.3.7.3.1 ECATCommErr_ref Structure

Parameter	Type	Description
CommErrorCounter	UINT	The Communication Error Counter for this port.
ConnectedSlaveAddress	INT	The EtherCAT address of the connected device.
ConnectedSlavePortID	UINT	The port number of the connected device.
LostLinkCounter	UINT	The Lost Link Counter for this port.
SlaveAddress	INT	The EtherCAT address of the device that owns the port.
SlavePortID	UINT	The port number.

2.3.7.3.2 EtherCAT Port Numbers Defines

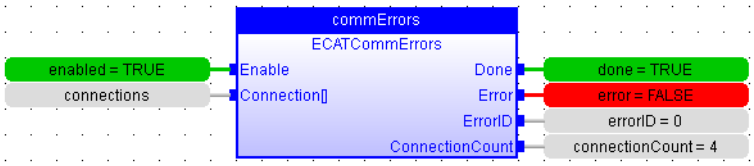
Define	Port
#define EC_PORT_A	0 (* Port A *)
#define EC_PORT_B	1 (* Port B *)
#define EC_PORT_C	2 (* Port C *)
#define EC_PORT_D	3 (* Port D*)

2.3.7.3.3 Possible Error Codes and Descriptions

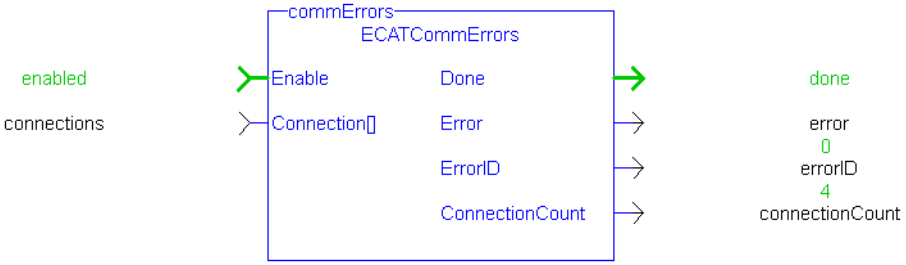
Error Code	Description
ECERR_DEVICE_ERROR	The EtherCAT driver is in a bad state.
ECERR_INVALID_ARRAY_SIZE	The size of the Connections is too small.

- When the array size is smaller than the number of bad connections:
  - The array is filled with the data to the size of the array.
  - The error ECERR_INVALID_ARRAY_SIZE is set.
- In this scenario, the output **ConnectionCount** is set to the size of the array and it is smaller than the number of actual bad connections.

2.3.7.4 FBD Language Example



2.3.7.5 FFLD Language Example



2.3.7.6 IL Language Example

Not available.

2.3.7.7 ST Language Example

```
(*******************************************************)
Read EtherCAT communication errors.
(*******************************************************)
commErrors( TRUE, Connection);
```

See Also

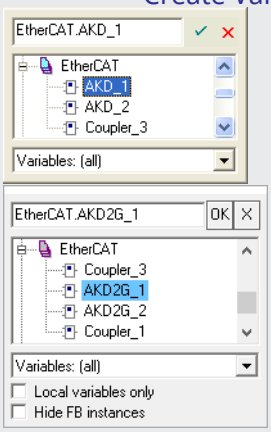
- "ECATDeviceStatus" (→ p. 43)
- "ECATMasterStatus" (→ p. 51)

2.3.8 ECATDeviceAction

PLCopen ✓ Pipe Network ✓

Function Block - Performs an action on an EtherCAT device.

2.3.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	When Execute is set to TRUE, an action is performed.
DeviceAddress	INT	No range	N/A	No default	The address of the device to perform an action on. • The first node usually has the value 1001. • The second node usually has the value 1002. • Alternately, use the members of the EtherCAT structure to specify a device's address to Create Variables. 

Action	INT	No range	N/A	No default	The action to be performed. Action can be one of these values:
					• DEVICE_ACTION_CONNECT
					  • Connects a slave into the EtherCAT network.
					  • This action can only be executed when the motion engine is stopped.
					• DEVICE_ACTION_DISCONNECT
					  • Disconnects a slave from the EtherCAT network.
					  • This action can only be executed when the motion engine is stopped.

2.3.8.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates whether this function block has completed **without** error.
Error	BOOL	No range	N/A	Indicates whether this function block has completed **with** error.
ErrorID	DINT	No range	N/A	The **ECATDeviceAction** error result if Error is TRUE. • See "Error Codes" (→ p. 42). • Upon success, Error is set to 0 (zero).

2.3.8.3 Remarks

- This function returns immediately.
- It can be called multiple times in one cycle.

2.3.8.3.1 Supported Actions

2.3.8.3.1.1 DEVICE_ACTION_DISCONNECT

- The EtherCAT master is notified to expect the node to be removed from the EtherCAT when ECATDeviceAction is called with DEVICE_ACTION_DISCONNECT.
- Any Axis/Axes mapped to a drive node is automatically simulated.
- The last position from the physical drive is carried over to the simulation when the EtherCAT restarts.

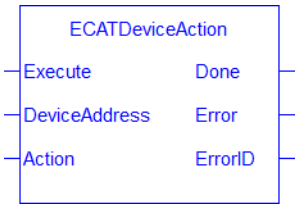
2.3.8.3.1.2 DEVICE_ACTION_CONNECT

- This action connects the already disconnected node to the EtherCAT Network.
- The axis (or axes) that was acting as simulated axis becomes a normal axis when the drive node is connected.
- The axis/axes position comes directly from the configured drive feedback.
  - The position is **not** automatically transferred from the simulated axis.

2.3.8.3.2 Usage

The EtherCAT network needs to be stopped to use this function block.

- This is achieved by either:
  - Calling **ECATDeviceAction** prior to calling "MLMotionStart" (→ p. 552).
  - First calling "MLMotionStop" (→ p. 556) to stop the network.
- See Modular EtherCAT Concept for more information on modular machine design.

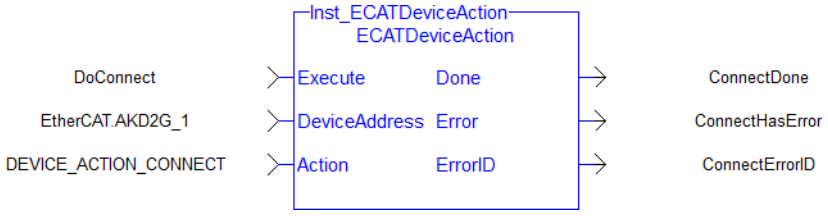


2.3.8.4 Error Codes

Value Dec (hex)	Error Code	Description
0	ECERR_OK	The action succeeded.
1000 (0x3E8)	ECERR_DEVICE_ACTION_IS_INVALID	The specified device action is invalid.
1001 (0x3E9)	ECERR_DEVICE_IS_DC_CLOCK_ MASTER	The Device is a DC Clock Master.
1002 (0x3EA)	ECERR_DEVICE_IS_IN_DC_MASTER_ UPSTREAM	The device is before the DC Clock Master.
1003 (0x3EB)	ECERR_DEVICE_ACTION_MINIMUM_ ONE_NODE_REQD	A minimum of one node is required in the EtherCAT network.
1792 (0x700)	ECERR_DEVICE_ERROR	EtherCAT device is not accessible.
1799 (0x707)	ECERR_DEVICE_NOTREADY	Device is not in a ready state. The EtherCAT network is not in the desired state for the specified action.
1832 (0x728)	ECERR_DEVICE_INVALIDADDR	The specified EtherCAT node address is invalid.

2.3.8.5 FBD Language Example

2.3.8.6 FFLD Language Example



2.3.8.7 IL Language Example

Not available.

2.3.8.8 ST Language Example

```
MLMotionStop();
MotionEngineStatus := MLMotionStatus();
On MotionEngineStatus = MLSTATUS_STOPPED Do
   Inst_ECATDeviceAction(True, EtherCAT.AKD_3, DEVICE_ACTION_DISCONNECT);
   If Inst_ECATDeviceAction.Error Then
       // Handle Error
       // Error ID value is in Inst_ECATDeviceAction.Error
   End_If;
End_Do;
```

See Also

- "MLMotionStart" (→ p. 552)
- "MLMotionStop" (→ p. 556)

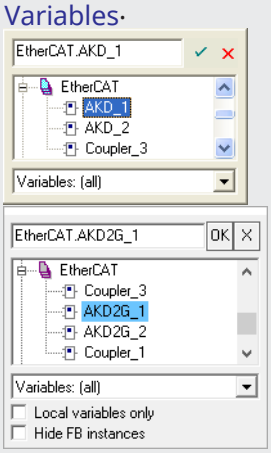
2.3.9 ECATDeviceStatus

PLCopen ✓ Pipe Network ✓

Function Block - Provides the EtherCAT state and the port link status information for the EtherCAT device.

2.3.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	Read the device status on the rising edge.

Input	Data Type	Range	Unit	Default	Description
DeviceAddress	INT	No range	N/A	No default	The address of the device data is read from. • The first node usually has the value 1001. • The second node usually has the value 1002. • Alternately, use the members of the EtherCAT structure to specify a device's address to Create Variables. 

2.3.9.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates whether this function block has completed **without** error.
State	UINT	No range	N/A	Indicates the EtherCAT state of the device. • See "State Defines" (→ p. 45) for more information. • A value of 0 (zero) indicates there is no communication with the device and the state is unknown. • Bits 3:0 indicate the actual state of the device. • An EC_STATE_ERROR (bit 4 set to 1) indicates the device is not in the EtherCAT master requested state due to error conditions (e.g., loss of communication).
LinkStatus	UINT	No range	N/A	Provides the physical link status of the device's ports. • See "LinkStatus Defines" (→ p. 45) for more information. • If no communication is possible with the device, bit 0 is set to 1. • If a link is detected on a port (A-D), the corresponding bit (4-7) is set to 1. • If no link is detected the corresponding bit is set to 0.
Error	BOOL	No range	N/A	Indicates whether this function block has completed **with** error.

Output	Data Type	Range	Unit	Description
ErrorID	DINT	No range	N/A	The function call error result, if Error is TRUE. • Upon success, Error is set to 0 (zero).

2.3.9.3 Remarks

- If the EtherCAT network communication is not running due to a shutdown, the device status contains information taken at the time the network was shutdown.
- This function block is useful in locating the devices with communication errors when the "ECATWCStatus" (→ p. 58) function indicates there are EtherCAT communication errors.

TIP

See Check the Device States in EtherCAT Communication Diagnosis Steps for an example of implementing this function.

2.3.9.3.1 State Defines

```
#define   EC_STATE_NO_COMMUNICATION   0   (* 0x00 = No Communication to
device *)
#define   EC_STATE_INIT               1   (* 0x01 = Device in Init state *)
#define   EC_STATE_PREOP              2   (* 0x02 = Device in Pre-operational
state *)
#define   EC_STATE_BOOTSTRAP          3   (* 0x03 = Device in Bootstrap state
*)
#define   EC_STATE_SAFEOP             4   (* 0x04 = Device in Safe-
Operational state *)
#define   EC_STATE_OP                 8   (* 0x08 = Device in Operational
state *)
#define   EC_STATE_ERROR              16  (* 0x10 bit 4 set to 1; Device not
in requested state error *)
```

2.3.9.3.2 LinkStatus Defines

```
#define   EC_LINK_NO_COMMUNICATION  1   (* 0x1 = No communication to device;
bit 0 set to 1 *)
#define   EC_LINK_PORT_A            16  (* 0x10 = Link detected on Port A;
bit 4 set to 1 *)
#define   EC_LINK_PORT_B            32  (* 0x20 = Link detected on Port B;
bit 5 set to 1 *)
#define   EC_LINK_PORT_C            64  (* 0x40 = Link detected on Port C;
bit 6 set to 1 *)
#define   EC_LINK_PORT_D            128 (* 0x80 = Link detected on Port D;
bit 7 set to 1 *)
```

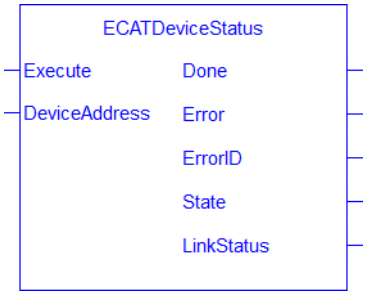
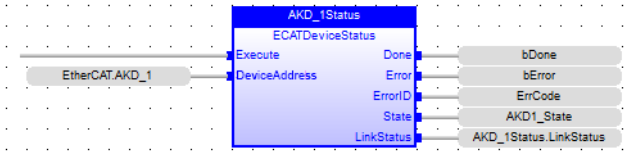
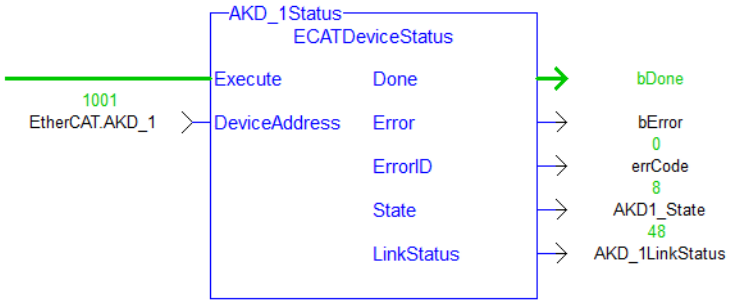


Figure 2-5: ECATDeviceStatus

2.3.9.4 FBD Language Example



2.3.9.5 FFLD Language Example



2.3.9.6 IL Language Example

Not available.

2.3.9.7 ST Language Example

```
(*******************************************************)
(* Read AKD_1 device state and link status*)
(*******************************************************)

Inst_EcDeviceStatus(TRUE,EtherCAT.AKD_1);
```

See Also

- "ECATMasterStatus" (→ p. 51)
- "ECATWCStatus" (→ p. 58)

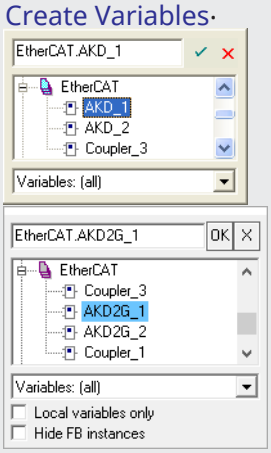
2.3.10 ECATDevReadParam

PLCopen ✓ Pipe Network ✓

Function Block - Returns the EtherCAT device-specific information.

2.3.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	Requests to read the EtherCAT device-specifc parameter.

Input	Data Type	Range	Unit	Default	Description
DeviceAddress	INT	No range	N/A	No default	The address of the device data is read from. • The first node usually has the value 1001. • The second node usually has the value 1002. • Alternately, use the members of the EtherCAT structure to specify a device's address to Create Variables. 
ParameterNumber	INT	No range	N/A	No default	Parameter number. See "EtherCAT Device Parameters" (→ p. 48) for more information.

2.3.10.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates whether this function block has completed **without** error.
Error	BOOL	No range	N/A	Indicates an invalid input.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE. See "Error Codes" (→ p. 47) for more information.
Value	LREAL	No range	N/A	Value of the parameter.

2.3.10.3 Remarks

See "EtherCAT Function Blocks that Work with Drive Parameters" (→ p. 24) for information about function blocks not supported by ML and MC function blocks.

2.3.10.3.1 Error Codes

Value Dec (hex)	Error Code	Description
0	ECERR_OK	No Error
1792 (0x700)	ECERR_DEVICE_ERROR	EtherCAT device is not accessible.

Value Dec (hex)	Error Code	Description
1832 (0x728)	ECERR_DEVICE_INVALIDADDR	The specified EtherCAT node address is invalid.
1921 (0x781	ECERR_DEVICE_PARAM_NOT_FOUND	Parameter was not found.

2.3.10.3.2 EtherCAT Device Parameters

This is a list of supported parameters read by **ECATDevParamRead**.

<table>
<tr><th>Parameter</th><th>ID</th><th>Name</th><th>R/W</th><th>Description</th></tr>
<tr><td>DEVICE_PARAM_DEVICE_TYPE</td><td>1</td><td>Device Type</td><td>Read Only</td><td>EtherCAT Device Type
<table>
<tr><th>Output Value</th><th>Numerical Value</th><th>Description</th></tr>
<tr><td>DEVICE_TYPE_OTHER</td><td>0</td><td>The EtherCAT device is not a drive.</td></tr>
<tr><td>DEVICE_TYPE_DRIVE</td><td>1</td><td>The EtherCAT device is a drive.</td></tr>
</table></td></tr>
<tr><td>DEVICE_PARAM_DRIVE_FAMILY</td><td>2</td><td>Drive Family</td><td>Read Only</td><td>EtherCAT device drive family details.
<table>
<tr><th>Output Value</th><th>Numerical Value</th><th>Description</th></tr>
<tr><td>DEVICE_NOT_A_DRIVE</td><td>-1</td><td>The EtherCAT device is not a drive.</td></tr>
<tr><td>DRIVE_FAMILY_OTHER</td><td>0</td><td>The drive family cannot be determined.</td></tr>
<tr><td>DRIVE_FAMILY_S300_S700</td><td>1</td><td>The device is an S300/S700 family drive.</td></tr>
<tr><td>DRIVE_FAMILY_AKD</td><td>2</td><td>The device is in the AKD2G, AKD-N, or AKD servo drive family.</td></tr>
<tr><td>DRIVE_FAMILY_AKT2G_STEPPER</td><td>3</td><td>The device is an AKT2G Stepper family drive.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Parameter</th><th>ID</th><th>Name</th><th>R/W</th><th>Description</th></tr>
<tr><td>DEVICE_PARAM_DRIVE_GEN</td><td>3</td><td>Drive Generation</td><td>Read Only</td><td>EtherCAT drive generation details.
<table>
<tr><th>Output Value</th><th>Numerical Value</th><th>Description</th></tr>
<tr><td>DEVICE_NOT_A_DRIVE</td><td>-1</td><td>The EtherCAT device is not a drive.</td></tr>
<tr><td>DRIVE_GEN_UNKNOWN</td><td>0</td><td>The drive generation cannot be determined.</td></tr>
<tr><td>DRIVE_GEN_1</td><td>1</td><td>The device is a 1st generation drive (e.g., AKD, AKD-N, S300, S700, AKT2G-SM-L15, AKT2G-SM-L50).</td></tr>
<tr><td>DRIVE_GEN_2</td><td>2</td><td>The device is a 2nd generation drive (e.g., AKD2G).</td></tr>
</table>
</td></tr>
</table>

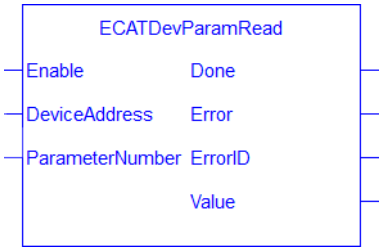
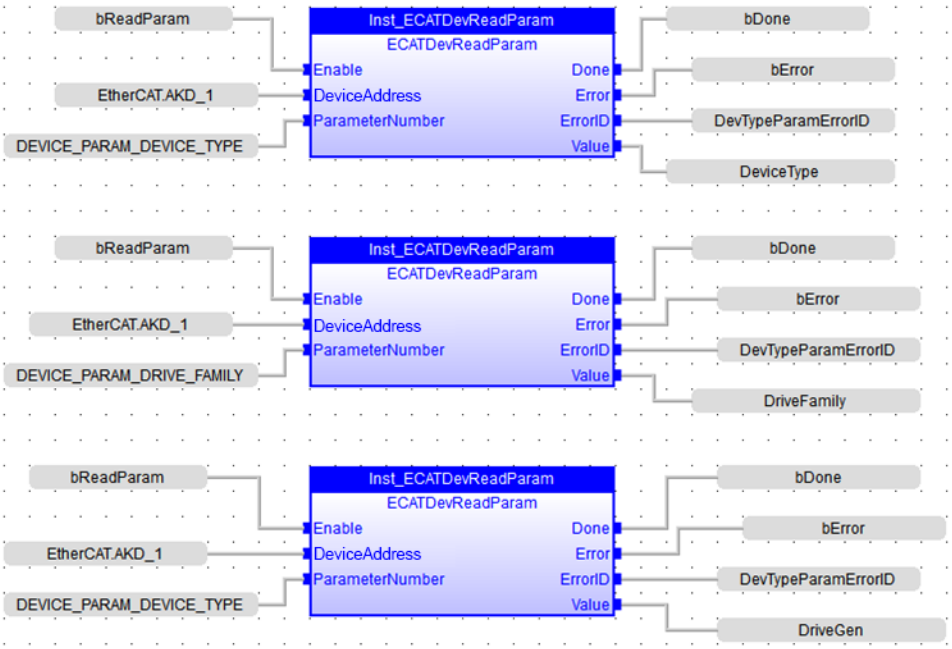
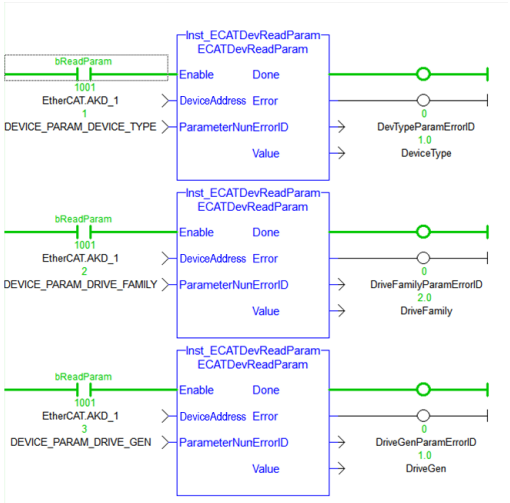


Figure 2-6: ECATDevReadParam

2.3.10.4 FBD Language Example



2.3.10.5 FFLD Language Example



2.3.10.6 IL Language Example

Not available.

2.3.10.7 ST Language Example

```
(* ECATDevReadParam ST example *)
Inst_ECATDevReadParam( TRUE, EtherCAT.AKD_2, DEVICE_PARAM_DEVICE_TYPE );
DeviceType := Inst_ECATDevReadParam.Value;

Inst_ECATDevReadParam( TRUE, EtherCAT.AKD_2, DEVICE_PARAM_DRIVE_FAMILY );
DeviceFamily := Inst_ECATDevReadParam.Value;

Inst_ECATDevReadParam( TRUE, EtherCAT.AKD_2, DEVICE_PARAM_DRIVE_GEN );
DriveGen := Inst_ECATDevReadParam.Value;
```

2.3.11 ECATGetObjVal

PLCopen ✓ Pipe Network ✓

NOTE

Function - Deprecated as of KAS v2.7.
The recommended best practice is to map a PLC variable to a PDO object.

2.3.12 ECATMasterStatus

PLCopen ✓ Pipe Network ✓

Function Block - Reads the EtherCAT master state and the lost frame counter to determine if EtherCAT is running normally.

2.3.12.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	Request to read the EtherCAT master state and the lost frame count. Continuously reads the master state and the lost frame count as long as the Enable remains high.

2.3.12.2 Outputs

Output	Data Type	Range	Unit	Description
Valid	BOOL	No range	N/A	Indicates the values at the **State** and **LostFrameCount** outputs are valid.
Error	BOOL	No range	N/A	Indicates whether this function block has completed **with** error.
ErrorID	DINT	No range	N/A	Error code when the function block failed due to error.
State	UINT	No range	N/A	Indicates the EtherCAT state of the Master. See "State Defines" (→ p. 51) for more information.
LostFrameCount	UDINT	No range	N/A	Total cumulative number of cyclic frames sent with no-response since the EtherCAT started by calling the MLMoitonStart. • Missing return frames will generate an A38 alarm. • The Counter is reset to 0 (zero) when the MLMotionStart is called.

2.3.12.3 Remarks

TIP

See EtherCAT Communication Diagnosis Steps for more information.

2.3.12.3.1 State Defines

```
#define    EC_STATE_NO_COMMUNICATION    0    (* 0x00 = No Communication to
device *)
#define    EC_STATE_INIT                1    (* 0x01 = Device in Init state *)
#define    EC_STATE_PREOP               2    (* 0x02 = Device in Pre-operational
```

```
state *)
#define    EC_STATE_BOOTSTRAP          3    (* 0x03 = Device in Bootstrap state
*)
#define    EC_STATE_SAFEOP             4    (* 0x04 = Device in Safe-
Operational state *)
#define    EC_STATE_OP                 8    (* 0x08 = Device in Operational
state *)
```

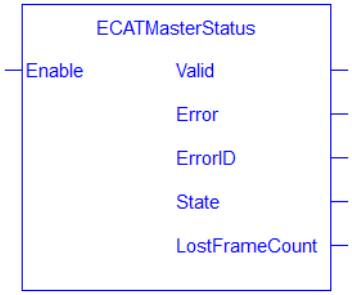
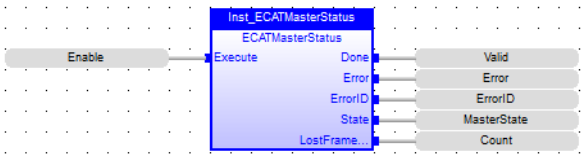
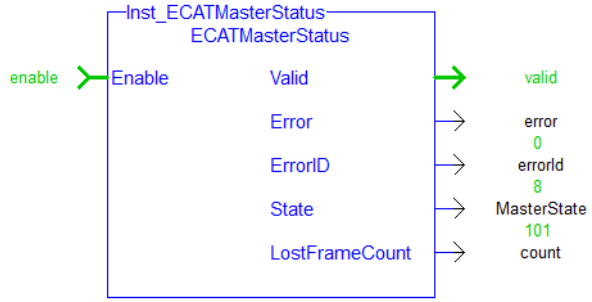


Figure 2-7: ECATMasterStatus

2.3.12.4 FBD Language Example



2.3.12.5 FFLD Language Example



2.3.12.6 IL Language Example

Not available.

2.3.12.7 ST Language Example

```
//  ECATMasterStatus
Inst_ECATMasterStatus( True );
MasterSTate := Inst_ECATMasterStatus.State;
MasterLastFrameCount := Inst_ECATMasterStatus.LostFrameCount;
```

See Also

- "ECATDeviceStatus" (→ p. 43)
- "ECATWCStatus" (→ p. 58)

2.3.13 ECATReadData

PLCopen ✓ Pipe Network ✓

Function - Reads cyclic parameter (byte offset format).

2.3.13.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Offset	UINT	0 to 1500	Bytes	No default	Offset in bytes from the beginning of the frame. The maximum size of an Ethernet frame is 1500. **IMPORTANT** The Offset value required to access may change when: The firmware for any device on the EtherCAT network is updated or Whenever the EtherCAT network topology changes. When performing an update of a network device or changing the network topology, export the ENI file and check the Offset value needed to access the desired information.
Nbytes	SINT	1, 2, or 4	Bytes	No default	Number of bytes to read.
Direction	BOOL	0, 1	N/A	No default	Direction of the frame. • TRUE = output image. • FALSE = input image.

2.3.13.2 Outputs

Output	Data Type	Range	Unit	Description
Value	DINT	No range	N/A	Value of the EtherCAT frame.

2.3.13.3 Remarks

IMPORTANT

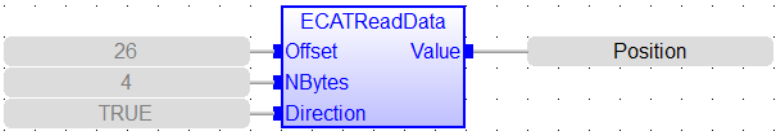
This is a low level function and should be used carefully by **advanced users**.

NOTE

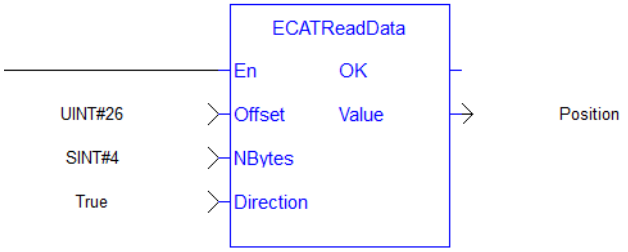
The valid ranges for the **Value** parameter are:
For 1 byte: 0 to 255
For 2 bytes: 0 to 65535
For 4 bytes: -2147483648 to 2147483647 (The sign bit represents the most significant bit in the data word).

- Allows a direct access to the memory EtherCAT Process Image of the EtherCAT frame which is sent or received when you need to debug the application.
- Access the EtherCAT image element by giving the offset in the image and the size of the element.
- If you have a device other than the drive, this function is used for more than just debug.
  - It is used to get the status of the module (e.g., Stepper I/O slice) in the case the project is based on an external XML file because it contains unsupported EtherCAT device.

2.3.13.4 FBD Language Example



2.3.13.5 FFLD Language Example



2.3.13.6 IL Language Example

Not available.

2.3.13.7 ST Language Example

```
// Read 4 bytes starting at offset 26 of the output image
Position := ECATReadData(26, 4, true);
```

See Also

"ECATWriteData" (→ p. 59)

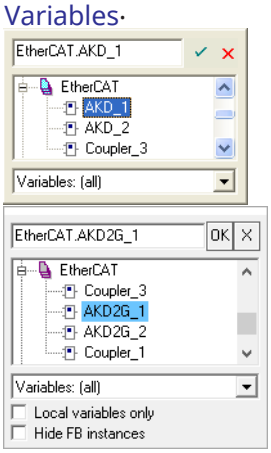
2.3.14 ECATReadSDO

PLCopen ✓ Pipe Network ✓

Function Block - Reads a 32-bit word from I/O nodes using a CANopen SDO read command.

2.3.14.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge of Execute, an SDO read command is issued. • The function block only handles one SDO command at a time. • If Execute is toggled quickly so that another rising edge occurs before the SDO command has completed, the function block does not issue a second SDO command.

Input	Data Type	Range	Unit	Default	Description
Index	INT	No range	N/A	No default	The object directory index of the data to be read. For more information, see: • Communication SDOs • Manufacturer specific SDOs • Profile specific SDOs To read/write an SDO object with an index greater than 16#7FFF (32767), the value must be entered in this form: `any_to_int(index # in hex format)`. Example: `any_to_int(16#8321)`.
SubIndex	SINT	No range	N/A	No default	The sub-index of the object directory variable to be read. For more information, see: • Communication SDOs • Manufacturer specific SDOs • Profile specific SDOs To read/write an SDO object with an index greater than 16#7FFF (32767), the value must be entered in this form: `any_to_int(index # in hex format)`. Example: `any_to_int(16#8321)`.
Size	SINT	1 to 4	N/A	No default	The size (i.e., number of bytes) to write.
DeviceAddress	INT	No range	N/A	No default	The EtherCAT address of the device from which data is written to. • The first node usually has the value 1001. • The second node usually has the value 1002. • Alternately, use the members of the EtherCAT structure to specify a device's address to Create Variables. 

2.3.14.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates whether this function block has completed **without** error.
Error	BOOL	No range	N/A	Indicates whether this function block has completed **with** error.
ErrorID	DINT	No range	N/A	The SDO call error result, if Error is TRUE. • See "ECATReadSDO" (→ p. 54). • Upon success, Error is set to 0 (zero).
Value	DINT	No range	N/A	The value of the object directory variable being read. Value is only set when an SDO read command has successfully completed.

2.3.14.3 Remarks

NOTE

This function block **uses and reserves** the EtherCAT SDO Channel.
The SDO Channel remains reserved until the done output is TRUE.
This FB should be called at each cycle until the done output is TRUE.
If it is not called at each cycle, the rest of SDO communication (e.g., the AKD GUI Views) is blocked.
Using this FB in SFC P0 or P1 steps is not recommended because these steps are executed only once.
If this FB is used in P0 or P1, then it must be used in an SFC N step to ensure the FB completes.

- Is typically used to query the status of inputs.
- See "EtherCAT Function Blocks that Work with SDOs" (→ p. 26) for information about function blocks which are used to work with drive or remote I/O parameters that are not supported by ML and MC function blocks.

2.3.14.3.1 State Diagram

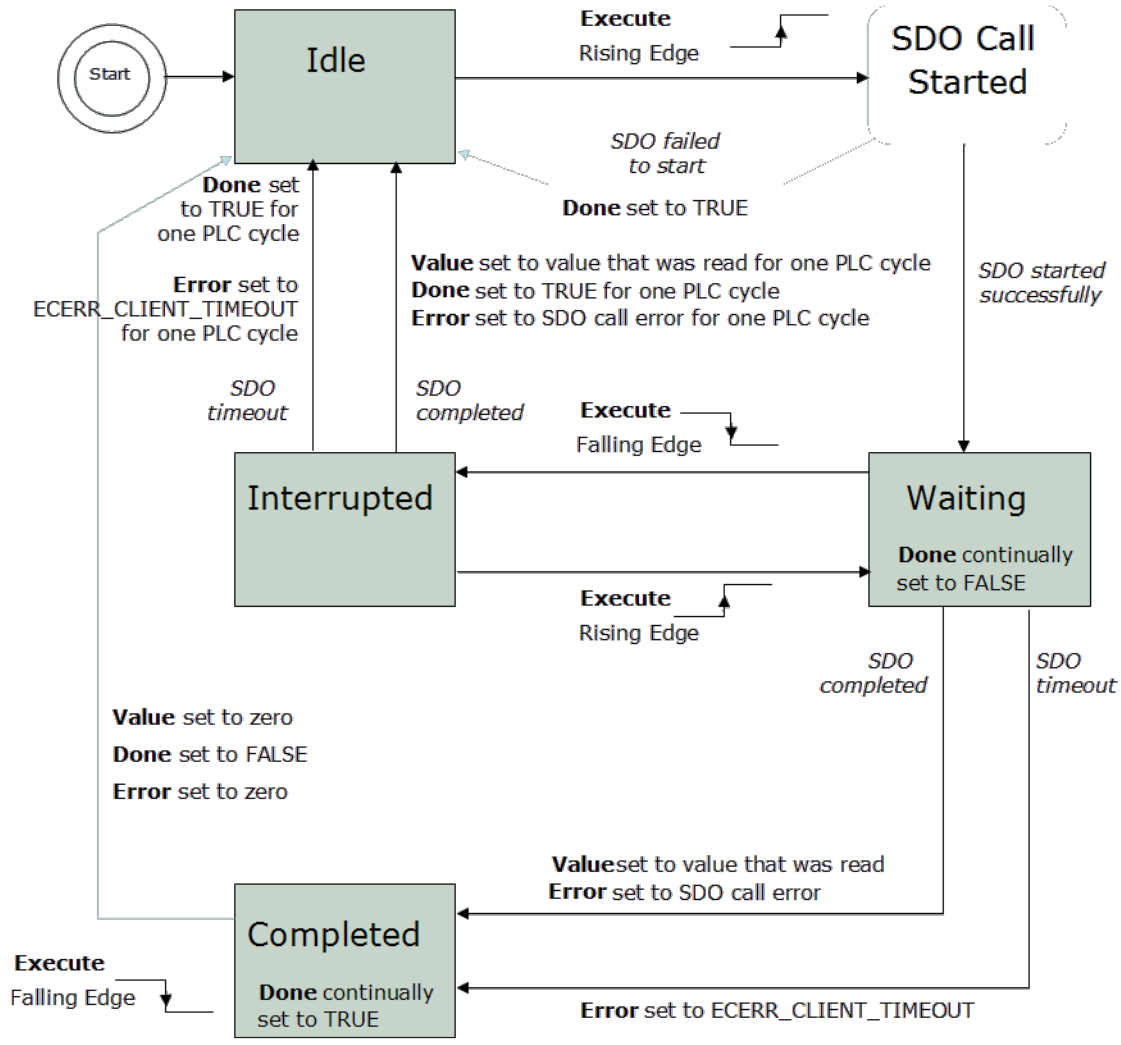


Figure 2-8: ECATReadSDO State Diagram

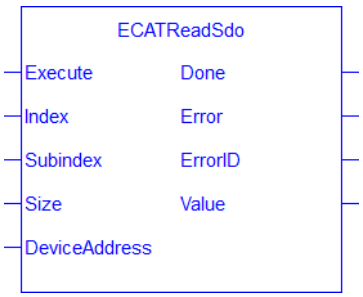
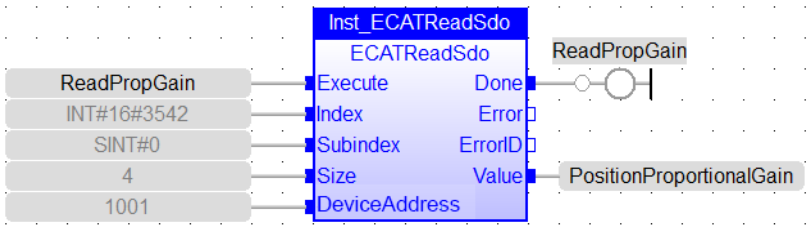
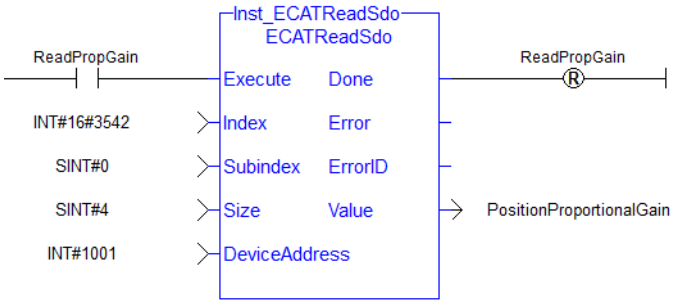


Figure 2-9: ECATReadSDO

2.3.14.4 FBD Language Example



2.3.14.5 FFLD Language Example



2.3.14.6 IL Language Example

Not available.

2.3.14.7 ST Language Example

```
(* Read PL.KP on first AKD Drive on EtherCAT network *)
Inst_ECATReadSdo( TRUE, 16#3542, 0, 4, 1001 );
PositionProportionalGain := Inst_ECATReadSdo.Value;
```

```
(* Read the 4 byte data in SDO index 8321h (33569 decimal), sub-index 1 on
the first AKD Drive
Inst_ECATReadSdo( TRUE, any_to_int(16#8321), 1, 4, 1001 );
ParamValue := Inst_ECATReadSdo.Value;
```

See Also

"ECATWriteSDO" (→ p. 61)

2.3.15 ECATWCStatus

PLCopen ✓ Pipe Network ✓

Function - Returns the current number of working counter errors for the Sync unit.

2.3.15.1 Inputs

Input	Data Type	Range	Unit	Default	Description
SyncUnit	INT	0 (zero)	N/A	No default	Sync Unit Index For future compatibility with multiple frames.

2.3.15.2 Outputs

Output	Data Type	Range	Unit	Description
Count	UDINT	No range	N/A	Working Counter error.

2.3.15.3 Remarks

- The working counter errors are cleared to zero when the EtherCAT network is taken from **Init** to **OP** state.

- Value **0 (zero)**means no working counter errors.
  - When the value is non zero, the master automatically reduces the count by **1** for every thousand good frames received.
- When the working counter error exceeds the Working Counter Error Limit, the EtherCAT network is stopped.

TIP

See Check for Working Counter Errors in EtherCAT Communication Diagnosis Steps for an example of implementing this function.

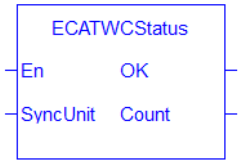
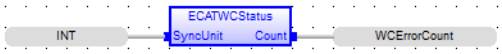
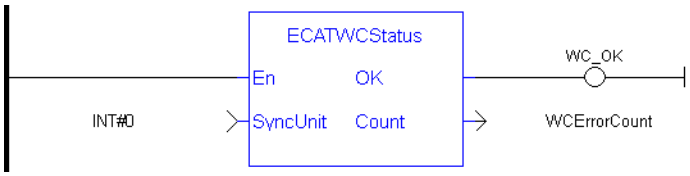


Figure 2-10: ECATWCStatus

2.3.15.4 FBD Language Example



2.3.15.5 FFLD Language Example



2.3.15.6 IL Language Example

Not available.

2.3.15.7 ST Language Example

```
(*******************************************************)
(* read Ethercat Working counter value *)
(*******************************************************)
wcErrorCounter := ECATWCStatus( 0 );
```

See Also

- "ECATDeviceStatus" (→ p. 43)
- "ECATMasterStatus" (→ p. 51)

2.3.16 ECATWriteData

PLCopen ✓ Pipe Network ✓

Function - Writes cyclic parameter (byte offset format).

2.3.16.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Offset	UINT	0 to 1500	Bytes	No default	Offset in bytes from the beginning of the frame. The maximum size of an Ethernet frame is 1500. **IMPORTANT** The Offset value required to access may change when: The firmware for any device on the EtherCAT network is updated or Whenever the EtherCAT network topology changes. When performing an update of a network device or changing the network topology, export the ENI file and check the Offset value needed to access the desired information.
Nbytes	SINT	1, 2, or 4	Bytes	No default	Number of bytes to write.
Value	DINT	-2147483648 to 2147483647	N/A	No default	Value to be written in the image. Only the number of bytes specified by Nbytes is copied.

2.3.16.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	TRUE if the data was written.

2.3.16.3 Remarks

IMPORTANT

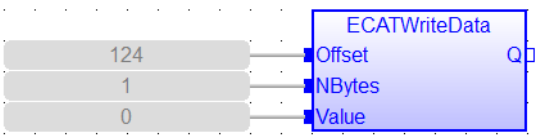
This is a low level function and should be used carefully by **advanced users**.

NOTE

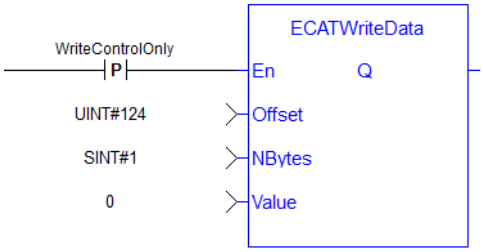
The valid ranges for the **Value** parameter are:
For 1 byte: 0 to 255
For 2 bytes: 0 to 65535
For 4 bytes: -2147483648 to 2147483647 (The sign bit represents the most significant bit in the data word).

- Modify the EtherCAT process image by directly writing values in it.
- If you have a device other than the drive, this function is used for more than just debug.
  - It is used to set the status of the module (e.g., Stepper I/O slice) in the case the project is based on an external XML file because it contains unsupported EtherCAT device.

2.3.16.4 FBD Language Example



2.3.16.5 FFLD Language Example



2.3.16.6 IL Language Example

Not available.

2.3.16.7 ST Language Example

```
//For use with Kollmorgen Thermocouple slice I/O to read in deg C
//Lookup offset by exporting ENI file after EtherCAT network is scanned
//Use offst 124 (byte) to write 0 in control word to allow temperature to be
shown on status byte

ON WriteControlOnly DO
ECATWriteData( 124, 1, 0 );
END_DO
```

See Also

"ECATReadData" (→ p. 52)

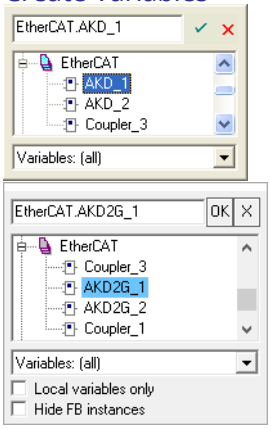
2.3.17 ECATWriteSDO

PLCopen ✓ Pipe Network ✓

Function Block - Writes a 32-bit word to I/O nodes using a CANopen SDO write command.

2.3.17.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge of Execute, an SDO write command is issued. • The function block only handles one SDO command at a time. • If Execute is toggled quickly so that another rising edge occurs before the SDO command has completed, the function block does not issue a second SDO command.

Input	Data Type	Range	Unit	Default	Description
Index	INT	No range	N/A	No default	The object directory index of the data to be written to. For more information, see: • Communication SDOs • Manufacturer specific SDOs • Profile specific SDOs To read/write an SDO object with an index greater than 16#7FFF (32767), the value must be entered in this form: `any_to_int(index # in hex format).` Example: `any_to_int(16#8321).`
SubIndex	SINT	No range	N/A	No default	The sub-index of the object directory variable to be written to. For more information, see: • Communication SDOs • Manufacturer specific SDOs • Profile specific SDOs To read/write an SDO object with an index greater than 16#7FFF (32767), the value must be entered in this form: `any_to_int(index # in hex format).` Example: `any_to_int(16#8321).`
Size	SINT	1 to 4	N/A	No default	The size (i.e., number of bytes) to write.
DeviceAddress	INT	No range	N/A	No default	The EtherCAT address of the device from which data is written to. • The first node usually has the value 1001. • The second node usually has the value 1002. • Alternately, use the members of the EtherCAT structure to specify a device's address to Create Variables. 

Input	Data Type	Range	Unit	Default	Description
Value	DINT	-2147483648 to 2147483647	N/A	No default	The value to write to the object directory variable.

2.3.17.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates whether this function block has completed **without** error.
Error	BOOL	No range	N/A	Indicates whether this function block has completed **with** error.
ErrorID	DINT	No range	N/A	The SDO call error result, if Error is TRUE. • See "ECATWriteSDO" (➔ p. 61). • Upon success, Error is set to 0 (zero).

2.3.17.3 Remarks

NOTE

This function block **uses and reserves** the EtherCAT SDO Channel.
The SDO Channel remains reserved until the done output is TRUE.
This FB should be called at each cycle until the done output is TRUE.
If it is not called at each cycle, the rest of SDO communication (e.g., the AKD GUI Views) is blocked.
Using this FB in SFC P0 or P1 steps is not recommended because these steps are executed only once.
If this FB is used in P0 or P1, then it must be used in an SFC N step to ensure the FB completes.

- See "EtherCAT Function Blocks that Work with SDOs" (➔ p. 26) for information about function blocks which are used to work with drive or remote I/O parameters that are not supported by ML and MC function blocks.

2.3.17.3.1 State Diagram

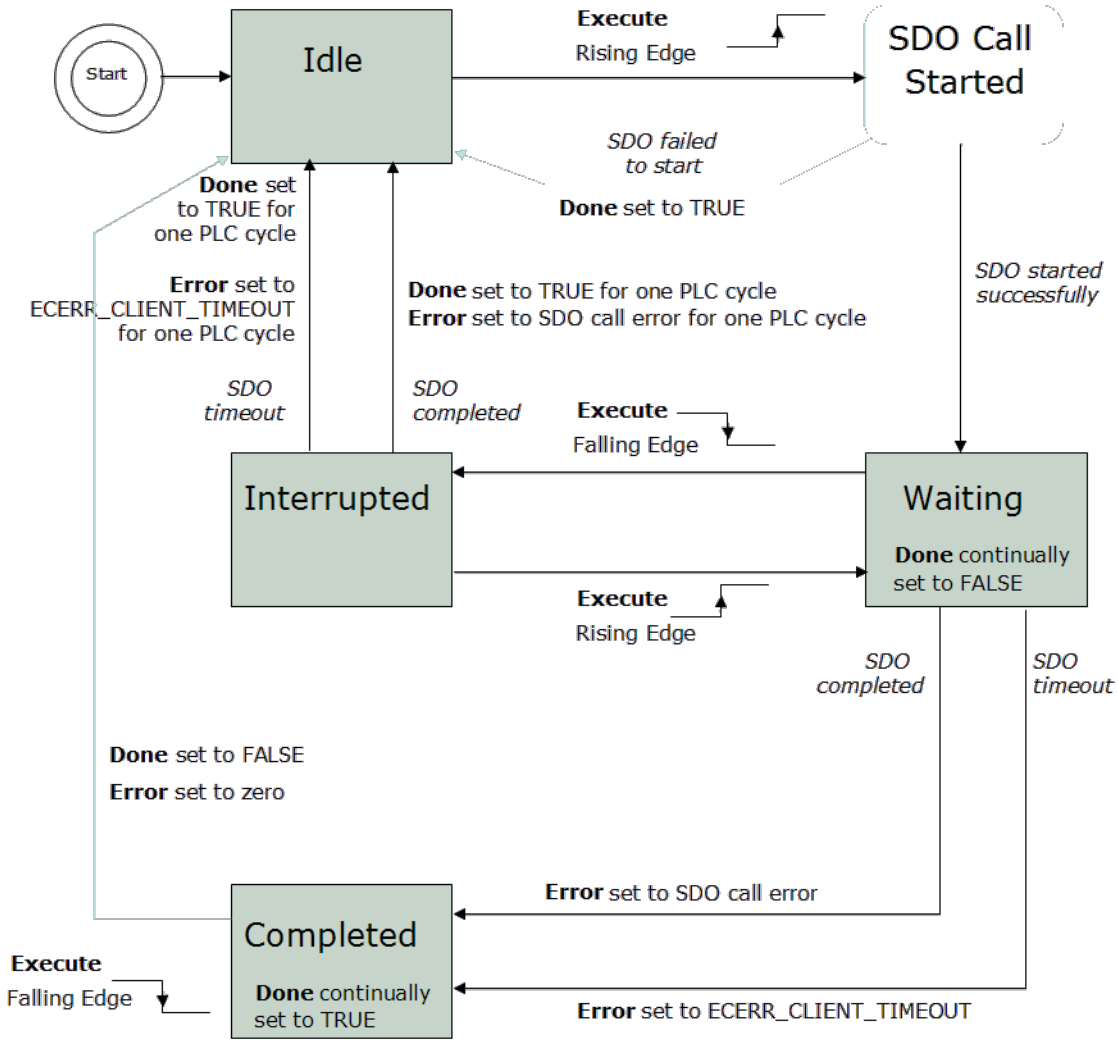


Figure 2-11: ECATWriteSDO State Diagram

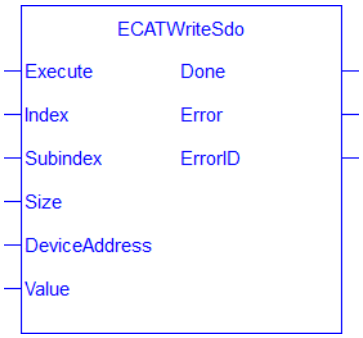
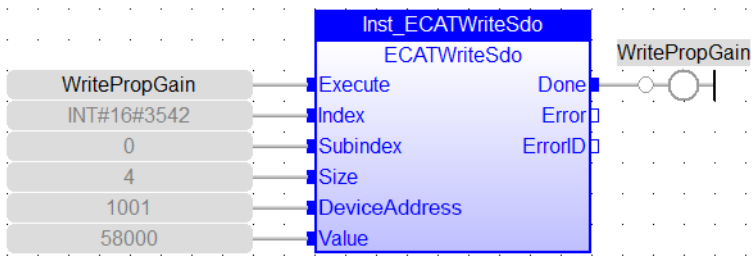
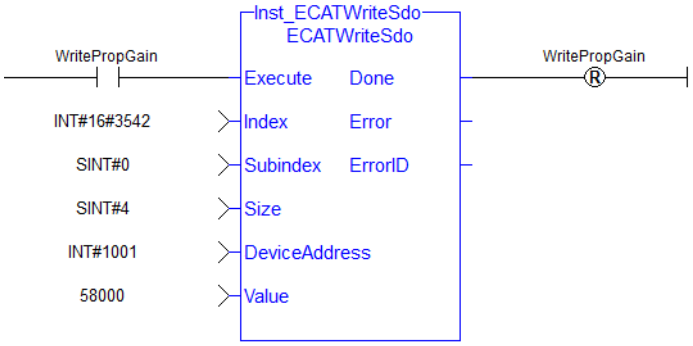


Figure 2-12: ECATWriteSDO

2.3.17.4 FBD Language Example



2.3.17.5 FFLD Language Example



2.3.17.6 IL Language Example

Not available.

2.3.17.7 ST Language Example

```
      (* Write 58.000 to PL.KP of first AKD D
      rive on Ethe
      rCAT network *)
Inst_ECATWriteSdo( TRUE, 16#3542, 0, 4, 1001, 58000 );
```

```
      (* Write a value of 246 to the 4 byte data in SDO index 8321h (33569
decimal), sub-index 1 on the first AKD Drive *)
Inst_ECATWriteSdo( TRUE, any_to_int(16#8321), 1, 4, 1001, 246 );
```

See Also

"ECATReadSDO" (→ p. 54)

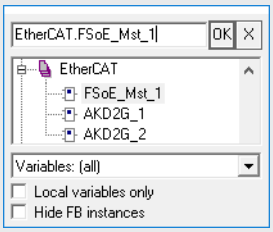
2.3.18 FSoEParamsInit

PLCopen ✓ Pipe Network ✓

Function Block - Transfers safety parameters from the safety master to safety slave devices to initialize the safety network.

2.3.18.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	TRUE, FALSE	N/A	No default	On the rising edge the function block initiates safety parameter transfer from the FSoE master at address `FSoEMasterAddres` to its safety slaves.

Input	Data Type	Range	Unit	Default	Description
FSoE Master's EtherCAT device address	INT	No range	N/A	No default	The EtherCAT slave address of the FSoE master that is asked to initialize safety parameters. • The first EtherCAT slave usually has the value 1001. • The second slave usually has the value 1002. • Alternately, use the members of the EtherCAT structure to specify the EtherCAT address of the safety master. 

2.3.18.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates whether this function block has completed **without** error.
Error	BOOL	No range	N/A	Indicates whether this function block has completed **with** error.
ErrorID	DINT	No range	N/A	The parameter initialization error result. • See the "EtherCAT Error Codes" (→ p. 66) for more information. • Upon success, Error is set to 0 (zero).

2.3.18.3 Remarks

This function block reads the safety parameters from the FSoE master from the input **FSoEMasterAddress** and transfers them to the intended safety slave device using EtherCAT SDO communication.

- The function block:
  - Checks the FSoE master's register for the safety parameter transfer on the rising edge of the **Execute** input.
  - Gets the EtherCAT address of the safety slave that receives the safety parameter if the FSoE master has any safety parameters to transfer.
    - Once the address is read and validated from the FSoE master, the function block reads the actual parameter from the FSoE master and writes the parameter to the safe slave.
- The **Done** output is set to 1 when all parameters to all of the intended Safety slaves are written.
- The **Error** output is set to 1 and the appropriate ErrorID is set in the **ErrorID** output if any error occurred during this process.
- FSoE masters supported by this function block: BBH SCU-1-EC

2.3.18.3.1 EtherCAT Error Codes

Error Codes List

Value Dec (hex)	Error Code	Description
0	ECERR_OK	The SDO call succeeded.
1792 (0x700)	ECERR_DEVICE_ERROR	EtherCAT device is not accessible.
1794 (0x702)	ECERR_DEVICE_INVALIDCMD	Invalid mailbox command.
1795 (0x703)	ECERR_DEVICE_INVALIDINDEX	An invalid value for the Index input was specified.
1796 (0x704)	ECERR_DEVICE_ INVALIDACCESS	Reading of the variable is not permitted.
1797 (0x705)	ECERR_DEVICE_INVALIDSIZE	An invalid size for the parameter was specified.
1798 (0x706)	ECERR_DEVICE_INVALIDDATA	Invalid parameter value(s) in SDO index and/or sub-index.
1799 (0x707)	ECERR_DEVICE_NOTREADY	Device is not in a ready state. The network is not in operational.
1800 (0x708)	ECERR_DEVICE_BUSY	Device is not available to respond.
1801 (0x709)	ECERR_DEVICE_ INVALIDCONTEXT	Device responded with an illegal error code, indicating the command is not allowed under the present conditions.
1802 (0x70A)	ECERR_DEVICE_NOMEMORY	EtherCAT mailbox is out of memory or device is out of disk space.
1803 (0x70B)	ECERR_DEVICE_INVALIDPARM	EtherCAT mailbox request was not valid.
1804 (0x70C)	ECERR_DEVICE_NOTFOUND	EtherCAT device not found.
1805 (0x70D)	ECERR_DEVICE_SYNTAX	An unexpected error occurred.
1810 (0x712)	ECERR_DEVICE_ INVALIDSTATE	The EtherCAT device is in an invalid state.
1817 (0x719)	ECERR_DEVICE_TIMEOUT	The EtherCAT device failed to respond, timing out.
1826 (0x722)	ECERR_DEVICE_ INSERTMAILBOX	Error while inserting the mailbox command into internal FIFO.
1827 (0x723)	ECERR_DEVICE_ INVALIDOFFSET	An invalid value for the SubIndex input was specified.
1828 (0x724)	ECERR_DEVICE_ UNKNOWNMAILBOXCMD	The master sent an unknown mailbox command to the slave.
1829 (0x725)	ECERR_DEVICE_ ACCESSDENIED	Device responded with an invalid access error code, indicating the command is not allowed.
1832 (0x728)	ECERR_DEVICE_INVALIDADDR	The specified EtherCAT node address is invalid.
1836 (0x72c)	ECERR_DEVICE_NOT_A_ FSOE_MASTER	Device is not a FSoE master.
1920 (0x780)	ECERR_DEVICE_PARAM_ ACCESS_ERROR	Unknown error occurred while accessing parameter.

Value Dec (hex)	Error Code	Description
1921 (0x781)	ECERR_DEVICE_PARAM_NOT_FOUND	Parameter was not found.
1922 (0x782)	ECERR_DEVICE_PARAM_NOT_INTEGER	Parameter is a floating-point value. Integer value required.
1923 (0x783)	ECERR_DEVICE_VALUE_IS_NEGATIVE	No negative values allowed. Value specified was negative.
1924 (0x784)	ECERR_DEVICE_VALUE_OUT_OF_RANGE	Value is out of data-range.
1925 (0x785)	ECERR_DEVICE_VALUE_GREATER_THAN_MAX	Value bigger than maximum.
1926 (0x786)	ECERR_DEVICE_VALUE_LOWER_THAN_MIN	Value lower than minimum.
2048 (0x800)	ECERR_CLIENT_ERROR	Error in Mailbox response to a previously sent mailbox command.
2049 (0x801)	ECERR_CLIENT_TIMEOUT	The SDO command timed out.
2050 (0x802)	ECERR_CLIENT_ INVALIDPARM	An invalid value was specified.
2051 (0x803)	ECERR_CLIENT_ INVALIDSIZE	An invalid value for the size input was specified.

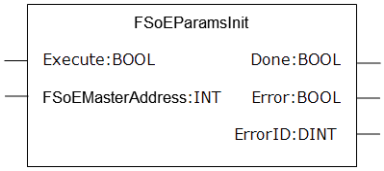
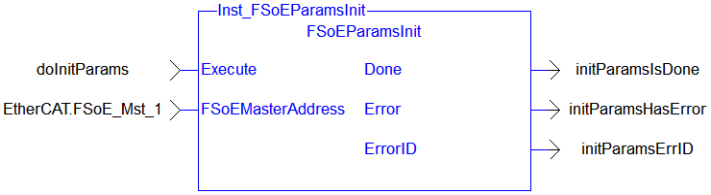


Figure 2-13: FSoEParamsInit

2.3.18.4 FBD Language Example

2.3.18.5 FFLD Language Example



2.3.18.6 IL Language Example

Not available.

2.3.18.7 ST Language Example

```
Inst_FSoEParamsInit(True, EtherCAT.FSoE_Mst_1);
```

See Also

- Set Up FSoE Master and an AKD2G with SafeMotion Monitor
- AKD2G Safety Parametrization Using FSoE with SCU-1-EC and PxMM/PCMM2G
- Troubleshooting FSoE Safety Parameters

2.4 Fieldbus - EtherNet/IP (ODVA)

These functions are used for explicit messaging.

Name	Description
eipAdapter	Provides information about the current state of the Scanner connection.
eipReadAttr	Sends an explicit message (UCMM) to an EtherNet/IP adapter for reading a single CIP attribute.
eipWriteAttr	Sends an explicit message (UCMM) to an EtherNet/IP adapter for reading a single CIP attribute.

2.4.1 eipAdapter

PLCopen

Function Block - Provides information about the current state of the Scanner connection.

2.4.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
N/A	N/A	N/A	N/A	N/A	This function block has no Inputs. It automatically refers to the adapter in the project.

2.4.1.2 Outputs

Output	Data Type	Range	Unit	Description
Run	BOOL			TRUE, if the EIP stack is running.
IOcnx	BOOL			TRUE, if an I/O connection is established with the Scanner.

2.4.1.3 Remarks

- This function block is used in a program on the Adapter side.
- The servers (adapters) accessed by this block must be configured in the EtherNet/IP Scanner fieldbus configuration.
- Only one explicit message (read or write) can be sent at one time to the same server.
  - If another message is pending you receive the error report 3 (busy) after calling the block to start a new exchange.
- Consider SerializeIn and SerializeOut functions for extracting data from the read buffer.

2.4.1.3.1 Example

```
Inst_eipAdapter();   // read the EtherNet/IP adapter status

EIP_running := Inst_eipAdapter.Run;    // is it running?
EIP_connected := Inst_eipAdapter.IOCnx;   // and connected?
```

See Also

- "eipReadAttr" (→ p. 70)
- "eipWriteAttr" (→ p. 72)

2.4.2 eipReadAttr

PLCopen

Function Block - Sends an explicit message (UCMM) to an EtherNet/IP adapter for reading a single CIP attribute.

2.4.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Attr	UINT				Identifier of the CIP attribute.
Class	UINT				Class identifier of the CIP object.
Data	array of UINT				Buffer where to store the received data. If the actual attribute length is greater than the size of this array, the value is truncated when read.
Inst	UINT				Instance identifier of the CIP object.
SrvIP	STRING				IP address of the server (adapter) (e.g., configured in the EtherNet/IP Scanner configuration).
Snd	BOOL				A rising edge on this input starts the exchange. The DONE output signals the end of exchange.

2.4.2.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			This output is TRUE during one cycle when the exchange is finished, whether the exchange succeeded or failed. **⚠WARNING** Warning: This output can be TRUE just after the call to block when starting a new exchange in case of invalid parameters.
EmErr	UINT			In case of a UCMM error, this is the CIP general status error code.
EmErrExt	UINT			In case of a UCMM error, this is the CIP extended status error code.
Err	UINT			Main error report. Can be one of these values: 0 = No error. 1 = Invalid input arguments. 2 = System is busy (see remarks). 3 = Timeout waiting for the answer (the timeout value is 3 seconds). 4 = UCMM error was returned by the server others = internal errors (reserved for technical support).
RcvSize	UINT			Actual size of the CIP attribute answered by the server. If this size is greater than the size of the DATA input array, it indicates the value was truncated.

2.4.2.3 Remarks

- The CIP instance number can be 16-bits [1...65535].
- The servers (adapters) accessed by this block must be configured in the EtherNet/IP Scanner fieldbus configuration.
- Only one explicit message (read or write) can be sent at one time to the same server.
  - If another message is pending you receive the error report 3 (busy) after calling the block to start a new exchange.
- Consider SerializeIn and SerializeOut functions for extracting data from the read buffer.

2.4.2.4 FBD Language Example

Not available.

2.4.2.5 FFLD Language Example

Not available.

2.4.2.6 IL Language Example

Not available.

2.4.2.7 ST Language Example

```
// used variables
// Inst_eipReadAttr : eipReadAttr ;
// bRead : BOOL ; (* request for READ *)
// DataRead : ARRAY [0 .. 15] OF USINT ; (* read data *)
// Server identification and CIP things
#define SRVIP '192.168.33.21'
#define CLASSID UINT#100
#define INSTID_READ UINT#1
#define ATTRID UINT#3
////////////////////////////////////////////////////////////////////////////
////////////
// requested READ command
if bRead then
    Inst_eipReadAttr (bRead, SRVIP, CLASSID, INSTID_READ, ATTRID, DataRead);
end_if;
// READ answer here ?
if Inst_eipReadAttr.Done then
    // check answer - if OK answered data is in DataRead array
    if Inst_eipReadAttr.Err = 0 then
        printf ('READ ok - size = %lu bytes',
            any_to_dint (Inst_eipReadAttr.RcvSize));
    else
        printf ('READ Error %lu (UCMM Error %lu, %lu)',
            any_to_dint (Inst_eipReadAttr.Err),
            any_to_dint (Inst_eipReadAttr.EmErr),
            any_to_dint (Inst_eipReadAttr.EmErrExt));
    end_if;
    // reset READ command and block input
    Inst_eipReadAttr (false, SRVIP, CLASSID, INSTID_READ, ATTRID, DataRead);
    bRead := false;
end_if;
```

See Also

- "eipAdapter" (→ p. 69)
- "eipWriteAttr" (→ p. 72)

2.4.3 eipWriteAttr

PLCopen

Function Block - Sends an explicit message (UCMM) to an EtherNet/IP adapter for reading a single CIP attribute.

2.4.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Attr	UINT				Identifier of the CIP attribute.
Class	UINT				Class identifier of the CIP object.
Data	array of UINT				Buffer containing the data to write.
Inst	UINT				Instance identifier of the CIP object.
SrvIP	STRING				IP address of the server (adapter) (e.g., configured in the EtherNet/IP Scanner configuration).
Size	UINT				Number of bytes to write. Cannot exceed 450 bytes.
Snd	BOOL				A rising edge on this input starts the exchange. The DONE output signals the end of exchange.

2.4.3.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			This output is TRUE during one cycle when the exchange is finished, whether the exchange succeeded or failed. Warning: This output can be TRUE just after the call to block when starting a new exchange in case of invalid parameters.
EmErr	UINT			In case of a UCMM error, this is the CIP general status error code.
EmErrExt	UINT			In case of a UCMM error, this is the CIP extended status error code.
Err	UINT			Main error report. Can be one of these values: 0 = No error. 1 = Invalid input arguments. 2 = System is busy (see remarks). 3 = Timeout waiting for the answer (the timeout value is 3 seconds). 4 = UCMM error was returned by the server others = internal errors (reserved for technical support).
RcvSize	UINT			Actual size of the CIP attribute answered by the server. If this size is greater than the size of the DATA input array, it indicates the value was truncated.

2.4.3.3 Remarks

- The CIP instance number can be 16-bits [1...65535].
- The servers (adapters) accessed by this block must be configured in the EtherNet/IP Scanner fieldbus configuration.
- Only one explicit message (read or write) can be sent at one time to the same server.
  - If another message is pending you receive the error report 3 (busy) after calling the block to start a new exchange.
- Consider SerializeIn and SerializeOut functions for extracting data from the read buffer.

2.4.3.4 FBD Language Example

Not available.

2.4.3.5 FFLD Language Example

Not available.

2.4.3.6 IL Language Example

Not available.

2.4.3.7 ST Language Example

```
// used variables
// Inst_eipWriteAttr : eipWriteAttr ;
// bWrite : BOOL ; (* request for WRITE *)
// DataWrite : ARRAY [0 .. 15] OF USINT; (* written data *)
// uiSizeWrite : UINT := UINT#16 ; (* number of bytes to read *)
// Server identification and CIP things
#define SRVIP '192.168.33.21'
#define CLASSID UINT#100
#define INSTID_WRITE UINT#2
#define ATTRID UINT#3
////////////////////////////////////////////////////////////////////////////
////////////
// requested WRITE command
if bWrite then
    Inst_eipWriteAttr (bWrite, SRVIP, CLASSID, INSTID_WRITE, ATTRID,
                       uiSizeWrite, DataWrite);
end_if;
// WRITE answer here ?
if Inst_eipWriteAttr.Done then
    // check answer
    if Inst_eipWriteAttr.Err = 0 then
        printf ('WRITE ok');
    else
        printf ('WRITE Error %lu - (UCMM Error %lu, %lu)',
            any_to_dint (Inst_eipWriteAttr.Err),
            any_to_dint (Inst_eipWriteAttr.EmErr),
            any_to_dint (Inst_eipWriteAttr.EmErrExt));
    end_if;
    // reset WRITE command and block input
    Inst_eipWriteAttr (false, SRVIP, CLASSID, INSTID_WRITE, ATTRID,
                       uiSizeWrite, DataWrite);
    bWrite := false;
end_if;
```

See Also

- "eipAdapter" (→ p. 69)
- "eipReadAttr" (→ p. 70)

3 Kollmorgen UDFBs

A Kollmorgen UDFB is a pre-defined function block created by Kollmorgen to simplify certain tasks or demonstrate a particular function.

- A Kollmorgen UDFB must be instantiated before it can be used.
- The code inside a Kollmorgen UDFB can be modified by creating an unlocked copy in the subprogram section in the project tree.

These are the UDFB functions and function blocks:

- "Common" (→ p. 75)
- "Pipe Network" (→ p. 75)
- "PLCopen" (→ p. 76)
- "PLC" (→ p. 76)

3.1 Common

Name	Description
FB_AKDFltRpt	Outputs AKD drive fault information.
FB_AxisPlsPosModulo	Used for any position of a modulo axis in both directions.
FB_AxisPlsPosNoModulo	Used for any position of a non-modulo axis in both directions.
FB_Cylinder	Used to control a cylinder and the Limit Switches.
FB_S700FltRpt	Outputs S700 drive fault Information.

3.2 Pipe Network

Name	Description
MLFB_DriveFault	Returns the fault status, number, and description of the requested axis mapped to a Kollmorgen drive.
MLFB_ECATRestart	Reinitializes the EtherCAT network and the motion engine.
MLFB_HomeFindHomeFastInput	Fast homing to a home switch.
MLFB_HomeFindHomeFastInputModulo	Fast homing to a home switch: Modulo mode.
MLFB_HomeFindHomeInput	Homing to a home switch.
MLFB_HomeFindHomeInputThenZeroAngle	Homing to a home switch plus zero angle.
MLFB_HomeFindLimitFastInput	Fast homing to a limit switch.
MLFB_HomeFindLimitFastInputModulo	Fast homing to a limit switch: Modulo mode.
MLFB_HomeFindLimitInput	Homing to a limit switch.
MLFB_HomeFindLimitInputThenZeroAngle	Homing to a limit switch plus zero angle.
MLFB_HomeFindZeroAngle	Homing to zero angle reference.
MLFB_HomeMoveUntilPosErrExceeded	Homing to the position error is exceeded.
MLFB_ HomeMoveUntilPosErrExceededThenZeroAngle	Homing to the position error is exceeded plus zero angle.

Name	Description
MLFB_HomeUsingCurrentPosition	Homing using current position.
MLFB_Jog	Define to jog an axis in the selected direction at a defined speed.
MLFB_PlsPosFw	Forward position range indicator.
MLFB_PlsPosFwBw	Forward/backward position range indicator.
MLFB_PlsTimeFw	Forward/backward Position/Time range indicator.

3.3 PLCopen

Name	Description
MCFB_AKDFault	Outputs AKD drive fault information.
MCFB_AKDFaultLookup	String message of the corresponding AKD drive fault number.
MCFB_DriveFault	Returns the fault status, number, and description of the requested axis mapped to a Kollmorgen drive.
MCFB_ECATRestart	Reinitializes the EtherCAT network and the motion engine.
MCFB_GearedWebTension	Facilitates dancer and tension control in an electronic geared master/slave machine design.
MCFB_Jog	Define to jog an axis in the selected direction at a defined speed.
MCFB_StepAbsolutes	Homing by setting Actual Position to the position of the feedback.
MCFB_StepAbsSwitch	Homing to a home switch.
MCFB_ StepAbsSwitchFastInput	Fast homing to a home switch.
MCFB_StepBlock	Homing to a physical object, mechanically blocking the movement.
MCFB_StepLimitSwitch	Homing to a limit switch.
MCFB_ StepLimitSwitchFastInput	Fast homing to a limit switch.
MCFB_StepRefPulse	Homing to a zero angle reference.

3.4 PLC

Name	Description
FB_ElapseTime	Keeps track of the time (oTotalOnTime) a Boolean input variable is on.
FB_FirstOrderDigitalFilter	Defined to filter an Analog signal.
FB_PWDutyOutput	Accepts an input value between the minimum and maximum input range and converts this to a duty cycle percentage.
FB_ScaleInput - Scaling Analog IO	Converts un-scaled DINT values from Analog Inputs into user units of type LREAL.
FB_ScaleOutput - Scaling Analog IO	Converts un-scaled LREAL values from a PLC program into units of type DINT.
FB_TemperaturePID	Provides PID temperature control with auto tuning.

Name	Description
PipeNetwork_FFLD - Special Function	Calls the PNCode function block in FFLD POUs.
ProfilesCode_FFLD - Special Function	Calls the Profiles Code Function Block in FFLD POUs.

3.5 Create or Edit UDFBs

3.5.1 Create a UDFB Instance

1. Open the PLC code.
2. Select the UDFB in the Library tree.
3. Drag-and-drop the UDFB in the PLC editor to create the instance of the UDFB.
   An instance of the UDFB has now been created in **Subprograms**.

NOTE

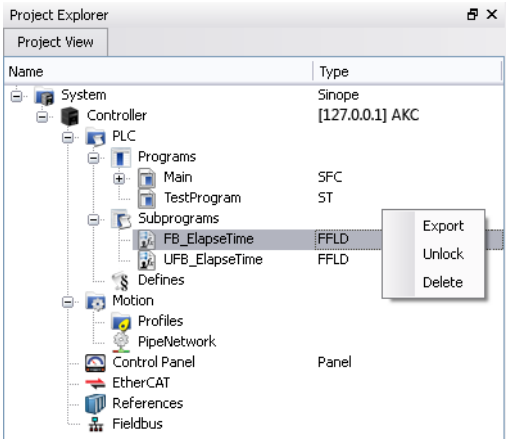
An instance of the UDFB cannot be created directly from the dictionary or from the PLC Editor.

3.5.2 Edit UDFBs

By default, all Kollmorgen UDFBs are protected; they cannot be modified or renamed.
A Kollmorgen UDFB dropped into an instance cannot be edited.

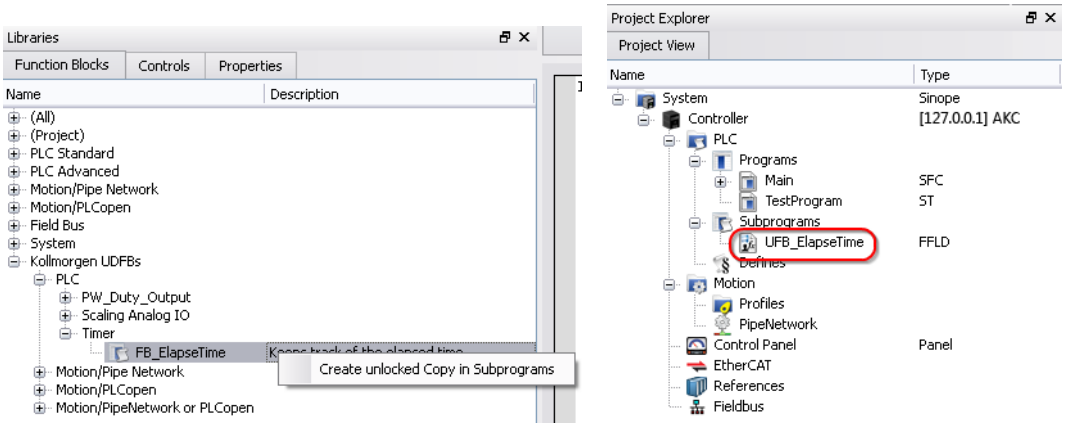
There are two solutions to edit a UDFB:

- Right click a Kollmorgen UDFB that was dropped into an instance (in **Subprograms**) and click **Unlock**.
  This creates an unlocked version of the UDFB with the name **U<sequence number><UDFB name>**.



- Instead of dropping a Kollmorgen UDFB into an instance, right-click the UDFB and click **Create unlocked copy** in **Subprograms**.
  This creates an unlocked instance of the UDFB with the name **U<sequence number><Kollmorgen**

UDFB name>.



After a Kollmorgen UDFB is unlocked, right-click the UDFB to rename or exported it.

- Renamed UDFBs must have unique names.
- Importing a saved UDFB increments the UDFB's name.

TIP

For a UDFB to modify a structure or array based on the output, it must first be defined it as an input. The input is automatically set as an INOUT parameter.
This is because OUTs are strictly simple types.

3.6 UDFB - Motion / Common

These are the Kollmorgen UDFB Motion/Common function and function blocks.

Name	Description
FB_AKDFltRpt	Outputs AKD drive fault information.
FB_AxisPlsPosModulo	Used for any position of a modulo axis in both directions.
FB_AxisPlsPosNoModulo	Used for any position of a non-modulo axis in both directions.
FB_Cylinder	Used to control a cylinder and the Limit Switches.
FB_S700FltRpt	Outputs S700 drive fault Information.

3.6.1 FB_AKDFltRpt

PLCopen ✓ Pipe Network ✓

Function Block - Outputs AKD drive fault information.

3.6.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the Kollmorgen UDFB. Used in the FFLD editor only.
iEtherCat_ID	INT	No range	N/A	No default	EtherCAT address of AKD Drive. Example: 1001 or AKD_1.
iRstFltHist	BOOL	0, 1	N/A	No default	When input is TRUE, clears all Faults saved to drives history.

3.6.1.2 Outputs

Output	Data Type	Range	Unit	Description
oFAULT	BOOL		N/A	TRUE if selected drive currently has a Fault.
oNumberFaults	DINT	0, 10	N/A	Number of faults saved in the drive's history.
oFirstFaultNumber	DINT	100, 999	N/A	Three-digit fault identifier.
oFirstFaultMessage	STRING		N/A	Description of the fault.
oSecondFaultNumber	DINT	100, 999	N/A	Three-digit fault identifier.
oSecondFaultMessage	STRING		N/A	Description of the fault.
oThirdFaultNumber	DINT	100, 999	N/A	Three-digit fault identifier.
oThirdFaultMessage	STRING		N/A	Description of the fault.
oDriveNotUsed	BOOL		N/A	Is this drive: • 0 (zero) = Real. • 1 = Simulated.

3.6.1.3 Remarks

TIP

This function block lists the **earliest occurring** fault first.
This may not be the same fault as is being reported on an AKD's display, which is based on priority.
The "MCFB_AKDFault" (→ p. 135) function block may be preferred as it reports the same error as displayed on the drive.

- The **oFAULT** output turns TRUE when the selected drive goes into a fault state.
- This function block outputs the:
  - Total number of faults in the drive fault history variable. (Pre-Defined Error Field Object 1003h)
  - Fault number and message for the last three drive faults.

- Each fault has two outputs: the fault number and a fault message.
  - The fault number is the same number reported on the display of the drive.
  - The fault message provides a short description of the fault.
  - Example: If the first fault is a feedback error with a F401 is shown on the front of the drive, the output of this FB are:
    - **oFirstFaultNumber** = 401.
    - **oFirstFaultMessage** = Failed To Set Feedback Type.
    - **iResetfaultHistory** resets the faults reported by the FB.
    - **oDriveNotUsed** outputs 1 (True) if the axis is configured to Simulated in the ProjectEtherCAT setup screen.

- This function block can be used with either the PLCopen or Pipe Network Motion engines.

This image shows the function or function block I/O.

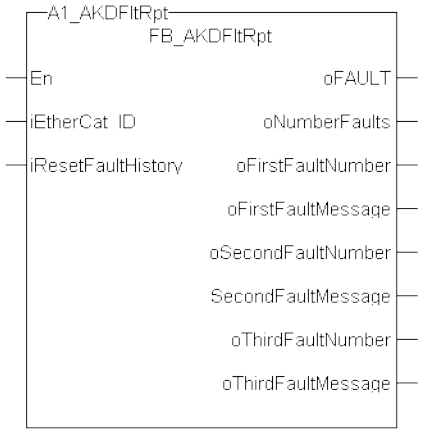
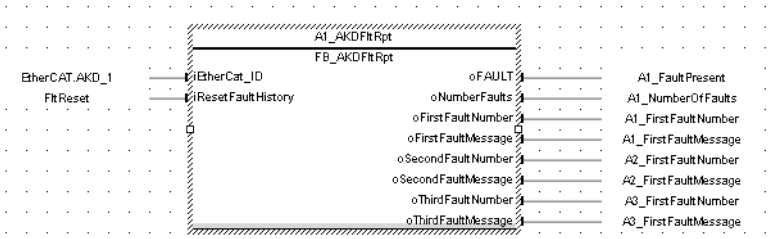


Figure 3-1: AKDFltRpt

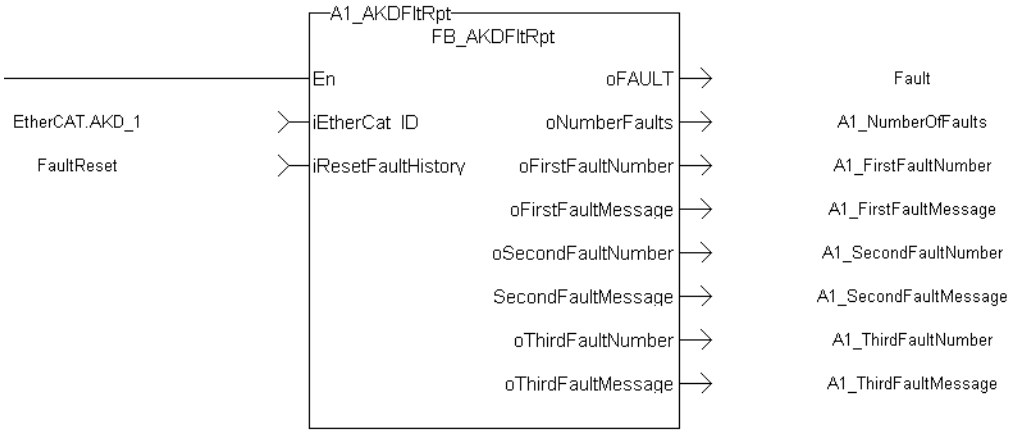
3.6.1.3.1 Usage

- Provide drive fault information the application program uses to determine next steps.
  - Example: Perform a machine-controlled stop or perform an immediate disable of the servo drives.
- In the application program, sends output fault information from this UDFB to the HMI for review by the machine operator.

3.6.1.4 FBD Language Example



3.6.1.5 FFLD Language Example



3.6.1.6 IL Language Example

Not available.

3.6.1.7 ST Language Example

```
//Execute the Function Block
1_AKDFltRpt (1001, resetFaultHistST);

//Read Function Block Outputs
AKD1_Fault:= A1_AKDFltRpt.oFault;
AKD1_NumFault:= A1_AKDFltRpt.oNumberFaults;
AKD1_FirstFaultNumber:= A1_AKDFltRpt.oFirstFaultNumber;
AKD1_FirstFaultMessage:= A1_AKDFltRpt.oFirstFaultMessage;
AKD1_SecondFaultNumber:= A1_AKDFltRpt.oSecondFaultNumber;
AKD1_SecondFaultMessage:= A1_AKDFltRpt.oSecondFaultMessage;
AKD1_ThirdFaultNumber:= A1_AKDFltRpt.oThirdFaultNumber;
AKD1_ThirdFaultMessage:= A1_AKDFltRpt.oThirdFaultMessage;
;
```

NOTE

A1_FaultReporting is an instance of the **FB_S700FltRpt** function block.

See Also

- "MC_ReadStatus" (→ p. 442) (PLCopen Motion Engine)
- "MCFB_AKDFault" (→ p. 135)
- "MLAxisStatus" (→ p. 267) (Pipe Network Motion Engine)

3.6.2 FB_AxisPlsPosModulo

PLCopen ✓ Pipe Network ✓

Function Block - Used for any position of a modulo axis in both directions.

3.6.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL				Enable PLS.
iPosition	LREAL				Any position of a modulo axis.
iModuloPosition	LREAL				Modulo position of the axis.
iStartPos	LREAL				Start position of PLS.
iEndPos	LREAL				End position of PLS.
iDelayTime	TIME				Delay time for compensation.
iHysteresis	LREAL				Hysteresis.
ibForce	BOOL				Force PLS.

3.6.2.2 Outputs

Output	Data Type	Range	Unit	Description
oPLS	BOOL		N/A	Position limit switch.

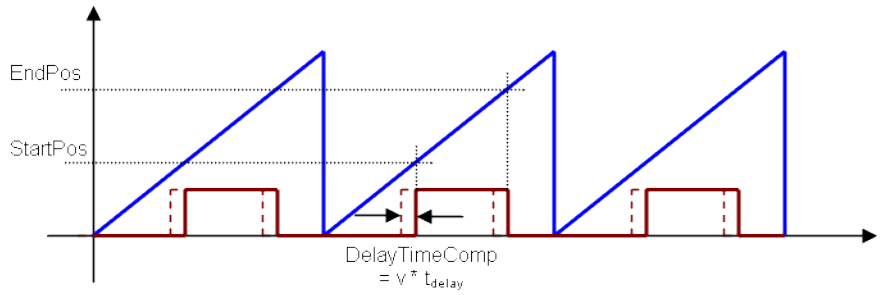
3.6.2.3 Remarks

- The Boolean output **oPLS** is set to:
  - TRUE if the position has crossed the start position.
  - FALSE if the position has crossed the end position.

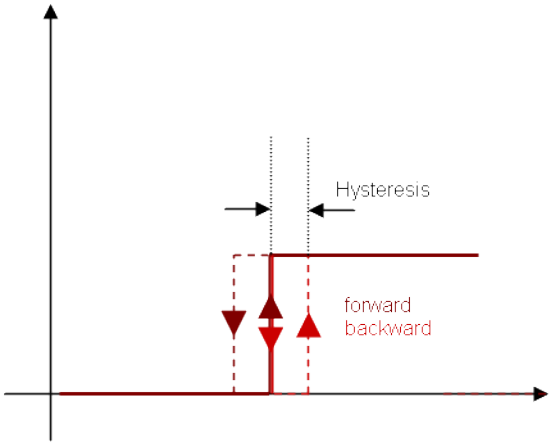
- The function block is executed cyclically.
  - The function block has the possibility to compensate a delay time of the connected device (e.g., glue nozzles).
- It is possible to define a hysteresis for switching on and off of the PLS.

3.6.2.3.1 Example

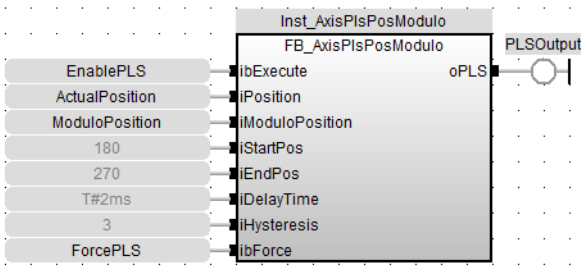
3.6.2.3.1.1 Timing



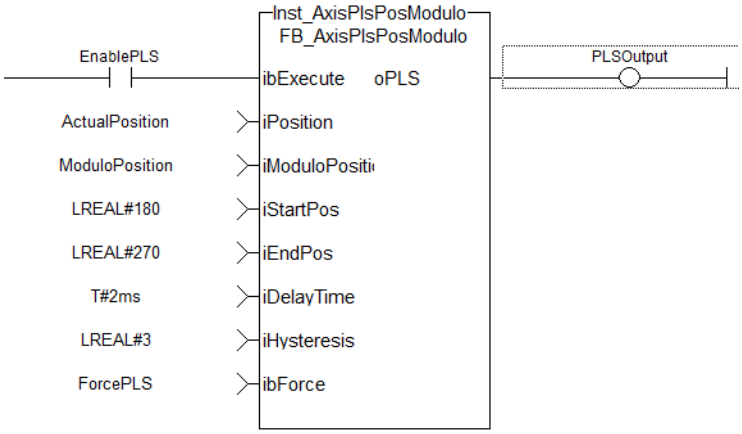
3.6.2.3.1.2 Hysteresis



3.6.2.4 FBD Language Example



3.6.2.5 FFLD Language Example



3.6.2.6 IL Language Example

Not available.

3.6.2.7 ST Language Example

```
//PLSOutput is True when position input is between 180 and 270 with a T#2ms
delay
//Can also force the output to be true with ForceOuput variable
//Hysteresis is on for 3 user units in case direction changes around start
point

Inst_FB_AxisPlsPosModulo( EnablePLS, ActualPosition, ModuloPosition, 180,
270, T#2ms, 3, ForcePLS );

PLSOutput := Inst_FB_AxisPlsPosModulo.oPLS;
```

3.6.3 FB_AxisPlsPosNoModulo

PLCopen ✓ Pipe Network ✓

Function Block - Used for any position of a non-modulo axis in both directions.

3.6.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL				Enable PLS.
iPosition	LREAL				Any position of a non-modulo axis.
iStartPos	LREAL				Start position of PLS.
iEndPos	LREAL				End position of PLS.
iDelayTime	TIME				Delay time for compensation.
iHysteresis	LREAL				Hysteresis.
ibForce	BOOL				Force PLS.

3.6.3.2 Outputs

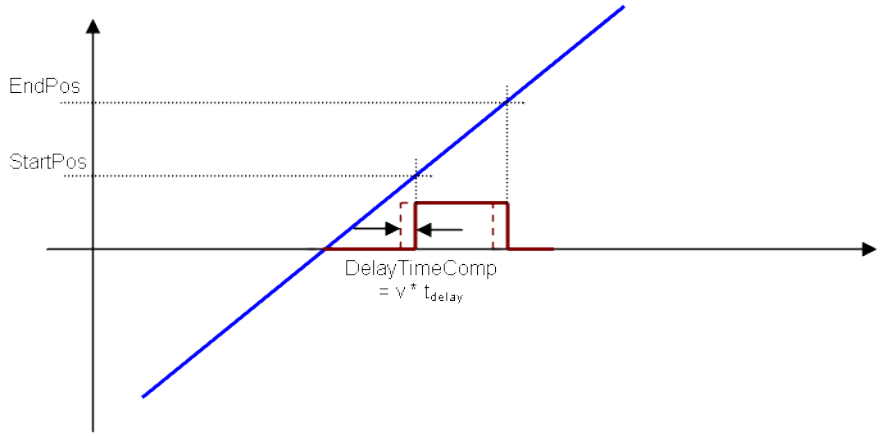
Output	Data Type	Range	Unit	Description
oPLS	BOOL		N/A	Position limit switch.

3.6.3.3 Remarks

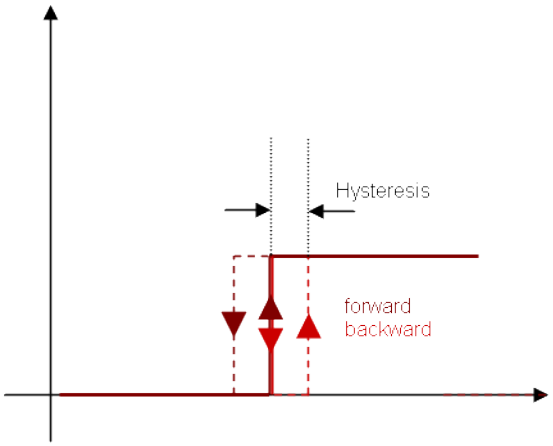
- The Boolean output **oPLS** is set to:
  - TRUE if the position has crossed the start position.
  - FALSE if the position has crossed the end position.
- The function block has the possibility to compensate a delay time of the connected device (e.g., glue nozzles).
- It is possible to define a hysteresis for switching on and off of the PLS.

3.6.3.3.1 Example

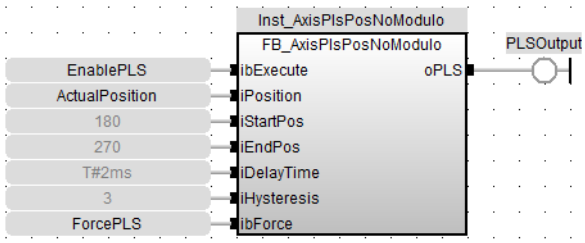
3.6.3.3.1.1 Timing



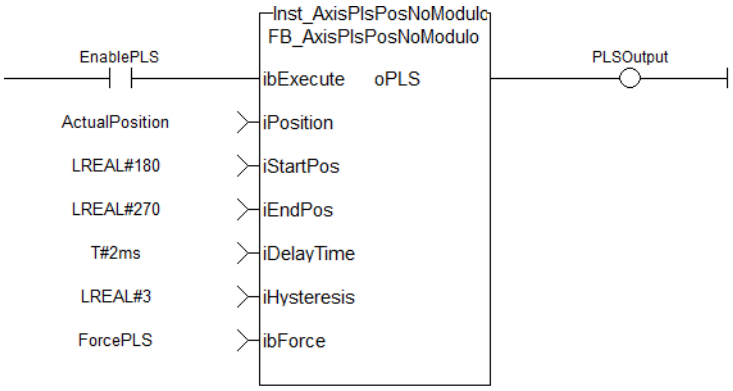
3.6.3.3.1.2 Hysteresis



3.6.3.4 FBD Language Example



3.6.3.5 FFLD Language Example



3.6.3.6 IL Language Example

Not available.

3.6.3.7 ST Language Example

```
//PLSOutput is True when position input is between 180 and 270 with a T#2ms
delay
//Can also force the output to be true with ForceOuput variable
//Hysteresis is on for 3 user units in case direction changes around start
point

Inst_FB_AxisPlsPosNoModulo( EnablePLS, ActualPosition, 180, 270, T#2ms, 3,
ForcePLS );

PLSOutput := Inst_FB_AxisPlsPosNoModulo.oPLS;
```

3.6.4 FB_Cylinder

PLCopen ✓ Pipe Network ✓

Function Block - Used to control a cylinder and the Limit Switches.

3.6.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
iInA	BOOL				Set direction A.
iInB	BOOL				Set direction B.
iLsA	BOOL				Limit Switch at end of direction A.
iLsB	BOOL				Limit Switch at end of direction B.
iCtrlTime	TIME				Max time until LimitSwitch has to be reached.
iResetFault	BOOL				Reset Fault (is set to FALSE by UDFB!).

3.6.4.2 Outputs

Output	Data Type	Range	Unit	Description
oDirA	BOOL			Direction A.
oDirB	BOOL			Direction B.
oFaultLsA	BOOL			Fault of LimitSwitch at the end of direction A.
oFaultLsB	BOOL			Fault of LimitSwitch at the end of direction B.

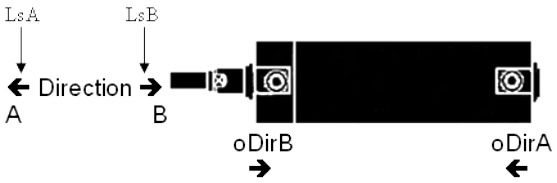
3.6.4.3 Remarks

- There are two inputs, **iLnA** and **iInB**, to set the direction of the movement and the belonging LimSwitches iLsA and iLsB.
- If iInA is set to TRUE, the output oDirA is set to TRUE.
  - After a time value defined by CtrlTime, the iLsA has to become TRUE otherwise a fault FaultLsA appears.
    - Just as in direction B.

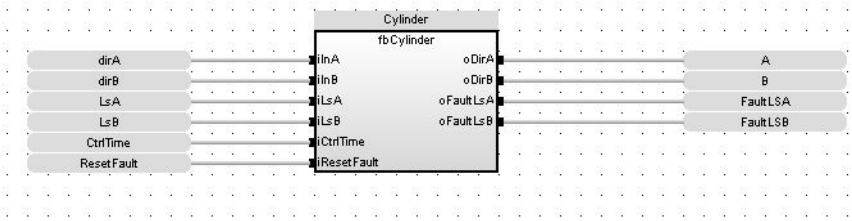
- If both iLsA and iLsB are TRUE, then a Fault depending of the output is set.
- If both iLnA and iLnB are given (e.g., to stop the cylinder movement), no limit switch is controlled.
- All faults can be reset by input iResetFault.

3.6.4.3.1 Usage

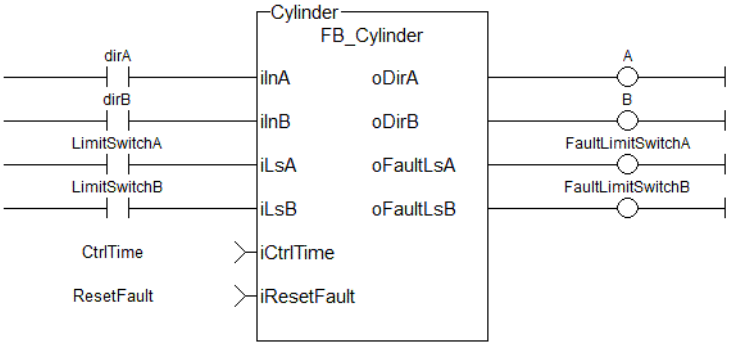
- The signal flow is valid for both directions (A and B)
- If oDirA AND oDirB are active there is no Fault Control.
- The Fault can be reset by iRestFault = True.



3.6.4.4 FBD Language Example



3.6.4.5 FFLD Language Example



3.6.4.6 IL Language Example

Not available.

3.6.4.7 ST Language Example

```
//Electric Cylinder with limit switch controls
Inst_FB_Cylinder( dirA, dirB, LimitSwitchA, LimitSwitchB, CtrlTime,
ResetFault );
A := Inst_FB_Cylinder.oDirA;
B := Inst_FB_Cylinder.oDirB;
FaultLimitSwitchA := Inst_FB_Cylinder.oFaultLsA;
FaultLimitSwitchB := Inst_FB_Cylinder.oFaultLsB;
```

3.6.5 FB_S700FltRpt

PLCopen ✓ Pipe Network ✓

Function Block - Outputs S700 drive fault Information.

3.6.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the Kollmorgen UDFB. Used in the FFLD editor only.
iEtherCat_ID	INT	No range	N/A	No default	EtherCAT address of AKD Drive. Example: 1001 or AKD_1.
iRstFltHist	BOOL	0, 1	N/A	No default	When input is TRUE, clears all Faults saved to drives history.

3.6.5.2 Outputs

Output	Data Type	Range	Unit	Description
oFAULT	BOOL		N/A	TRUE if selected drive currently has a Fault.
oNumberFaults	DINT	0, 10	N/A	Number of faults saved in the drive's history.
oFirstFaultNumber	DINT	100, 999	N/A	Three-digit fault identifier.
oFirstFaultMessage	STRING		N/A	Description of the fault.
oSecondFaultNumber	DINT	100, 999	N/A	Three-digit fault identifier.

Output	Data Type	Range	Unit	Description
oSecondFaultMessage	STRING		N/A	Description of the fault.
oThirdFaultNumber	DINT	100, 999	N/A	Three-digit fault identifier.
oThirdFaultMessage	STRING		N/A	Description of the fault.
oDriveNotUsed	BOOL		N/A	Is this drive: • 0 (zero) = Real. • 1 = Simulated.

3.6.5.3 Remarks

- The **oFAULT** output turns TRUE when the selected drive goes into a fault state.
- This function block outputs the:
  - Total number of faults in the drive fault history variable. (FLTHIST)
  - Fault number and message for the last three drive faults.
- Each fault has two outputs: the fault number and a fault message.
  - The fault number is the same number reported on the display of the drive.
  - The fault message provides a short description of the fault.
  - Example: If the first fault is a feedback error with a F04 is displayed on the front of the drive, the output of this FB are:
    - **oFirstFaultNumber** = 04
    - **oFirstFaultMessage** = Feedback Error
    - **iResetfaultHistory** resets the faults reported by the FB.
    - **oDriveNotUsed** outputs 1 (True) if the axis is configured to Simulated in the ProjectEtherCAT setup screen.
- This function block can be used with either the PLCopen or Pipe Network Motion engines.

This image shows the function or function block I/O.

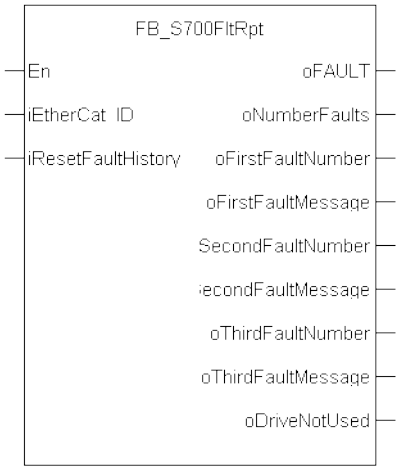
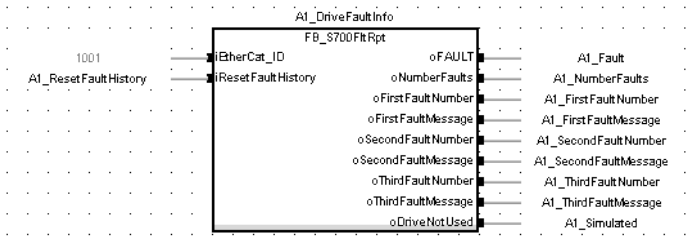


Figure 3-2: S700FltRpt

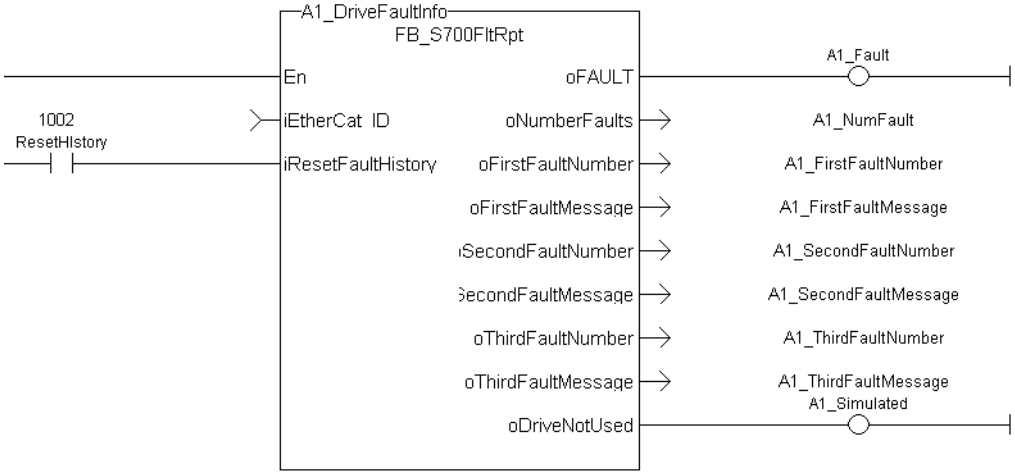
3.6.5.3.1 Usage

- Provide drive fault information the application program uses to determine next steps.
  - Example: Perform a machine-controlled stop or perform an immediate disable of the servo drives.
- In the application program, sends output fault information from this UDFB to the HMI for review by the machine operator.

3.6.5.4 FBD Language Example



3.6.5.5 FFLD Language Example



3.6.5.6 IL Language Example

Not available.

3.6.5.7 ST Language Example

```
//Execute the Function Block
A1_FaultReporting (1001, 0);

//Read Function Block Outputs
A1_Fault:= A1_FaultReporting.oFault;
A1_NumFault:= A1_FaultReporting.oNumberFaults;
A1_FirstFaultNumber:= A1_FaultReporting.oFirstFaultNumber;
A1_FirstFaultMessage:= A1_FaultReporting.oFirstFaultMessage;
A1_SecondFaultNumber:= A1_FaultReporting.oSecondFaultNumber;
A1_SecondFaultMessage:= A1_FaultReporting.oSecondFaultMessage;
A1_ThirdFaultNumber:= A1_FaultReporting.oThirdFaultNumber;
A1_ThirdFaultMessage:= A1_FaultReporting.oThirdFaultMessage;
A1_Simulated:= A1_FaultReporting.oDriveNotUsed;
```

NOTE

A1_FaultReporting is an instance of the **FB_S700FltRpt** function block.

See Also

- "MC_ReadStatus" (→ p. 442) (PLCopen Motion Engine)
- "MLAxisStatus" (→ p. 267) (Pipe Network Motion Engine)

3.7 UDFB - Motion / Pipe Network

These are the Kollmorgen UDFB Motion/Pipe Network function and function blocks.

Name	Description
MLFB_DriveFault	Returns the fault status, number, and description of the requested axis mapped to a Kollmorgen drive.
MLFB_ECATRestart	Reinitializes the EtherCAT network and the motion engine.
MLFB_HomeFindHomeFastInput	Fast homing to a home switch.
MLFB_HomeFindHomeFastInputModulo	Fast homing to a home switch: Modulo mode.
MLFB_HomeFindHomeInput	Homing to a home switch.
MLFB_HomeFindHomeInputThenZeroAngle	Homing to a home switch plus zero angle.
MLFB_HomeFindLimitFastInput	Fast homing to a limit switch.
MLFB_HomeFindLimitFastInputModulo	Fast homing to a limit switch: Modulo mode.
MLFB_HomeFindLimitInput	Homing to a limit switch.
MLFB_HomeFindLimitInputThenZeroAngle	Homing to a limit switch plus zero angle.
MLFB_HomeFindZeroAngle	Homing to zero angle reference.
MLFB_HomeMoveUntilPosErrExceeded	Homing to the position error is exceeded.
MLFB_ HomeMoveUntilPosErrExceededThenZeroAngle	Homing to the position error is exceeded plus zero angle.
MLFB_HomeUsingCurrentPosition	Homing using current position.
MLFB_Jog	Define to jog an axis in the selected direction at a defined speed.
MLFB_PlsPosFw	Forward position range indicator.
MLFB_PlsPosFwBw	Forward/backward position range indicator.
MLFB_PlsTimeFw	Forward/backward Position/Time range indicator.

3.7.1 MLFB_DriveFault

Function Block - Returns the fault status, number, and description of the requested axis mapped to a Kollmorgen drive.

3.7.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the Kollmorgen UDFB. Used in the FFLD editor only.
AxisID	DINT	-2147483648 to 2147483647	N/A	No default	ID of the axis block of the Pipe Network.

3.7.1.2 Outputs

Output	Data Type	Range	Unit	Description
FAULT	BOOL	0, 1	N/A	TRUE if the selected drive currently has a Fault.

FaultNumber	DINT	No range	N/A	If the axis is:
				• S300/S700
				  • Three-digit fault identifier.
				  • See S300 & S700 Errors and Warnings on KDN for a list of fault codes.
				• AKD
				  • Three-digit fault identifier.
				  • See Fault and Warning Messages for a list of fault codes.
				• AKD2G
				  • Four-digit fault identifier.
				  • See Faults and Warnings View for a list of fault codes.
				• AKT2G Stepper
				  • Drive Status word (bitmask).
				  • See "Bitmask Causes" (→ p. 91) for more information.

FaultDescription	STRING	N/A	N/A	Description of the fault.

3.7.1.2.1 Bitmask Causes

Bit	Description	Cause
0	Saturated	Drive stage operates with maximum duty cycle.
1	Over temperature.	Internal temperature is higher than 80 °C.
2	Torque overload.	Motor current is higher than the rated current.
3	Under voltage.	Motor supply voltage is either: • 20% lower than the configured nominal voltage (warning). • Less than 8V.
4	Over voltage.	Motor supply voltage is 10% higher than the configured nominal voltage.
5	Short circuit A.	Short circuit in motor coil A.
6	Short circuit B.	Short circuit in motor coil B.
7	No control power.	Control voltage at the power contacts is less than 12V.
8	Misc. error.	Either: • The terminal initialization failed. • Supply voltage is less than 8V. • Internal terminal temperature is higher than 100 °C.
9	Configuration error.	CoE change has not yet been adopted into the current configuration.

3.7.1.3 Remarks

IMPORTANT

This function block requires "FB_S700FltRpt" (→ p. 87), "MCFB_AKDFault" (→ p. 135), and "MCFB_AKDFaultLookup" (→ p. 136) subprograms imported to project to compile and function.

- The FAULT output returns TRUE when the selected drive goes into a fault state.
- The fault number and description depend on the drive type mapped to the axis.
  - If the drive is an:
    - AKD or AKD2G, the fault number is the same number as reported on the display of the drive.
    - AKT2G Stepper, the fault number represents the drive status word.
      - This word is a bitmask that represents the various error conditions.

This image shows the function or function block I/O.

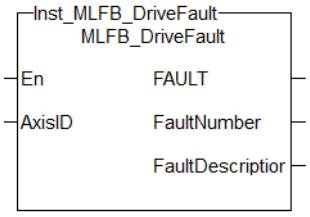
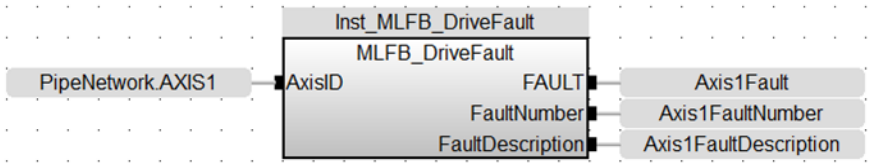


Figure 3-3: MCFB_DriveFault

3.7.1.3.1 Usage

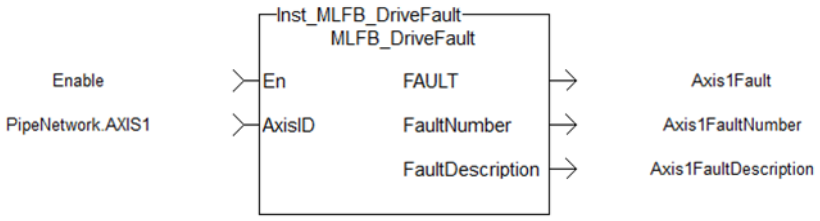
- Provide drive fault information the application program uses to determine next steps.
  - Example: Perform a machine-controlled stop or perform an immediate disable of the servo drives.
- In the application program, sends output fault information from this UDFB to the HMI for review by the machine operator.

3.7.1.4 FBD Language Example



3.7.1.5 FFLD Language Example

Not available.



3.7.1.6 IL Language Example

Not available.

3.7.1.7 ST Language Example

```
//Execute and Read the Function Block
Inst_MLFB_DriveFault(PipeNetwork.AXIS1);
Axis1Fault := Inst_MLFB_DriveFault.FAULT;
Axis1FaultNumber := Inst_MLFB_DriveFault.FaultNumber;
Axis1FaultDescription := Inst_MLFB_DriveFault.FaultDescription;
```

See Also

- "FB_S700FltRpt" (→ p. 87)
- "MCFB_AKDFaultLookup" (→ p. 136)

3.7.2 MLFB_ECATRestart

Pipe Network

Function Block - Reinitializes the EtherCAT network and the motion engine.

3.7.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
iEN	BOOL	0, 1	N/A	No default	Enables the Kollmorgen UDFB. Used in the FFLD editor only.
iRSTERR	BOOL	1, 256	0, 1	No default	Clears the motion engine and EtherCAT network errors in case of any faults.

3.7.2.2 Outputs

Output	Data Type	Range	Unit	Description
oOK	BOOL	0, 1	N/A	Function block activated status.
oDONE	BOOL	0, 1	N/A	Execution complete.
oERR	BOOL	0, 1	N/A	TRUE if the system initialization fails.

3.7.2.3 Remarks

- This function block clears motion engine errors, motion bus driver errors and EtherCAT network errors before reinitializing the motion engine, if requested to do so.
- This UDFB allows the EtherCAT and motion engines to restart without having to restart the entire project.

Examples:

- EtherCAT network wire is replaced or accidentally disconnected.
- Axis setup parameters defined by CreateAxis and/or InitAxis function need to be changed while the application is running.

This image shows the function or function block I/O.

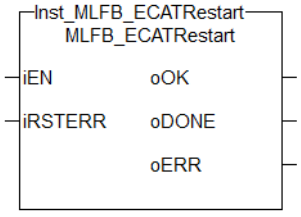
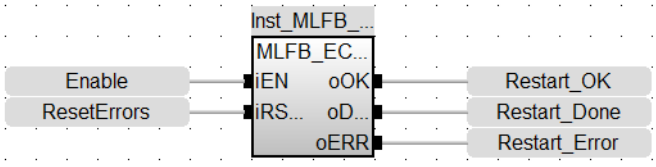
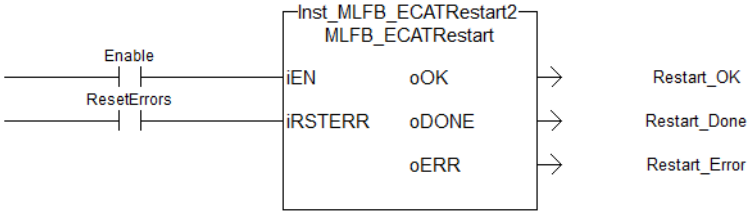


Figure 3-4: MLFB_ECATRestart

3.7.2.4 FBD Language Example



3.7.2.5 FFLD Language Example



3.7.2.6 IL Language Example

Not available.

3.7.2.7 ST Language Example

```
Inst_MLFB_ECATRestart( Restart, ResetErrors );
IF Inst_MLFB_ECATRestart.oDONE THEN
   RestartComplte:=1;
End_IF;
```

See Also

- "ClearCtrlErrors" (→ p. 652)
- "MLMotionInit" (→ p. 549)
- "MLMotionRstErr" (→ p. 551)
- "MLMotionStart" (→ p. 552)

3.7.3 Fast Homing - UDFB - Motion/Pipe Network

These are the UDFB Motion/Pipe Network Fast Homing function blocks.

Name	Description
"MLFB_HomeFindHomeFastInput" (→ p. 94)	Performs a single-axis home to a limit switch connected to a High Speed Input.
"MLFB_HomeFindHomeFastInputModulo" (→ p. 98)	Performs a single-axis home to a limit switch connected to a High Speed Input.
"MLFB_HomeFindLimitFastInput" (→ p. 103)	Performs a single-axis home to a limit switch connected to a High Speed Input.
"MLFB_HomeFindLimitFastInputModulo" (→ p. 106)	Performs a single-axis home to a limit switch connected to a High Speed Input.

3.7.3.1 MLFB_HomeFindHomeFastInput

Pipe Network

Function Block - Fast homing to a home switch.

3.7.3.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL	0, 1	N/A	No default	Request the homing step procedure at the rising edge.
iAxisID	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
iPosition	LREAL	No range	User units	No default	Offset position applied after home switch is found.
ibDirection	BOOL	0, 1	N/A	No default	Defines the axis homing direction. • 0 (zero) = Clockwise rotation. • 1 = Counterclockwise rotation.
iVelocity	LREAL	No range	User unit/sec	No default	Commanded velocity for the homing move.
iAcceleration	LREAL	No range	User unit/sec^2	No default	Commanded acceleration for the homing move.
iDeceleration	LREAL	No range	User unit/sec^2	No default	Commanded deceleration for the homing move.
ibHomeSwitchMode	BOOL	0, 1	N/A	No default	Limit switch state to complete homing. • 0 (zero) = Rising edge of switch. • 1 = Falling edge of switch.
ibPosLimitSwitch	BOOL	0, 1	N/A	No default	The positive direction limit switch input I/O point.
ibNegLimitSwitch	BOOL	0, 1	N/A	No default	The negative direction limit switch input I/O point.
iTimeout	TIME	No range	Sec	No default	Maximum time for homing move to complete. • If exceeded, the homing procedure errors out. • 0 (zero) = no time limit.
ibFastInputNumber	BOOL	0, 1	N/A	No default	Limit switch state to complete homing. • 0 (zero) = Fast Input Number 1. • 1 = Fast Input Number 2.
iCycleTime	LREAL	No range	Microseconds	No default	EtherCAT cycle time. Either 250, 500, or 1000.

3.7.3.1.2 Outputs

Output	Data Type	Range	Unit	Description
obDone	BOOL		N/A	Indicates the move completed successfully. The Command Position has reached the endpoint.
obActive	BOOL		N/A	Indicates this move is the Active move.
obError	BOOL		N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
oErrorID	DINT		N/A	Indicates the error if the **Error** output is set to TRUE. Error identifier:

ErrorID	Description
1	Axis in error.
2	Axis is not enabled.
3	Timeout exceeded or expired.
4	SDO read/write error.
5	Input parameter out of range.

3.7.3.1.3 Remarks

- Performs a single-axis home to a limit switch connected to a High Speed Input.
- The motor starts to move according to the direction setting.
  - The home position has been found as soon as the fast input selected is triggered on the edge selected.
- An absolute move is made to the triggered position and the position value is set.
  - The hardware limit switches are monitored during the homing procedure.
- The drive behaves this way in case a hardware limit switch is active before the home-switch has been activated:
  The motor changes the direction until the home switch is crossed.

This image shows the function or function block I/O.

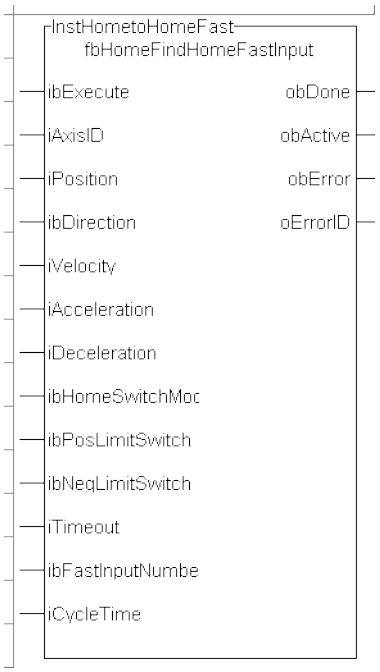
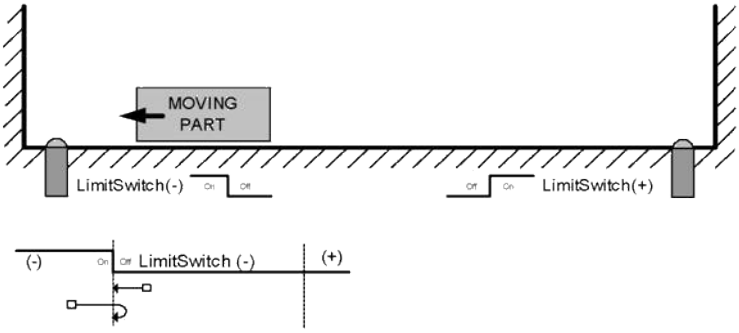


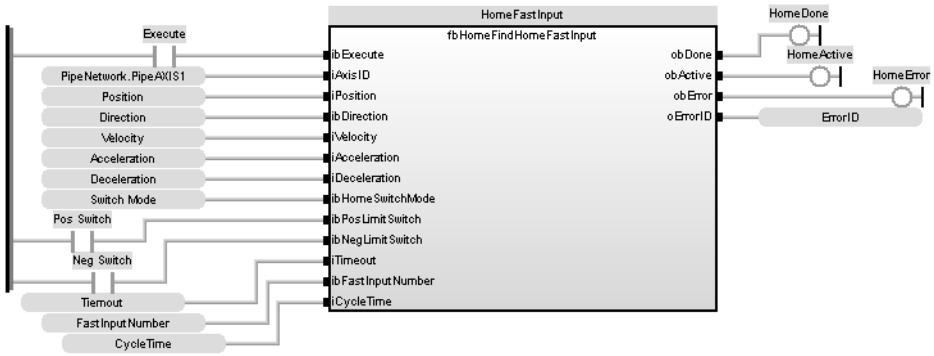
Figure 3-5: MLFB HomeFindHomeFastInput

3.7.3.1.3.1 Usage

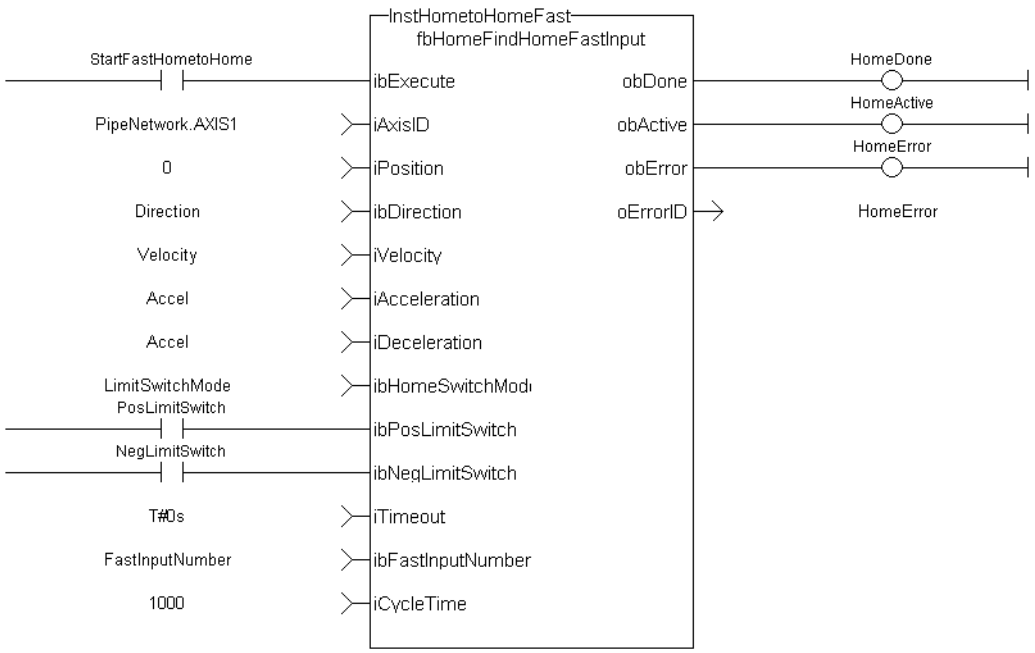
- This procedure performs a homing function searching for a sensor using only High Speed Input Switches.
  - A High Speed Limit Switch has 1 Off (or On) area.
- Home is commanded by the user in the designated homing direction at the selected or programmed Velocity.
- If **LimitSwitch** is found On with a rising Execute, the process is started in the opposite direction as specified.
- If **LimitSwitch** is found Off Edge (released), the process is restarted again in original direction. This can be On, depending on **LimitSwitchMode** setting.
  - This ensures the end conditions are always the same.
- The Timeout can cause an error if exceeded.



3.7.3.1.4 FBD Language Example



3.7.3.1.5 FFLD Language Example



3.7.3.1.6 IL Language Example

Not available.

3.7.3.1.7 ST Language Example

```
Direction:= 0;
Position:=1000;
Velocity:=1000;
Acceleration:=10000;
Deceleration:=10000;
SwitchMode:=0;
Timeout:=T#100;
FastInputNumber:=0;
CycleTime:=1000;

inst_fbHomeFindHomeFastInput(True, Axis1, Position, Direction, Velocity,
Acceleration, Deceleration, HomeSwitchMode, PosLimitSwitch, NegLimitSwitch,
Timeout, FastInputNumber, CycleTime);

HomeComplete :=inst_fbHomeFindHomeFastInput.Done;
HomeActive :=inst_fbHomeFindHomeFastInput.Active;
HomeError :=inst_fbHomeFindHomeFastInput.Error;
HomeErrorID :=inst_fbHomeFindHomeFastInput.ErrorID;

(* PosLimitSwitch and NegLimitSwtch are declared I/O points *)
```

See Also

- "MLFB_HomeFindHomeFastInputModulo" (→ p. 98)
- "MLFB_HomeFindLimitFastInput" (→ p. 103)
- "MLFB_HomeFindLimitFastInputModulo" (→ p. 106)

3.7.3.2 MLFB_HomeFindHomeFastInputModulo

Function Block - Fast homing to a home switch: Modulo mode.

3.7.3.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL	0, 1	N/A	No default	Request the homing step procedure at the rising edge.
iAxisID	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
iPosition	LREAL	No range	User units	No default	Offset position applied after home switch is found.
ibDirection	BOOL	0, 1	N/A	No default	Defines the axis homing direction. • 0 (zero) = Clockwise rotation. • 1 = Counterclockwise rotation.
iVelocity	LREAL	No range	User unit/sec	No default	Commanded velocity for the homing move.
iAcceleration	LREAL	No range	User unit/sec^2	No default	Commanded acceleration for the homing move.
iDeceleration	LREAL	No range	User unit/sec^2	No default	Commanded deceleration for the homing move.
ibLimitSwitchMode	BOOL	0, 1	N/A	No default	Limit switch state to complete homing. • 0 (zero) = Rising edge of switch. • 1 = Falling edge of switch.
ibPosLimitSwitch	BOOL	0, 1	N/A	No default	The positive direction limit switch input I/O point.
ibNegLimitSwitch	BOOL	0, 1	N/A	No default	The negative direction limit switch input I/O point.
iTimeout	TIME	No range	Sec	No default	Maximum time for homing move to complete. • If exceeded, the homing procedure errors out. • 0 (zero) = no time limit.
ibFastInputNumber	BOOL	0, 1	N/A	No default	Limit switch state to complete homing. • 0 (zero) = Fast Input Number 1. • 1 = Fast Input Number 2.
iCycleTime	LREAL	No range	Microseconds	No default	EtherCAT cycle time. Either 250, 500, or 1000.

3.7.3.2.2 Outputs

Output	Data Type	Range	Unit	Description
obDone	BOOL		N/A	Indicates the move completed successfully. The Command Position has reached the endpoint.
obActive	BOOL		N/A	Indicates this move is the Active move.
obError	BOOL		N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
oErrorID	DINT		N/A	Indicates the error if the **Error** output is set to TRUE. Error identifier:

ErrorID	Description
1	Axis in error.
2	Axis is not enabled.
3	Timeout exceeded or expired.
4	SDO read/write error.
5	Input parameter out of range.

3.7.3.2.3 Remarks

- Performs a single-axis home to a limit switch connected to a High Speed Input.
- This function is used when the axis is set-up in Modulo mode.
- The motor starts to move according to the direction setting.
  - The home position has been found as soon as the fast input selected is triggered on the edge selected.
- An absolute move is made to the triggered position and the position value is set.
  - The hardware limit switches are monitored during the homing procedure.
- The drive behaves this way in case a hardware limit switch is active before the home-switch has been activated:
  The motor changes the direction until the home switch is crossed.

This image shows the function or function block I/O.

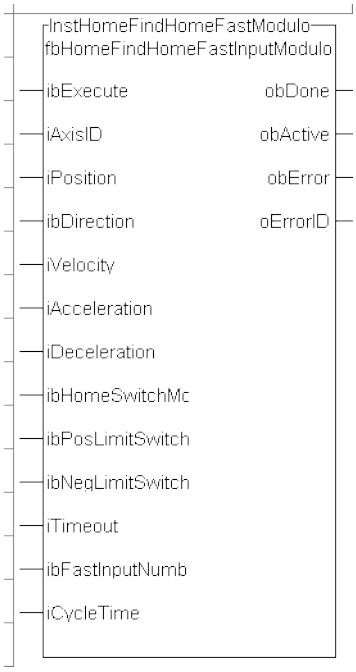
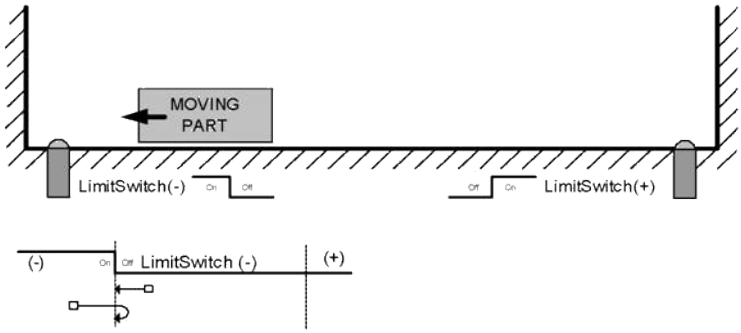


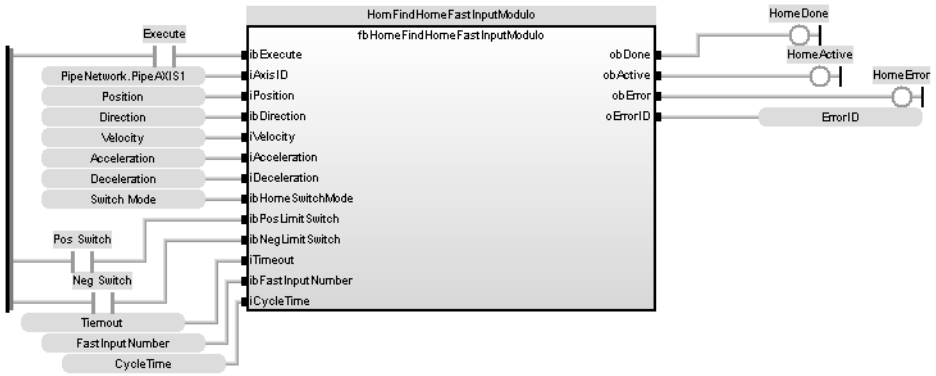
Figure 3-6: MLFB HomeFindHomeFastInputModulo

3.7.3.2.3.1 Usage

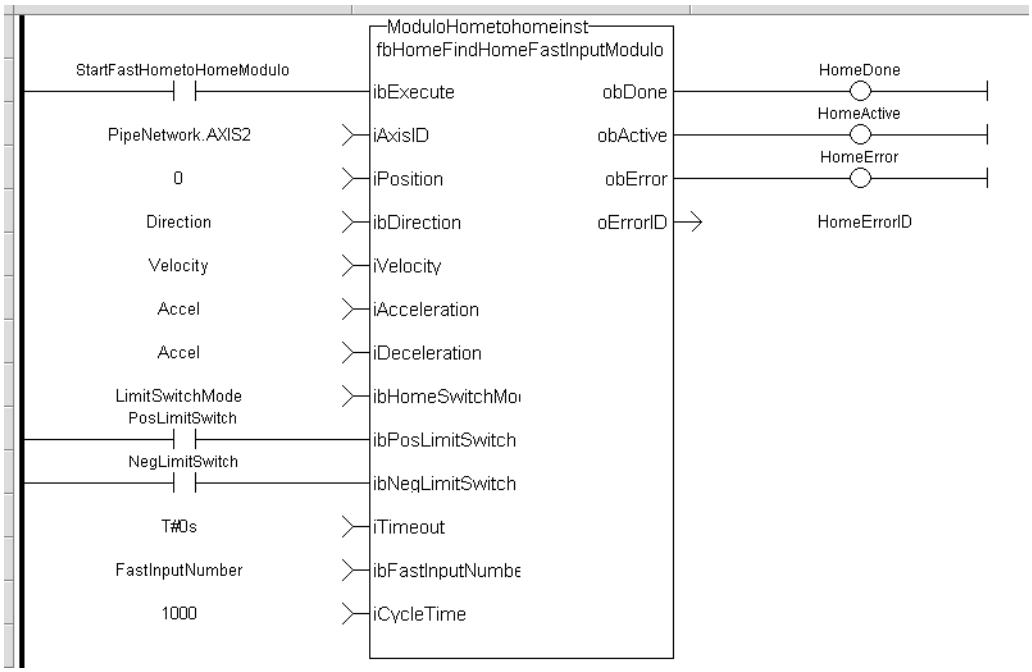
- This procedure performs a homing function searching for a sensor using only High Speed Input Switches.
  - A High Speed Limit Switch has 1 Off (or On) area.
- Home is commanded by the user in the designated homing direction at the selected or programmed Velocity.
- If **LimitSwitch** is found On with a rising Execute, the process is started in the opposite direction as specified.
- If **LimitSwitch** is found Off Edge (released), the process is restarted again in original direction. This can be On, depending on **LimitSwitchMode** setting.
  - This ensures the end conditions are always the same.
- The Timeout can cause an error if exceeded.



3.7.3.2.4 FBD Language Example



3.7.3.2.5 FFLD Language Example



3.7.3.2.6 IL Language Example

Not available.

3.7.3.2.7 ST Language Example

```
Direction:= 0;
Position:=1000;
Velocity:=1000;
Acceleration:=10000;
Deceleration:=10000;
SwitchMode:=0;
Timeout:=T#100;
FastInputNumber:=0;
CycleTime:=1000;

inst_fbHomeFindHomeFastInputModulo(True, Axis1, Position, Direction,
Velocity, Acceleration, Deceleration, PosLimitSwitch, NegLimitSwitch,
Timeout, FastInputNumber, CycleTime);

HomeComplete :=inst_fbHomeFindHomeFastInputModulo.Done;
HomeActive :=inst_fbHomeFindHomeFastInputModulo.Active;
```

```
HomeError :=inst_fbHomeFindHomeFastInputModulo.Error;
HomeErrorID :=inst_fbHomeFindHomeFastInputModulo.ErrorID;

(* PosLimitSwitch and NegLimitSwtch are declared I/O points *)
```

See Also

- "MLFB_HomeFindHomeFastInput" (→ p. 94)
- "MLFB_HomeFindLimitFastInput" (→ p. 103)
- "MLFB_HomeFindLimitFastInputModulo" (→ p. 106)

3.7.3.3 MLFB_HomeFindLimitFastInput

Function Block - Fast homing to a limit switch.

3.7.3.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL	0, 1	N/A	No default	Request the homing step procedure at the rising edge.
iAxisID	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
iPosition	LREAL	No range	User units	No default	Offset position applied after home switch is found.
ibDirection	BOOL	0, 1	N/A	No default	Defines the axis homing direction. • 0 (zero) = Clockwise rotation. • 1 = Counterclockwise rotation.
iVelocity	LREAL	No range	User unit/sec	No default	Commanded velocity for the homing move.
iAcceleration	LREAL	No range	User unit/sec^2	No default	Commanded acceleration for the homing move.
iDeceleration	LREAL	No range	User unit/sec^2	No default	Commanded deceleration for the homing move.
ibLimitSwitchMode	BOOL	0, 1	N/A	No default	Limit switch state to complete homing. • 0 (zero) = Rising edge of switch. • 1 = Falling edge of switch.

Input	Data Type	Range	Unit	Default	Description
ibFastInputNumber	BOOL	0, 1	N/A	No default	Limit switch state to complete homing. • 0 (zero) = Fast Input Number 1. • 1 = Fast Input Number 2.
iCycleTime	LREAL	No range	Microseconds	No default	EtherCAT cycle time. Either 250, 500, or 1000.

3.7.3.3.2 Outputs

Output	Data Type	Range	Unit	Description
obDone	BOOL		N/A	Indicates the move completed successfully. The Command Position has reached the endpoint.
obActive	BOOL		N/A	Indicates this move is the Active move.
obError	BOOL		N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
oErrorID	DINT		N/A	Indicates the error if the **Error** output is set to TRUE. Error identifier:

ErrorID	Description
1	Axis in error.
2	Axis is not enabled.
3	Timeout exceeded or expired.
4	SDO read/write error.
5	Input parameter out of range.

3.7.3.3.3 Remarks

- Performs a single-axis home to a limit switch connected to a High Speed Input.
- The motor starts to move according to the direction setting.
  - The home position has been found as soon as the fast input selected is triggered on the edge selected.
- An absolute move is made to the triggered position and the position value is set.

This image shows the function or function block I/O.

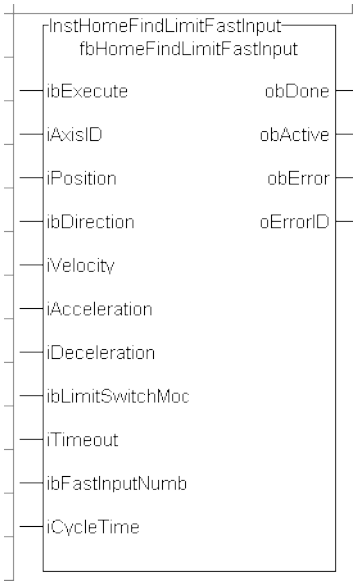
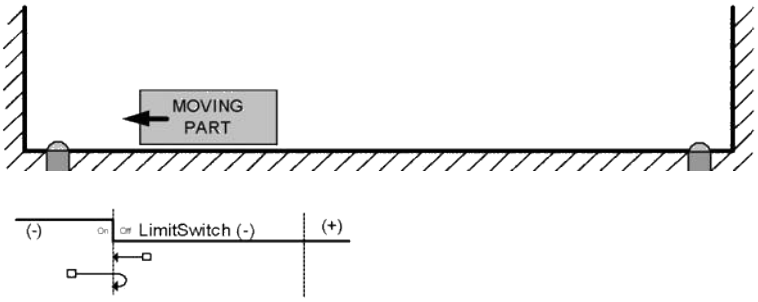


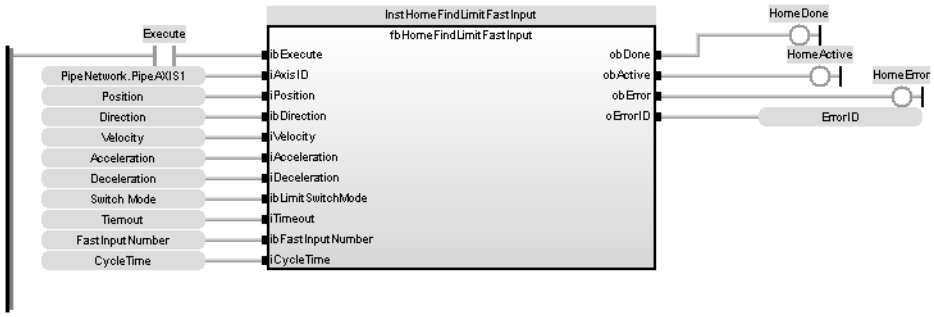
Figure 3-7: MLFB HomeFindLimitFastInput

3.7.3.3.3.1 Usage

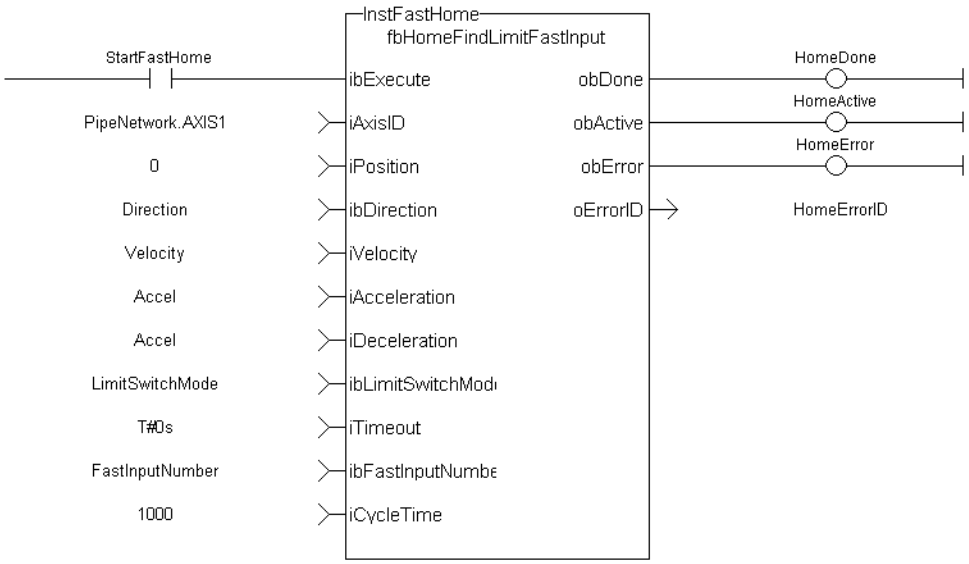
- This procedure performs a homing function searching for a sensor using only High Speed Input Switches.
  - A High Speed Limit Switch has 1 Off (or On) area.
- Home is commanded by the user in the designated homing direction at the selected or programmed Velocity.
- The Timeout can cause an error if exceeded.



3.7.3.3.4 FBD Language Example



3.7.3.3.5 FFLD Language Example



3.7.3.3.6 IL Language Example

Not available.

3.7.3.3.7 ST Language Example

```
Direction:= 0;
Position:=1000;
Velocity:=1000;
Acceleration:=10000;
Deceleration:=10000;
SwitchMode:=0;
Timeout:=T#100;
FastInputNumber:=0;
CycleTime:=1000;

inst_fbHomeFindLimitFastInput(True, Axis1, Position, Direction, Velocity,
Acceleration, Deceleration, LimitSwitchMode, Timeout, FastInputNumber,
CycleTime);

HomeComplete :=inst_fbHomeFindLimitFastInput.Done;
HomeActive :=inst_fbHomeFindLimitFastInput.Active;
HomeError :=inst_fbHomeFindLimitFastInput.Error;
HomeErrorID :=inst_fbHomeFindLimitFastInput.ErrorID;
```

See Also

- "MLFB_HomeFindHomeFastInput" (→ p. 94)
- "MLFB_HomeFindHomeFastInputModulo" (→ p. 98)
- "MLFB_HomeFindLimitFastInputModulo" (→ p. 106)

3.7.3.4 MLFB_HomeFindLimitFastInputModulo

Pipe Network

Function Block - Fast homing to a limit switch: Modulo mode.

3.7.3.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL	0, 1	N/A	No default	Request the homing step procedure at the rising edge.
iAxisID	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
iPosition	LREAL	No range	User units	No default	Offset position applied after home switch is found.
ibDirection	BOOL	0, 1	N/A	No default	Defines the axis homing direction. • 0 (zero) = Clockwise rotation. • 1 = Counterclockwise rotation.
iVelocity	LREAL	No range	User unit/sec	No default	Commanded velocity for the homing move.
iAcceleration	LREAL	No range	User unit/sec^2	No default	Commanded acceleration for the homing move.
iDeceleration	LREAL	No range	User unit/sec^2	No default	Commanded deceleration for the homing move.
ibLimitSwitchMode	BOOL	0, 1	N/A	No default	Limit switch state to complete homing. • 0 (zero) = Rising edge of switch. • 1 = Falling edge of switch.
iTimeout	TIME	No range	Sec	No default	Maximum time for homing move to complete. • If exceeded, the homing procedure errors out. • 0 (zero) = no time limit.
ibFastInputNumber	BOOL	0, 1	N/A	No default	Limit switch state to complete homing. • 0 (zero) = Fast Input Number 1. • 1 = Fast Input Number 2.
iCycleTime	LREAL	No range	Microseconds	No default	EtherCAT cycle time. Either 250, 500, or 1000.

3.7.3.4.2 Outputs

Output	Data Type	Range	Unit	Description
obDone	BOOL		N/A	Indicates the move completed successfully. The Command Position has reached the endpoint.
obActive	BOOL		N/A	Indicates this move is the Active move.

Output	Data Type	Range	Unit	Description
obError	BOOL		N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
oErrorID	DINT		N/A	Indicates the error if the **Error** output is set to TRUE. Error identifier:

ErrorID	Description
1	Axis in error.
2	Axis is not enabled.
3	Timeout exceeded or expired.
4	SDO read/write error.
5	Input parameter out of range.

3.7.3.4.3 Remarks

- Performs a single-axis home to a limit switch connected to a High Speed Input.
- This function is used when the axis is set-up in Modulo mode.
- The motor starts to move according to the direction setting.
  - The home position has been found as soon as the fast input selected is triggered on the edge selected.
- An absolute move is made to the triggered position and the position value is set.

This image shows the function or function block I/O.

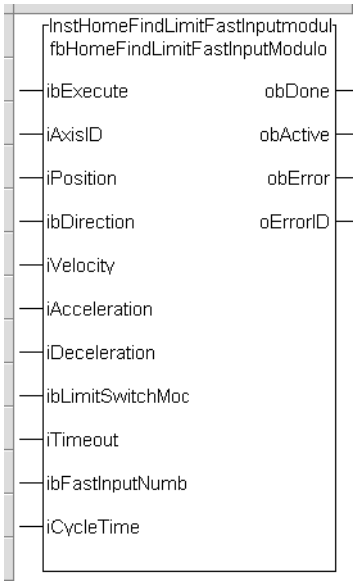
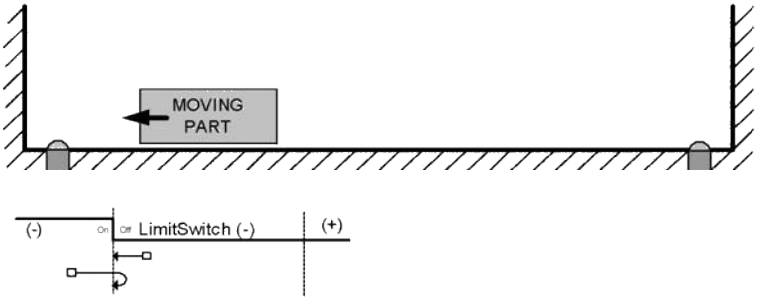


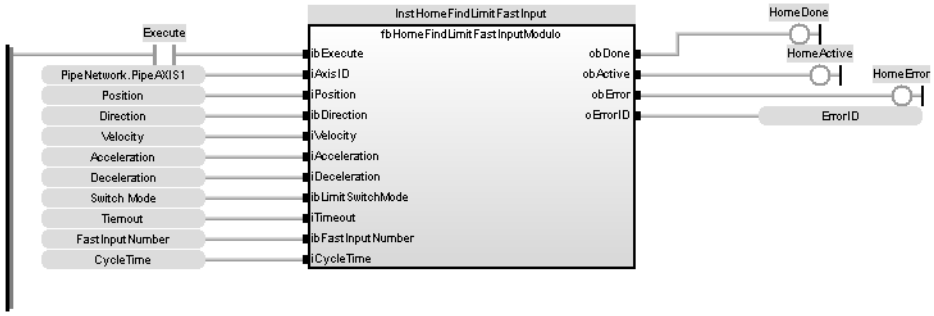
Figure 3-8: MLFB HomeFindLimitFastInputModulo

3.7.3.4.3.1 Usage

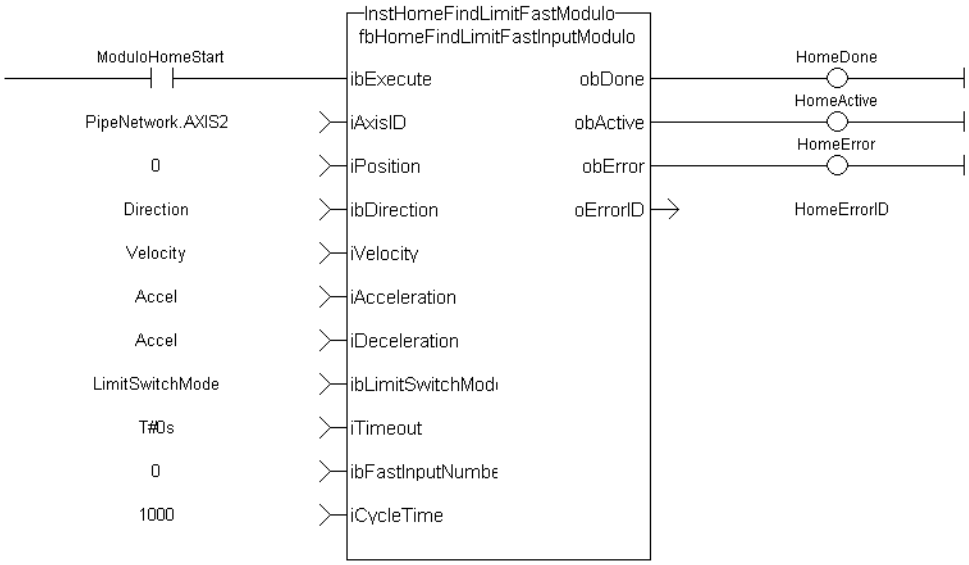
- This procedure performs a homing function searching for a sensor using only High Speed Input Switches.
  - A High Speed Limit Switch has 1 Off (or On) area.
- Home is commanded by the user in the designated homing direction at the selected or programmed Velocity.
- The Timeout can cause an error if exceeded.



3.7.3.4.4 FBD Language Example



3.7.3.4.5 FFLD Language Example



3.7.3.4.6 IL Language Example

Not available.

3.7.3.4.7 ST Language Example

```
Direction:= 0;
Position:=1000;
Velocity:=1000;
Acceleration:=10000;
Deceleration:=10000;
SwitchMode:=0;
Timeout:=T#100;
FastInputNumber:=0;
```

```
CycleTime:=1000;

inst_fbHomeFindLimitFastInputModulo(True, Axis1, Position, Direction,
Velocity, Acceleration, Deceleration, LimitSwitchMode, Timeout,
FastInputNumber, CycleTime);

HomeComplete :=inst_fbHomeFindLimitFastInputModulo.Done;
HomeActive :=inst_fbHomeFindLimitFastInputModulo.Active;
HomeError :=inst_fbHomeFindLimitFastInputModulo.Error;
HomeErrorID :=inst_fbHomeFindLimitFastInputModulo.ErrorID;
```

See Also

- "MLFB_HomeFindHomeFastInput" (→ p. 94)
- "MLFB_HomeFindHomeFastInputModulo" (→ p. 98)
- "MLFB_HomeFindLimitFastInput" (→ p. 103)

3.7.4 Homing - UDFB - Motion/Pipe Network

These are the UDFB Motion/Pipe Network Homing function blocks.

Name	Description
"MLFB_HomeFindHomeInput" (→ p. 110)	Homing to a home switch.
"MLFB_HomeFindHomeInputThenZeroAngle" (→ p. 113)	Homing to a home switch plus zero angle.
"MLFB_HomeFindLimitInput" (→ p. 115)	Homing to a limit switch.
"MLFB_HomeFindLimitInputThenZeroAngle" (→ p. 117)	Homing to a limit switch plus zero angle.
"MLFB_HomeFindZeroAngle" (→ p. 119)	Homing to zero angle reference.
"MLFB_HomeMoveUntilPosErrExceeded" (→ p. 121)	Homing to the position error is exceeded.
"MLFB_HomeMoveUntilPosErrExceededThenZeroAngle" (→ p. 123)	Homing to the position error is exceeded plus zero angle.
"MLFB_HomeUsingCurrentPosition" (→ p. 126)	Homing using current position.

3.7.4.1 MLFB_HomeFindHomeInput

Pipe Network ✓

Function Block - Homing to a home switch.

3.7.4.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL				Start homing, edge-triggered.
iAxisID	DINT				ID of the axis block of the Pipe Network.
iPosition	LREAL				Reference position.
ibDirection	BOOL				• 0 (zero) = Positive. • 1 = Negative.

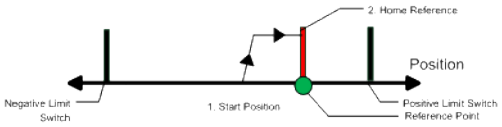
Input	Data Type	Range	Unit	Default	Description
iVelocity	LREAL				Reference speed.
iAcceleration	LREAL				Reference acceleration.
iDeceleration	LREAL				Reference deceleration.
ibHomeInput	BOOL				Home input, high-active.
ibPosLimitSwitch	BOOL				Positive limit switch, high-active.
ibNegLimitSwitch	BOOL				Negative limit switch, high-active.
iTimeout	TIME				Time monitoring (T#0ms: off).

3.7.4.1.2 Outputs

Output	Data Type	Range	Unit	Description
obDone	BOOL			Done bit.
obActive	BOOL			Active bit.
obError	BOOL			Error bit.
oErrorID	DINT			Error identifier:

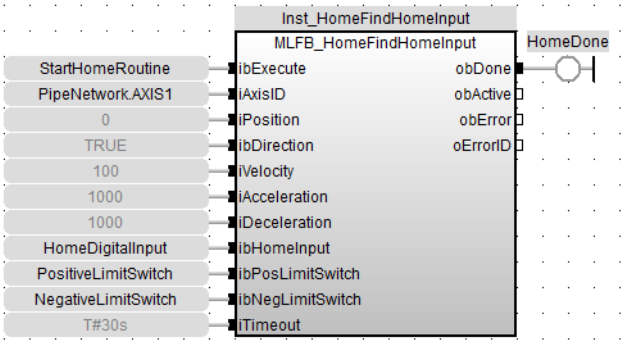
ErrorID	Description
1	Axis in error.
2	Axis is not enabled.
3	Timeout exceeded or expired.
4	SDO read/write error.
5	Input parameter out of range.

3.7.4.1.3 Remarks

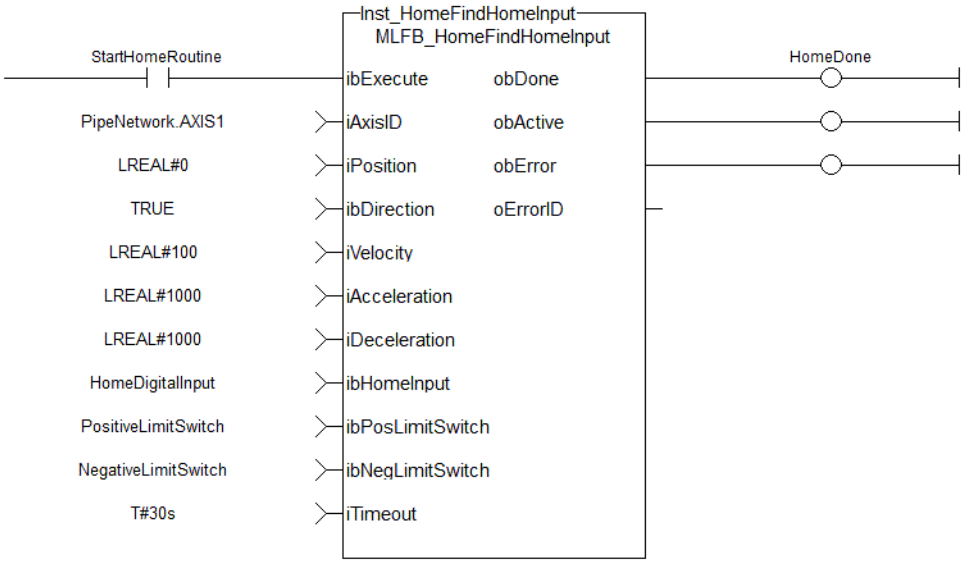


- The motor starts to move according to the direction setting.
  - The home position has been found as soon as the home-switch becomes active during a motion in direction of the direction setting.
- The hardware limit switches are monitored during the homing procedure.
- The drive behaves this way in case a hardware limit switch is active before the home-switch has been activated:
  The motor changes the direction until the home switch is crossed.
- The motor ramps down to 0 (zero) velocity and reverses direction again after crossing the home-switch.
  - The home-switch is now activated according to the direction setting and the home-position has been found.
- The command position of the drive is immediately set to the position value and the motor ramps down to 0 (zero) velocity.

3.7.4.1.4 FBD Language Example



3.7.4.1.5 FFLD Language Example



3.7.4.1.6 IL Language Example

Not available.

3.7.4.1.7 ST Language Example

```
//Call homing function on Axis1 with preset velocity, accel, and decel values
//Start in negative direction, change if limit switch seen before home switch
//after seeing home switch, set axis position to zero
Inst_MLFB_HomeFindHomeInput( StartHomeRoutine,
   PipeNetwork.AXIS1,
   0,
   TRUE,
   100,
   1000,
   1000,
   HomeDigitalInput,
   PositiveLimitSwitch,
   NegativeLimitSwitch,
   T#30s );

HomeDone := Inst_MLFB_HomeFindHomeInput.obDone;
```

See Also

- "MLFB_HomeFindHomeInputThenZeroAngle" (→ p. 113)
- "MLFB_HomeFindLimitInput" (→ p. 115)
- "MLFB_HomeFindLimitInputThenZeroAngle" (→ p. 117)
- "MLFB_HomeFindZeroAngle" (→ p. 119)
- "MLFB_HomeMoveUntilPosErrExceeded" (→ p. 121)
- "MLFB_HomeMoveUntilPosErrExceededThenZeroAngle" (→ p. 123)
- "MLFB_HomeUsingCurrentPosition" (→ p. 126)

3.7.4.2 MLFB_HomeFindHomeInputThenZeroAngle

Function Block - Homing to a home switch plus zero angle.

3.7.4.2.1 Inputs

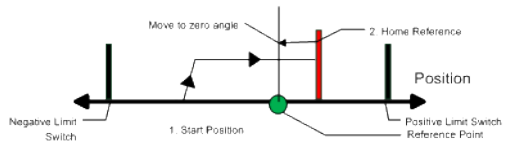
Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL				Start homing, edge-triggered.
iAxisID	DINT				ID of the axis block of the Pipe Network.
iPosition	LREAL				Reference position.
ibDirection	BOOL				• 0 (zero) = Positive. • 1 = Negative.
iVelocity	LREAL				Reference speed.
iAcceleration	LREAL				Reference acceleration.
iDeceleration	LREAL				Reference deceleration.
ibHomeInput	BOOL				Home input, high-active.
ibPosLimitSwitch	BOOL				Positive limit switch, high-active.
ibNegLimitSwitch	BOOL				Negative limit switch, high-active.
iTimeout	TIME				Time monitoring (T#0ms: off).

3.7.4.2.2 Outputs

Output	Data Type	Range	Unit	Description
obDone	BOOL			Done bit.
obActive	BOOL			Active bit.
obError	BOOL			Error bit.
oErrorID	DINT			Error identifier:

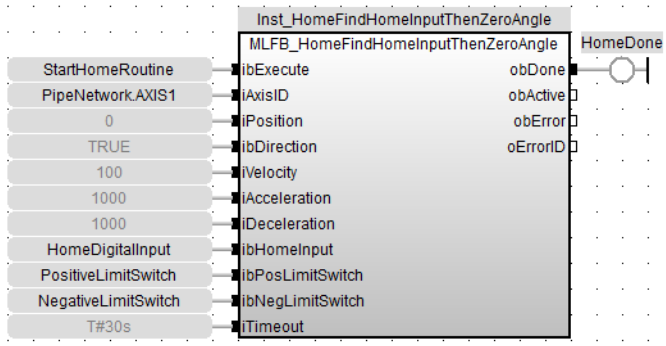
ErrorID	Description
1	Axis in error.
2	Axis is not enabled.
3	Timeout exceeded or expired.
4	SDO read/write error.
5	Input parameter out of range.

3.7.4.2.3 Remarks

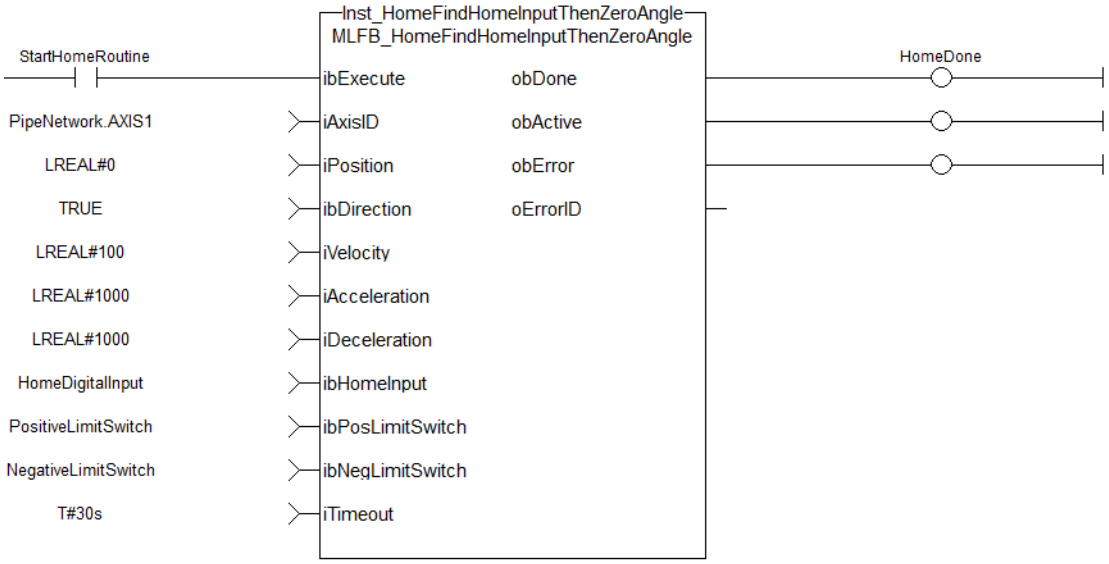


- Similar to the Find Home Limit method.
  - The find input home then find zero angle.
  - Mode follows the same steps, but upon completion of the move, it continues to move to find the zero angle reference of the motor.

3.7.4.2.4 FBD Language Example



3.7.4.2.5 FFLD Language Example



3.7.4.2.6 IL Language Example

Not available.

3.7.4.2.7 ST Language Example

```
//Call homing function on Axis1 with preset velocity, accel, and decel values
//Start in negative direction, change if limit switch seen before home switch
//after seeing home switch and moving to zero angle, set axis position to
zero
Inst_MLFB_HomeFindHomeInputThenZeroAngle( StartHomeRoutine,
   PipeNetwork.AXIS1,
   0,
   TRUE,
   100,
```

```
    1000,
    1000,
    HomeDigitalInput,
    PositiveLimitSwitch,
    NegativeLimitSwitch,
    T#30s );

HomeDone := Inst_MLFB_HomeFindHomeInputThenZeroAngle.obDone;
```

See Also

- "MLFB_HomeFindHomeInput" (→ p. 110)
- "MLFB_HomeFindLimitInput" (→ p. 115)
- "MLFB_HomeFindLimitInputThenZeroAngle" (→ p. 117)
- "MLFB_HomeFindZeroAngle" (→ p. 119)
- "MLFB_HomeMoveUntilPosErrExceeded" (→ p. 121)
- "MLFB_HomeMoveUntilPosErrExceededThenZeroAngle" (→ p. 123)
- "MLFB_HomeUsingCurrentPosition" (→ p. 126)

3.7.4.3 MLFB_HomeFindLimitInput

Function Block - Homing to a limit switch.

3.7.4.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL				Start homing, edge-triggered.
iAxisID	DINT				ID of the axis block of the Pipe Network.
iPosition	LREAL				Reference position.
ibDirection	BOOL				• 0 (zero) = Positive. • 1 = Negative.
iVelocity	LREAL				Reference speed.
iAcceleration	LREAL				Reference acceleration.
iDeceleration	LREAL				Reference deceleration.
ibLimitSwitch	BOOL				Positive or negative limit switch, high-active. Depends on the ibDirection.
iTimeout	TIME				Time monitoring (T#0ms: off).

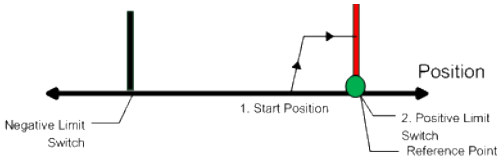
3.7.4.3.2 Outputs

Output	Data Type	Range	Unit	Description
obDone	BOOL			Done bit.
obActive	BOOL			Active bit.
obError	BOOL			Error bit.

Output	Data Type	Range	Unit	Description
oErrorID	DINT			Error identifier:

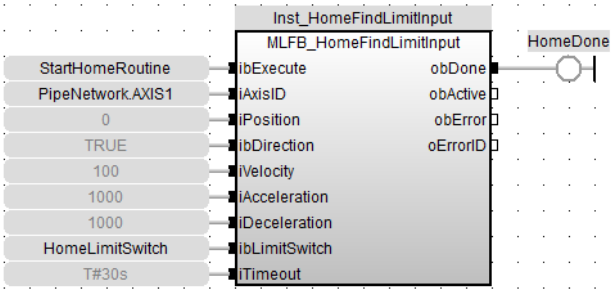
ErrorID	Description
1	Axis in error.
2	Axis is not enabled.
3	Timeout exceeded or expired.
4	SDO read/write error.
5	Input parameter out of range.

3.7.4.3.3 Remarks

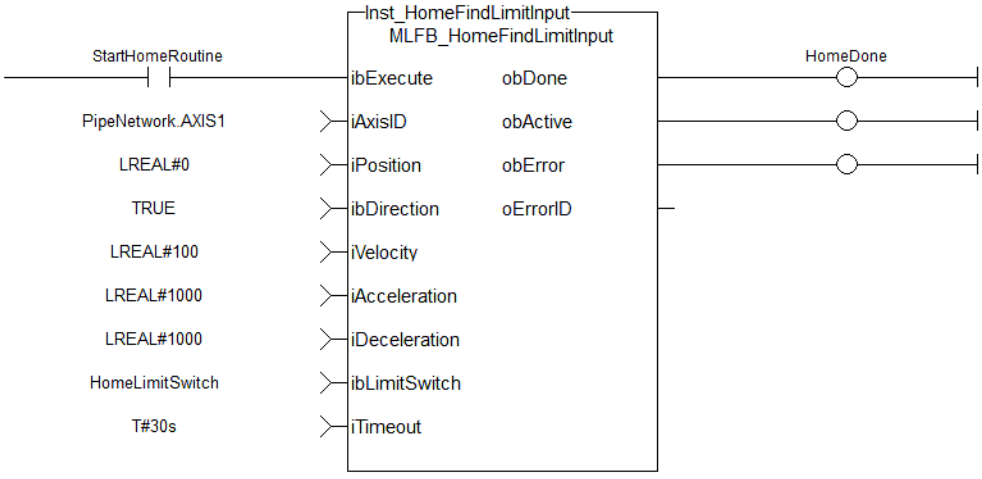


- The find limit input mode moves to a limit input.
- This method is used if you have a positive or negative limit switch available that you want to establish as a home reference point.

3.7.4.3.4 FBD Language Example



3.7.4.3.5 FFLD Language Example



3.7.4.3.6 IL Language Example

Not available.

3.7.4.3.7 ST Language Example

```
//Call homing function on Axis1 with preset velocity, accel, and decel values
//Start in negative direction and stop when axis hits limit switch or times
out
//after seeing limit switch, set axis position to zero
Inst_MLFB_HomeFindLimitInput( StartHomeRoutine,
   PipeNetwork.AXIS1,
   0,
   TRUE,
   100,
   1000,
   1000,
   HomeDigitalInput,
   T#30s );

HomeDone := Inst_MLFB_HomeFindLimitInput.obDone;
```

See Also

- "MLFB_HomeFindHomeInput" (→ p. 110)
- "MLFB_HomeFindHomeInputThenZeroAngle" (→ p. 113)
- "MLFB_HomeFindLimitInputThenZeroAngle" (→ p. 117)
- "MLFB_HomeFindZeroAngle" (→ p. 119)
- "MLFB_HomeMoveUntilPosErrExceeded" (→ p. 121)
- "MLFB_HomeMoveUntilPosErrExceededThenZeroAngle" (→ p. 123)
- "MLFB_HomeUsingCurrentPosition" (→ p. 126)

3.7.4.4 MLFB_HomeFindLimitInputThenZeroAngle

Pipe Network ✓

Function Block - Homing to a limit switch plus zero angle.

3.7.4.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL				Start homing, edge-triggered.
iAxisID	BOOL				ID of the axis block of the Pipe Network.
iPosition	BOOL				Reference position.
ibDirection	BOOL				• 0 (zero) = Positive. • 1 = Negative.
iVelocity	BOOL				Reference speed.
iAcceleration	BOOL				Reference acceleration.
iDeceleration	BOOL				Reference deceleration.
ibLimitSwitch	BOOL				Positive or negative limit switch, high-active. Depends on the ibDirection.
iTimeout	BOOL				Time monitoring (T#0ms: off).

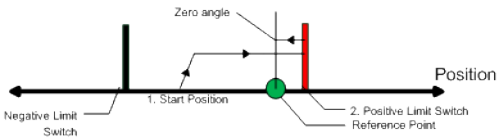
3.7.4.4.2 Outputs

Output	Data Type	Range	Unit	Description
obDone	BOOL			Done bit.
obActive	BOOL			Active bit.

Output	Data Type	Range	Unit	Description
obError	BOOL			Error bit.
oErrorID	DINT			Error identifier:

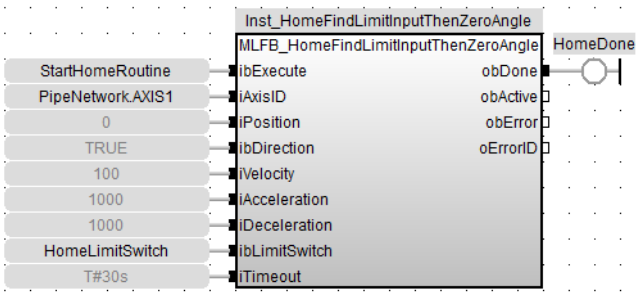
ErrorID	Description
1	Axis in error.
2	Axis is not enabled.
3	Timeout exceeded or expired.
4	SDO read/write error.
5	Input parameter out of range.

3.7.4.4.3 Remarks

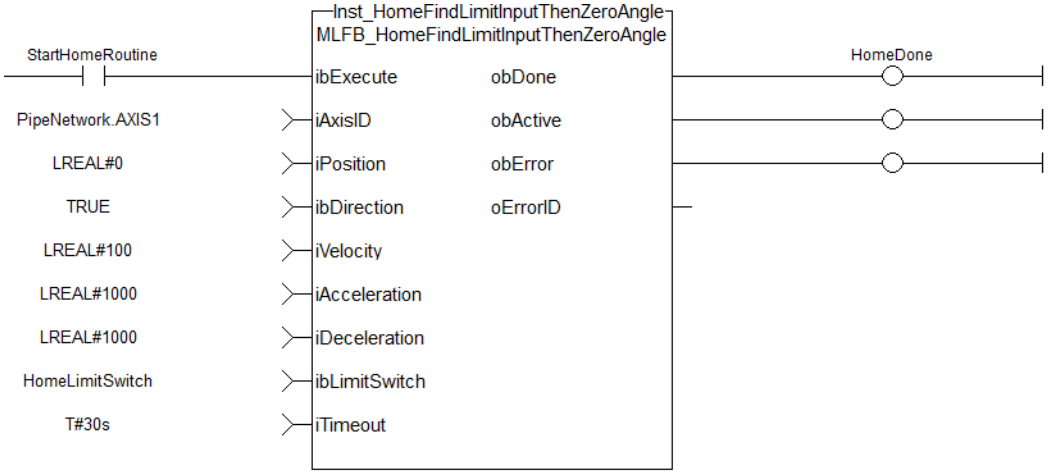


- Similar to the Find Input Limit method, the find input limit then find zero angle.
- Mode follows the same steps, but upon completion of the move, it continues to move to find the zero angle reference of the motor.

3.7.4.4.4 FBD Language Example



3.7.4.4.5 FFLD Language Example



3.7.4.4.6 IL Language Example

Not available.

3.7.4.4.7 ST Language Example

```
//Call homing function on Axis1 with preset velocity, accel, and decel values
//Start in negative direction and stop when axis hits limit switch or times
out
//after seeing limit switch, moves to zero angle and set axis position to
zero
Inst_MLFB_HomeFindLimitInputThenZeroAngle( StartHomeRoutine,
PipeNetwork.AXIS1,
0,
TRUE,
100,
1000,
1000,
HomeDigitalInput,
T#30s );

HomeDone := Inst_MLFB_HomeFindLimitInputThenZeroAngle.obDone;
```

See Also

- "MLFB_HomeFindHomeInput" (→ p. 110)
- "MLFB_HomeFindHomeInputThenZeroAngle" (→ p. 113)
- "MLFB_HomeFindLimitInput" (→ p. 115)
- "MLFB_HomeFindZeroAngle" (→ p. 119)
- "MLFB_HomeMoveUntilPosErrExceeded" (→ p. 121)
- "MLFB_HomeMoveUntilPosErrExceededThenZeroAngle" (→ p. 123)
- "MLFB_HomeUsingCurrentPosition" (→ p. 126)

3.7.4.5 MLFB_HomeFindZeroAngle

Pipe Network ✓

Function Block - Homing to zero angle reference.

3.7.4.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL				Start homing, edge-triggered.
iAxisID	DINT				ID of the axis block of the Pipe Network.
iPosition	LREAL				Reference position.
iDirectionType	DINT				• 0 (zero) = Positive. • 1 = Negative. • 2 = Shortest.
iVelocity	LREAL				Reference speed.
iAcceleration	LREAL				Reference acceleration.
iDeceleration	LREAL				Reference deceleration.
iTimeout	TIME				Time monitoring (T#0ms: off).

3.7.4.5.2 Outputs

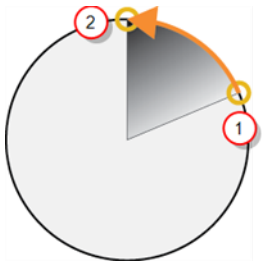
Output	Data Type	Range	Unit	Description
obDone	BOOL			Done bit.
obActive	BOOL			Active bit.
obError	BOOL			Error bit.
oErrorID	DINT			Error identifier:

ErrorID	Description
1	Axis in error.
2	Axis is not enabled.
3	Timeout exceeded or expired.
4	SDO read/write error.
5	Input parameter out of range.

3.7.4.5.3 Remarks

NOTE

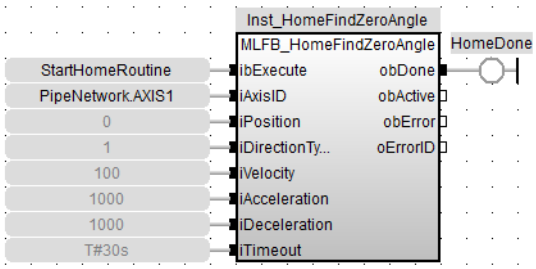
This function block is only applicable to motors with Resolver or SFD feedback.



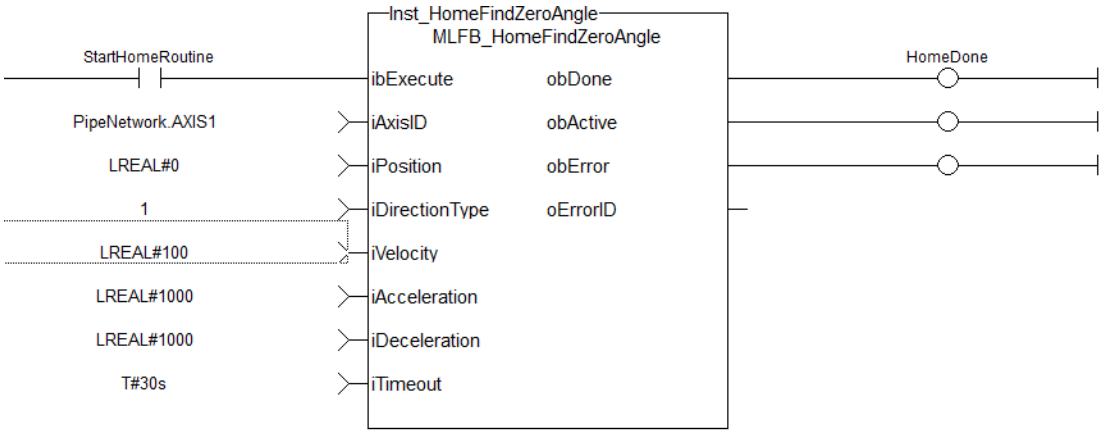
1. Start Position
2. End Position (Zero degrees)

Figure 3-9: Mode to find the zero angle reference of the motor.

3.7.4.5.4 FBD Language Example



3.7.4.5.5 FFLD Language Example



3.7.4.5.6 IL Language Example

Not available.

3.7.4.5.7 ST Language Example

```
//Call homing function on Axis1 with preset velocity, accel, and decel values
//Start in negative direction and go to zero angle or time out
//after reaching zero angle set axis position to zero
Inst_MLFB_HomeFindZeroAngle( StartHomeRoutine,
PipeNetwork.AXIS1,
0,
1,
100,
1000,
1000,
T#30s );

HomeDone := Inst_MLFB_HomeFindZeroAngle.obDone;
```

See Also

- "MLFB_HomeFindHomeInput" (→ p. 110)
- "MLFB_HomeFindHomeInputThenZeroAngle" (→ p. 113)
- "MLFB_HomeFindLimitInput" (→ p. 115)
- "MLFB_HomeFindLimitInputThenZeroAngle" (→ p. 117)
- "MLFB_HomeMoveUntilPosErrExceeded" (→ p. 121)
- "MLFB_HomeMoveUntilPosErrExceededThenZeroAngle" (→ p. 123)
- "MLFB_HomeUsingCurrentPosition" (→ p. 126)

3.7.4.6 MLFB_HomeMoveUntilPosErrExceeded

Pipe Network

Function Block - Homing to the position error is exceeded.

3.7.4.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL				Start homing, edge-triggered.
iAxisID	DINT				ID of the axis block of the Pipe Network.
iPosition	LREAL				Reference position.

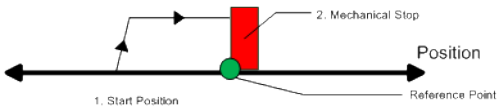
Input	Data Type	Range	Unit	Default	Description
ibDirection	BOOL				• 0 (zero) = Positive. • 1 = Negative.
iVelocity	LREAL				Reference speed.
iAcceleration	LREAL				Reference acceleration.
iDeceleration	LREAL				Reference deceleration.
iMaxPositionError	LREAL				Maximum position error.
iPeakCurrent	DINT				Peak current in mA.
iTimeout	TIME				Time monitoring (T#0ms: off).

3.7.4.6.2 Outputs

Output	Data Type	Range	Unit	Description
obDone	BOOL			Done bit.
obActive	BOOL			Active bit.
obError	BOOL			Error bit.
oErrorID	DINT			Error identifier:

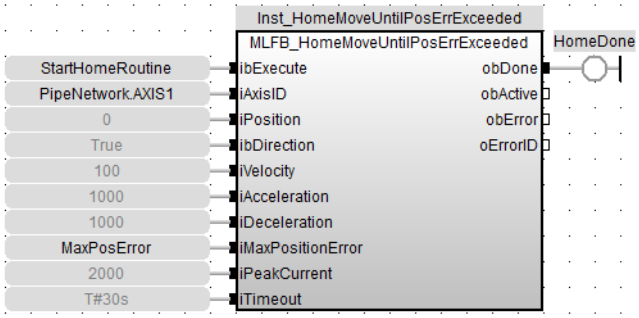
ErrorID	Description
1	Axis in error.
2	Axis is not enabled.
3	Timeout exceeded or expired.
4	SDO read/write error.
5	Input parameter out of range.

3.7.4.6.3 Remarks

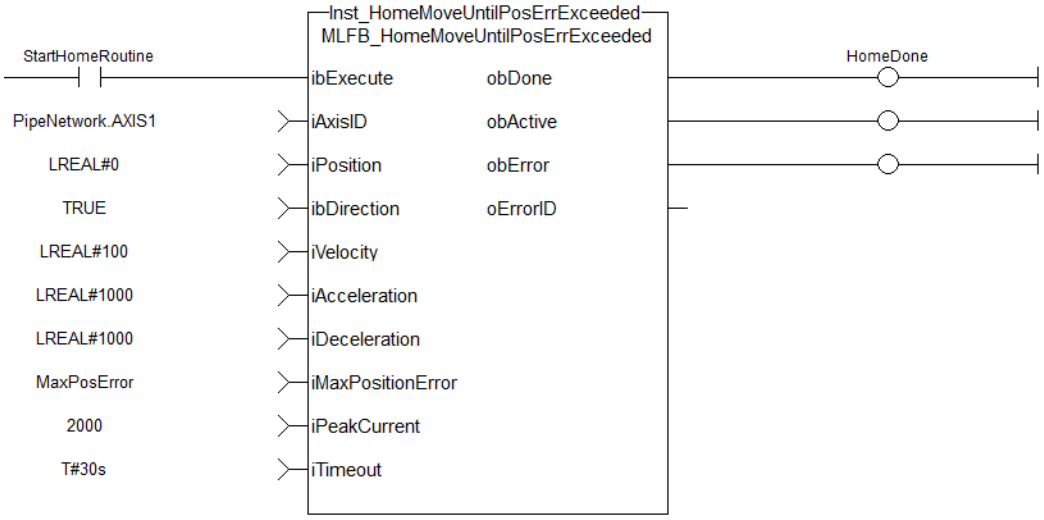


- When executed, the motor will move to the hard stop with a definable peak current.
- When the position error exceeds, the home Position is set.

3.7.4.6.4 FBD Language Example



3.7.4.6.5 FFLD Language Example



3.7.4.6.6 IL Language Example

Not available.

3.7.4.6.7 ST Language Example

```
//Call homing function on Axis1 with preset velocity, accel, and decel values
//Start in negative direction and go until position error exceeds input value
or time out
//afterterwards set axis position to zero
//function block temporarily writes new max current value to 2 Amp while home
routine active
Inst_MLFB_HomeMoveUntilPosErrExceeded( StartHomeRoutine,
   PipeNetwork.AXIS1,
   0,
   1,
   100,
   1000,
   1000,
   MaxPosError,
   2000,
   T#30s );

HomeDone := Inst_MLFB_HomeMoveUntilPosErrExceeded.obDone;
```

See Also

- "MLFB_HomeFindHomeInput" (→ p. 110)
- "MLFB_HomeFindHomeInputThenZeroAngle" (→ p. 113)
- "MLFB_HomeFindLimitInput" (→ p. 115)
- "MLFB_HomeFindLimitInputThenZeroAngle" (→ p. 117)
- "MLFB_HomeFindZeroAngle" (→ p. 119)
- "MLFB_HomeMoveUntilPosErrExceededThenZeroAngle" (→ p. 123)
- "MLFB_HomeUsingCurrentPosition" (→ p. 126)

3.7.4.7 MLFB_HomeMoveUntilPosErrExceededThenZeroAngle

Function Block - Homing to the position error is exceeded plus zero angle.

3.7.4.7.1 Inputs

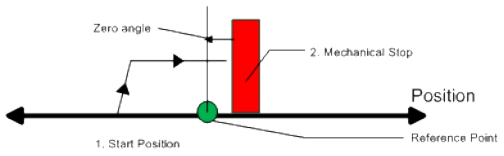
Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL				Start homing, edge-triggered.
iAxisID	DINT				ID of the axis block of the Pipe Network.
iPosition	LREAL				Reference position.
ibDirection	BOOL				• 0 (zero) = Positive. • 1 = Negative.
iVelocity	LREAL				Reference speed.
iAcceleration	LREAL				Reference acceleration.
iDeceleration	LREAL				Reference deceleration.
iMaxPositionError	LREAL				Maximum position error.
iPeakCurrent	DINT				Peak current in mA.
iTimeout	TIME				Time monitoring (T#0ms: off).

3.7.4.7.2 Outputs

Output	Data Type	Range	Unit	Description
obDone	BOOL			Done bit.
obActive	BOOL			Active bit.
obError	BOOL			Error bit.
oErrorID	DINT			Error identifier:

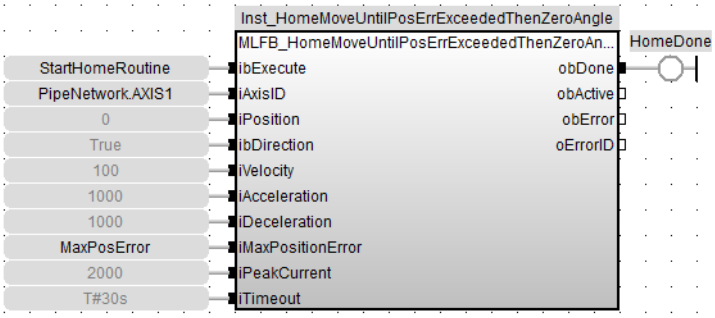
ErrorID	Description
1	Axis in error.
2	Axis is not enabled.
3	Timeout exceeded or expired.
4	SDO read/write error.
5	Input parameter out of range.

3.7.4.7.3 Remarks

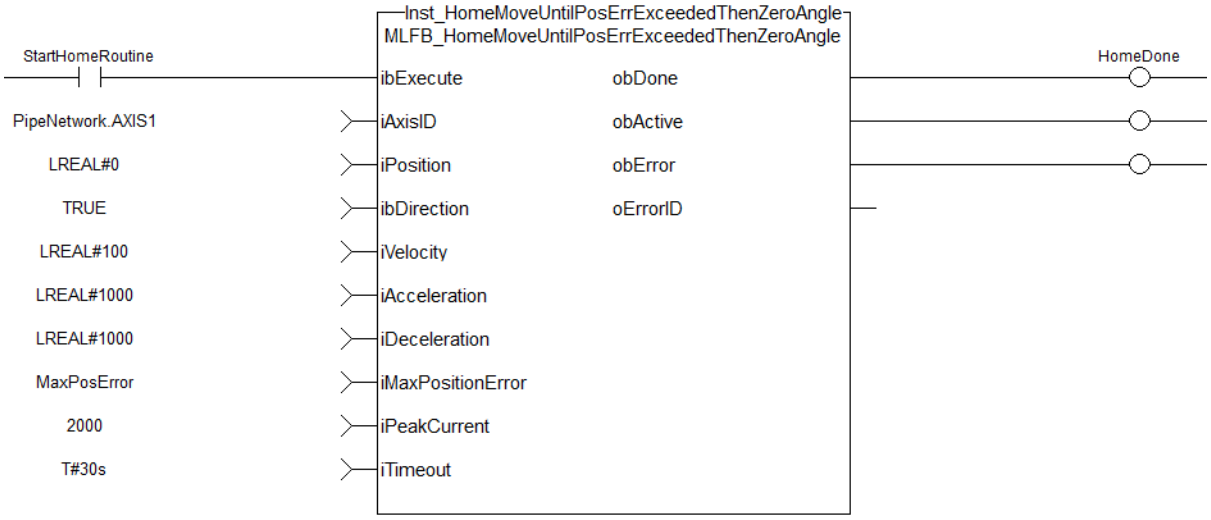


- Similar to the Move Until Position Error Exceeded method, the move until position error exceeded then find zero angle.
- Mode follows the same steps, but upon completion of the move, it continues to move to find the zero angle reference of the motor.

3.7.4.7.4 FBD Language Example



3.7.4.7.5 FFLD Language Example



3.7.4.7.6 IL Language Example

Not available.

3.7.4.7.7 ST Language Example

```
//Call homing function on Axis1 with preset velocity, accel, and decel values
//Start in negative direction and go until position error exceeds input value
or time out
//afterterwards moves to zero angle and sets axis position to zero
//function block temporarily writes new max current value to 2 Amp while home
routine active
Inst_MLFB_HomeMoveUntilPosErrExceededThenZeroAngle( StartHomeRoutine,
   PipeNetwork.AXIS1,
   0,
   1,
   100,
   1000,
   1000,
   MaxPosError,
   2000,
   T#30s );

HomeDone := Inst_MLFB_HomeMoveUntilPosErrExceededThenZeroAngle.obDone;
```

See Also

- "MLFB_HomeFindHomeInput" (→ p. 110)
- "MLFB_HomeFindHomeInputThenZeroAngle" (→ p. 113)
- "MLFB_HomeFindLimitInput" (→ p. 115)
- "MLFB_HomeFindLimitInputThenZeroAngle" (→ p. 117)
- "MLFB_HomeFindZeroAngle" (→ p. 119)
- "MLFB_HomeMoveUntilPosErrExceeded" (→ p. 121)
- "MLFB_HomeUsingCurrentPosition" (→ p. 126)

3.7.4.8 MLFB_HomeUsingCurrentPosition

Pipe Network

Function Block - Homing using current position.

3.7.4.8.1 Inputs

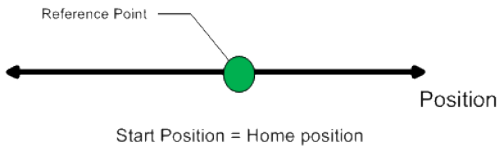
Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL				Start homing, edge-triggered.
iAxisID	DINT				ID of the axis block of the Pipe Network.
iPosition	LREAL				Reference position.

3.7.4.8.2 Outputs

Output	Data Type	Range	Unit	Description
obDone	BOOL			Done bit.
obActive	BOOL			Active bit.
obError	BOOL			Error bit.
oErrorID	DINT			Error identifier:

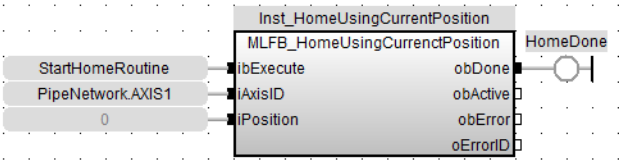
ErrorID	Description
1	Axis in error.
2	Axis is not enabled.
3	Timeout exceeded or expired.
4	SDO read/write error.
5	Input parameter out of range.

3.7.4.8.3 Remarks

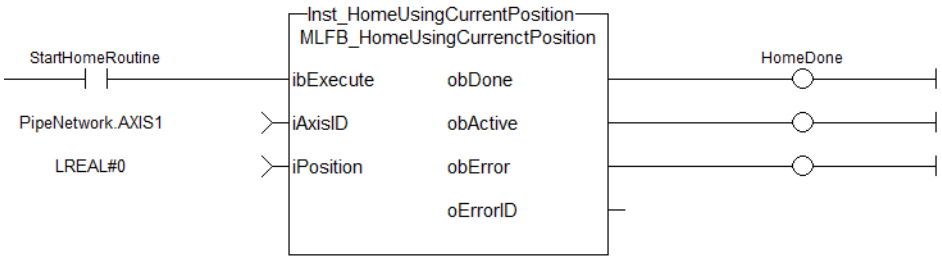


- Use this parameter to set the value of the home position other then zero.
  - This allows you to offset your home reference away from zero.
- Using the current position is the most basic homing method.
  - This method uses the current position of the motor as the home point reference.

3.7.4.8.4 FBD Language Example



3.7.4.8.5 FFLD Language Example



3.7.4.8.6 IL Language Example

Not available.

3.7.4.8.7 ST Language Example

```
//No movement, set current axis position to position input in this case zero
Inst_MLFB_HomeUsingCurrenctPosition( StartHomeRoutine, PipeNetwork.AXIS1, 0
);
HomeDone := Inst_MLFB_HomeUsingCurrenctPosition.obDone;
```

See Also

- "MLFB_HomeFindHomeInput" (→ p. 110)
- "MLFB_HomeFindHomeInputThenZeroAngle" (→ p. 113)
- "MLFB_HomeFindLimitInput" (→ p. 115)
- "MLFB_HomeFindLimitInputThenZeroAngle" (→ p. 117)
- "MLFB_HomeFindZeroAngle" (→ p. 119)
- "MLFB_HomeMoveUntilPosErrExceeded" (→ p. 121)
- "MLFB_HomeMoveUntilPosErrExceededThenZeroAngle" (→ p. 123)

3.7.5 MLFB_Jog

Pipe Network

Function - Define to jog an axis in the selected direction at a defined speed.

3.7.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	No range	N/A	No default	Enables execution. Used in the FFLD editor only.
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block. The **AxisID** selects the axis to jog.
JogPlus	BOOL	0, 1	N/A	No default	Enables a jog in the plus direction.
JogMinus	BOOL	0, 1	N/A	No default	Enables a jog in the minus direction.
Speed	LREAL	No range	User unit/sec	No default	Rate the axis moves.

3.7.5.2 Outputs

Output	Data Type	Range	Unit	Description
InMotion	BOOL		N/A	Jogging is active when TRUE.

3.7.5.3 Remarks

- This function or function block can be used with PLCopen Motion engines.
- This function or function block is used to command motion in a designated direction at a defined rate.
  - This may be used where continuous motion is required (e.g., a conveyor system or in a setup mode for manually jogging the axis).
- The **JogPlus** and **JogMinus** inputs select the direction the motion occurs in.
  - Motion starts when the **JogPlus** or **JogMinus** input is TRUE.
    Motion stops when the input is FALSE.
  - Only one of these inputs should be enabled at a time.
  - If both are selected, the motion stops.
- If another motion is active when the jog is requested, that motion is aborted and the jog starts.
- The En input (FFLD editor only) must be high.
  - Typically wired to the rail.

This image shows the function or function block I/O.

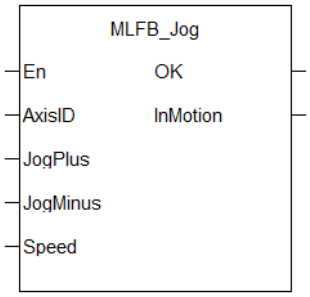
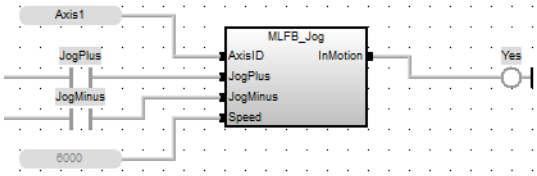
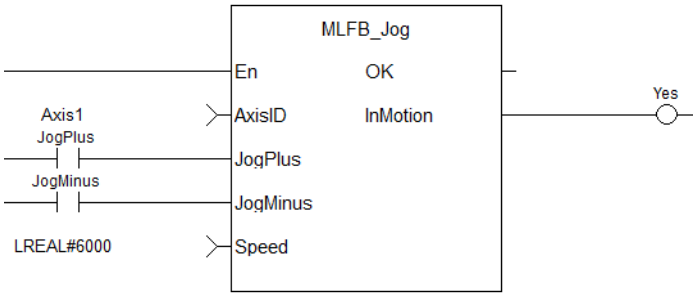


Figure 3-10: MLFB_Jog

3.7.5.4 FBD Language Example



3.7.5.5 FFLD Language Example



3.7.5.6 IL Language Example

Not available.

3.7.5.7 ST Language Example

```
//Jog Axis1 at 6000 user units a second when JogPlus or JogMinus variables
are TRUE
//Stop motion on falling edge of either variable
MLFB_Jog( PipeNetwork.AXIS1, JogPlus, JogMinus, 6000 );
```

See Also

"MLAxisMoveVel" (→ p. 250)

3.7.6 PLS - UDFB - Motion/Pipe Network

These are the UDFB Motion/Pipe Network PLS function and function blocks.

Name	Description
"MLFB_PlsPosFw" (→ p. 129)	Forward position range indicator.
"MLFB_PlsPosFwBw" (→ p. 130)	Forward/backward position range indicator.
"MLFB_PlsTimeFw" (→ p. 132)	Forward/backward Position/Time range indicator.

3.7.6.1 MLFB_PlsPosFw

Function Block - Forward position range indicator.

3.7.6.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL				Enable PLS.
iDedicatedCmpID	DINT				ID of the dedicated comparator.
iStartPos	LREAL				Start position of PLS.
iEndPos	LREAL				End position of PLS.
iDelayTime	TIME				Delay time for compensation.
ibForce	BOOL				Force PLS.

3.7.6.1.2 Outputs

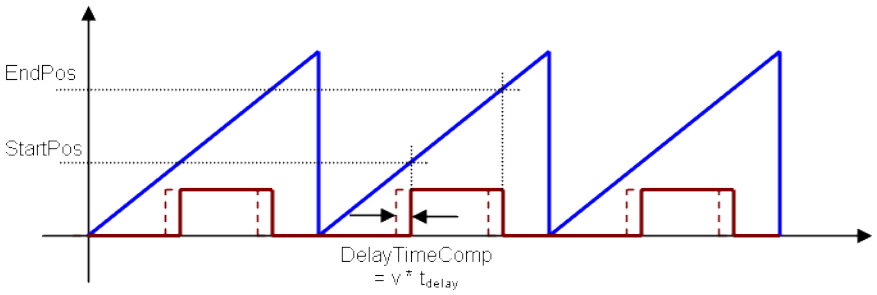
Output	Data Type	Range	Unit	Description
oPLS	BOOL		N/A	Position limit switch.

3.7.6.1.3 Remarks

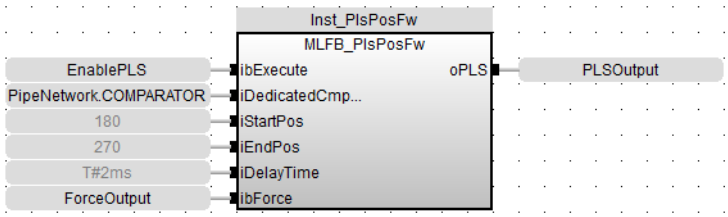
- Used in the command position path with ascending position.
- A dedicated comparator pipe block is needed.
- The Boolean output **oPLS** is set to:
  - TRUE if the position has crossed the start position.
  - FALSE if the position has crossed the end position.

- The function block is executed cyclically.
  - The modulo position is considered.
- The function block has the possibility to compensate a delay time of the connected device (e.g., glue nozzles).

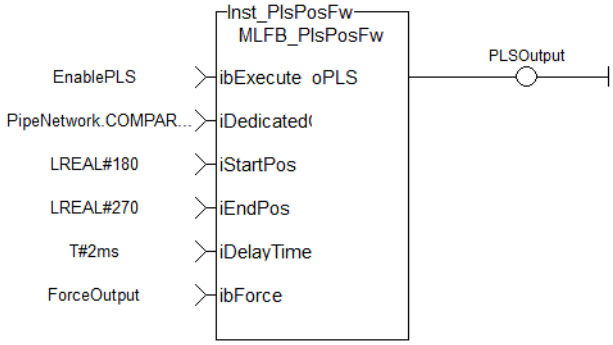
3.7.6.1.3.1 Timing



3.7.6.1.4 FBD Language Example



3.7.6.1.5 FFLD Language Example



3.7.6.1.6 IL Language Example

Not available.

3.7.6.1.7 ST Language Example

```
//PLSOutput is True when chosen comparator is between 180 and 270 with a
T#2ms delay
//Can also force the output to be true with ForceOuput variable
Inst_MLFB_PlsPosFw( EnablePLS, PipeNetwork.COMPARATOR, 180, 270, T#2ms,
ForceOutput );
PLSOutput := Inst_MLFB_PlsPosFw.oPLS;
```

3.7.6.2 MLFB_PlsPosFwBw

Pipe Network ✓

Function Block - Forward/backward position range indicator.

3.7.6.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL				Enable PLS.
iAnyModuloBlkID	DINT				Any modulo pipe network block ID.
iStartPos	LREAL				Start position of PLS.
iEndPos	LREAL				End position of PLS.
iDelayTime	TIME				Delay time for compensation.
iHysteresis	LREAL				Hysteresis.
ibForce	BOOL				Force PLS.

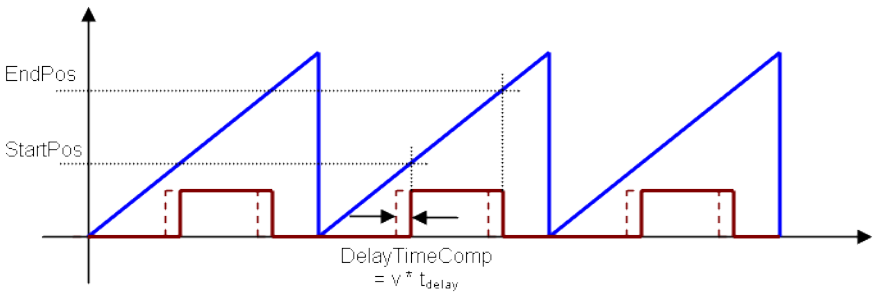
3.7.6.2.2 Outputs

Output	Data Type	Range	Unit	Description
oPLS	BOOL		N/A	Position limit switch.

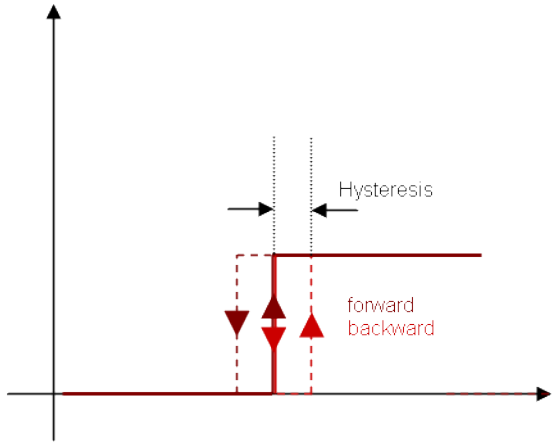
3.7.6.2.3 Remarks

- Used in the command or actual position path, e.g., sampler pipe with noisy position, in both directions.
- Any modulo pipe block is needed, which can also be used for another instance of this UDFB.
- It is also possible to define a hysteresis for switching on and off of the PLS.
- The Boolean output **oPLS** is set to:
  - TRUE if the position has crossed the start position.
  - FALSE if the position has crossed the end position.
- The function block is executed cyclically.
  - The modulo position is considered.
- The function block has the possibility to compensate a delay time of the connected device (e.g., glue nozzles).

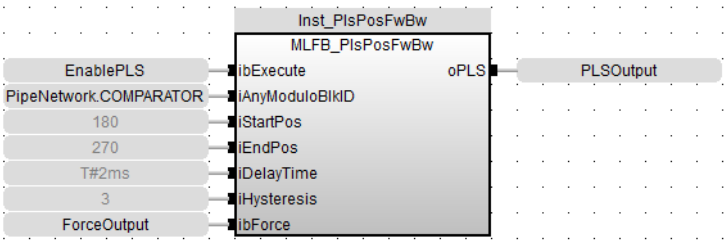
3.7.6.2.3.1 Timing



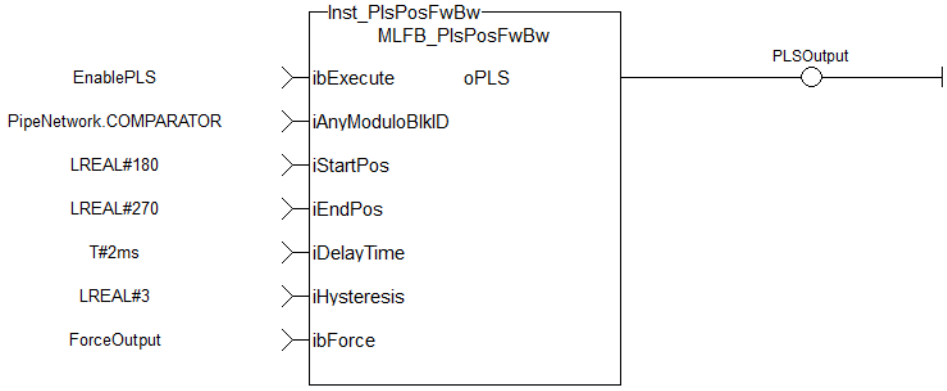
3.7.6.2.3.2 Hysteresis



3.7.6.2.4 FBD Language Example



3.7.6.2.5 FFLD Language Example



3.7.6.2.6 IL Language Example

Not available.

3.7.6.2.7 ST Language Example

```
//PLSOutput is True when chosen comparator is between 180 and 270 with a
T#2ms delay
//Can also force the output to be true with ForceOuput variable
//Hysteresis is on for 3 user units in case direction changes around start
point
Inst_MLFB_PlsPosFwBw( EnablePLS, PipeNetwork.COMPARATOR, 180, 270, T#2ms, 3,
ForceOutput );
PLSOutput := Inst_MLFB_PlsPosFwBW.oPLS;
```

3.7.6.3 MLFB_PlsTimeFw

Function Block - Forward/backward Position/Time range indicator.

3.7.6.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ibExecute	BOOL				Enable PLS.
iDedicatedCmpID	DINT				ID of the dedicated comparator.
iStartPos	LREAL				Start position of PLS.
iOnTime	TIME				Time the PLS is on.
iDelayTime	TIME				Delay time for compensation.
ibForce	BOOL				Force PLS.

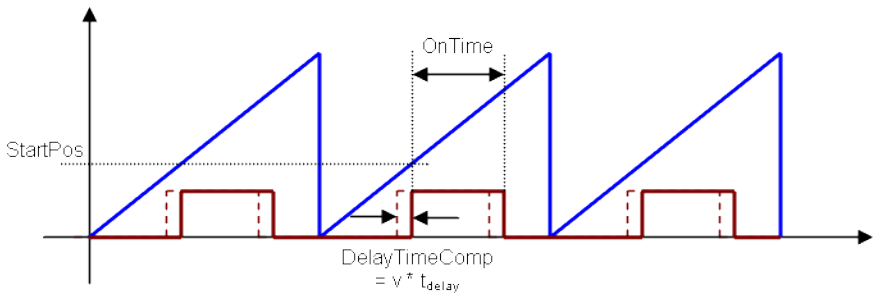
3.7.6.3.2 Outputs

Output	Data Type	Range	Unit	Description
oPLS	BOOL		N/A	Position limit switch.

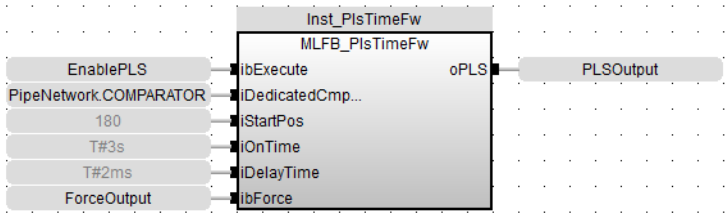
3.7.6.3.3 Remarks

- Used in the command position path with ascending position.
- A dedicated comparator pipe block is needed.
- The Boolean output **oPLS** is set to:
  - TRUE if the position has crossed the start position.
    - A timer with **iOnTime** must be started.
  - When the timer has expired the output is set to FALSE.
- The function block is executed cyclically.
  - The modulo position is considered.
- The function block has the possibility to compensate a delay time of the connected device (e.g., glue nozzles).

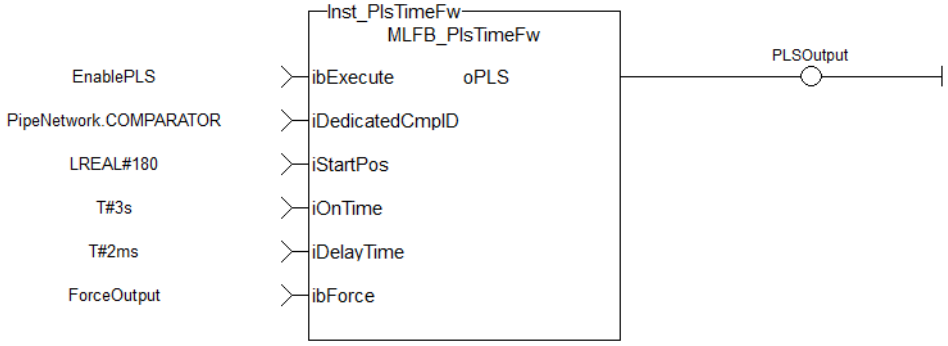
3.7.6.3.3.1 Timing



3.7.6.3.4 FBD Language Example



3.7.6.3.5 FFLD Language Example



3.7.6.3.6 IL Language Example

Not available.

3.7.6.3.7 ST Language Example

```
//PLSOutput is True when chosen comparator passes 180 for 3 seconds a T#2ms
delay
//Can also force the output to be true with ForceOuput variable
Inst_MLFB_PlsTimeFw( EnablePLS, PipeNetwork.COMPARATOR, 180, T#3s, T#2ms,
ForceOutput );
PLSOutput := Inst_MLFB_PlsTimeFw.oPLS;
```

3.8 UDFB - Motion / PLCopen

These are the Kollmorgen UDFB Motion/PLCopen function and function blocks.

Name	Description
MCFB_AKDFault	Outputs AKD drive fault information.
MCFB_AKDFaultLookup	String message of the corresponding AKD drive fault number.
MCFB_DriveFault	Returns the fault status, number, and description of the requested axis mapped to a Kollmorgen drive.
MCFB_ECATRestart	Reinitializes the EtherCAT network and the motion engine.
MCFB_GearedWebTension	Facilitates dancer and tension control in an electronic geared master/slave machine design.
MCFB_Jog	Define to jog an axis in the selected direction at a defined speed.
MCFB_StepAbsolutes	Homing by setting Actual Position to the position of the feedback.
MCFB_StepAbsSwitch	Homing to a home switch.
MCFB_ StepAbsSwitchFastInput	Fast homing to a home switch.
MCFB_StepBlock	Homing to a physical object, mechanically blocking the movement.
MCFB_StepLimitSwitch	Homing to a limit switch.
MCFB_ StepLimitSwitchFastInput	Fast homing to a limit switch.
MCFB_StepRefPulse	Homing to a zero angle reference.

3.8.1 Drive Fault Reporting - UDFB - Motion/PLCopen

These are the Kollmorgen UDFB Motion/PLCopen Drive Fault Reporting function blocks.

Name	Description
"MCFB_AKDFault" (→ p. 135)	Outputs AKD drive fault information.
"MCFB_AKDFaultLookup" (→ p. 136)	String message of the corresponding AKD drive fault number.
"MCFB_DriveFault" (→ p. 137)	Returns the fault status, number, and description of the requested axis mapped to a Kollmorgen drive.

3.8.1.1 MCFB_AKDFault

PLCopen ✓

Function Block - Outputs AKD drive fault information.

3.8.1.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the Kollmorgen UDFB. Used in the FFLD editor only.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function. • See AXIS_REF Structure for more information. • See Axis Name and Number for more information.

3.8.1.1.2 Outputs

Output	Data Type	Range	Unit	Description
FAULT	BOOL	0, 1	N/A	TRUE if the selected drive currently has a Fault.
FaultNumber	DINT	100, 999	N/A	The AKD drive fault number.

3.8.1.1.3 Remarks

TIP

This function block lists the **highest priority** fault as displayed on the AKD.
The "FB_AKDFltRpt" (→ p. 78) function block lists faults in the order they occur.

- This function or function block can be used with PLCopen Motion engines.
- The FAULT output returns TRUE when the selected drive goes into a fault state.
- The **FaultNumber** is the same number as reported on the display of the AKD drive.

This image shows the function or function block I/O.

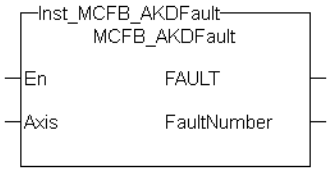


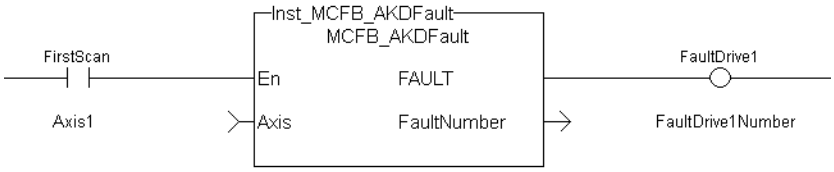
Figure 3-11: MCFB_AKDFault

3.8.1.1.3.1 Usage

- Provide drive fault information the application program uses to determine next steps.
  - Example: Perform a machine-controlled stop or perform an immediate disable of the servo drives.
- In the application program, sends output fault information from this UDFB to the HMI for review by the machine operator.

3.8.1.1.4 FBD Language Example

3.8.1.1.5 FFLD Language Example



3.8.1.1.6 IL Language Example

Not available.

3.8.1.1.7 ST Language Example

```
            //Execute and Read the Function Block

Inst_MCFB_AKDFault( Axis1(*lib:AXIS_REF*) );
FaultDrive1 := Inst_MCFB_AKDFault.FAULT;
FaultDrive1Number := Inst_MCFB_AKDFault.FaultNumber;

FaultDrive1Description := MCFB_AKDFaultLookup( FaultDrive1Number(*DINT*) );
```

See Also

- "FB_AKDFltRpt" (→ p. 78)
- "MC_ReadStatus" (→ p. 442) (PLCopen Motion Engine)
- "MCFB_AKDFaultLookup" (→ p. 136)

3.8.1.2 MCFB_AKDFaultLookup

PLCopen

Function Block - String message of the corresponding AKD drive fault number.

3.8.1.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the Kollmorgen UDFB. Used in the FFLD editor only.
FaultNumber	DINT	No range	N/A	No default	The AKD drive fault number.

3.8.1.2.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	0, 1	N/A	TRUE if there is a match for the **FaultNumber**.
FaultDescription	STRING	N/A	N/A	Description of the fault.

3.8.1.2.3 Remarks

- This function or function block can be used with PLCopen Motion engines.
- The **FaultNumber** is the same number as reported on the display of the AKD drive.
- OK output turns TRUE when the there is a match for the **FaultNumber**.
- The **FaultDescription** displays the corresponding text string.

This image shows the function or function block I/O.

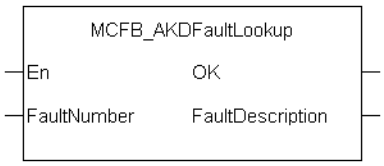


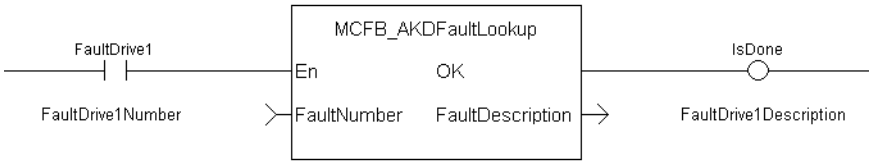
Figure 3-12: MCFB_AKDFaultLookup

3.8.1.2.3.1 Usage

- Provide drive fault information the application program uses to determine next steps.
  - Example: Perform a machine-controlled stop or perform an immediate disable of the servo drives.
- In the application program, sends output fault information from this UDFB to the HMI for review by the machine operator.

3.8.1.2.4 FBD Language Example

3.8.1.2.5 FFLD Language Example



3.8.1.2.6 IL Language Example

Not available.

3.8.1.2.7 ST Language Example

```
            //Execute and Read the Function Block

Inst_MCFB_AKDFault( Axis1(*lib:AXIS_REF*) );
FaultDrive1 := Inst_MCFB_AKDFault.FAULT;
FaultDrive1Number := Inst_MCFB_AKDFault.FaultNumber;

FaultDrive1Description := MCFB_AKDFaultLookup( FaultDrive1Number(*DINT*) );
```

See Also

- "FB_AKDFltRpt" (→ p. 78)
- "MC_ReadStatus" (→ p. 442) (PLCopen Motion Engine)
- "MCFB_AKDFault" (→ p. 135)

3.8.1.3 MCFB_DriveFault

PLCopen

Function Block - Returns the fault status, number, and description of the requested axis mapped to a Kollmorgen drive.

3.8.1.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the Kollmorgen UDFB. Used in the FFLD editor only.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function. • See AXIS_REF Structure for more information. • See Axis Name and Number for more information.

3.8.1.3.2 Outputs

Output	Data Type	Range	Unit	Description
FAULT	BOOL	0, 1	N/A	TRUE if the selected drive currently has a Fault.

FaultNumber	DINT	No range	N/A	If the axis is:
				• S300/S700
				  • Three-digit fault identifier.
				  • See S300 & S700 Errors and Warnings on KDN for a list of fault codes.
				• AKD
				  • Three-digit fault identifier.
				  • See Fault and Warning Messages for a list of fault codes.
				• AKD2G
				  • Four-digit fault identifier.
				  • See Faults and Warnings View for a list of fault codes.
				• AKT2G Stepper
				  • Drive Status word (bitmask).
				  • See "Bitmask Causes" (→ p. 138) for more information.

FaultDescription	STRING	N/A	N/A	Description of the fault.

3.8.1.3.2.1 Bitmask Causes

Bit	Description	Cause
0	Saturated	Drive stage operates with maximum duty cycle.
1	Over temperature.	Internal temperature is higher than 80 °C.
2	Torque overload.	Motor current is higher than the rated current.
3	Under voltage.	Motor supply voltage is either: • 20% lower than the configured nominal voltage (warning). • Less than 8V.
4	Over voltage.	Motor supply voltage is 10% higher than the configured nominal voltage.

Bit	Description	Cause
5	Short circuit A.	Short circuit in motor coil A.
6	Short circuit B.	Short circuit in motor coil B.
7	No control power.	Control voltage at the power contacts is less than 12V.
8	Misc. error.	Either: • The terminal initialization failed. • Supply voltage is less than 8V. • Internal terminal temperature is higher than 100 °C.
9	Configuration error.	CoE change has not yet been adopted into the current configuration.

3.8.1.3.3 Remarks

IMPORTANT

This function block requires "FB_S700FltRpt" (→ p. 87), "MCFB_AKDFault" (→ p. 135), and "MCFB_AKDFaultLookup" (→ p. 136) subprograms imported to project to compile and function.

- The FAULT output returns TRUE when the selected drive goes into a fault state.
- The fault number and description depend on the drive type mapped to the axis.
  - If the drive is an:
    - AKD or AKD2G, the fault number is the same number as reported on the display of the drive.
    - AKT2G Stepper, the fault number represents the drive status word.
      - This word is a bitmask that represents the various error conditions.

This image shows the function or function block I/O.

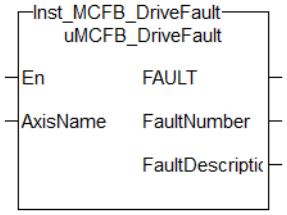
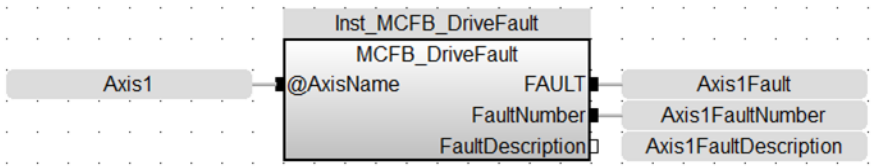


Figure 3-13: MCFB_DriveFault

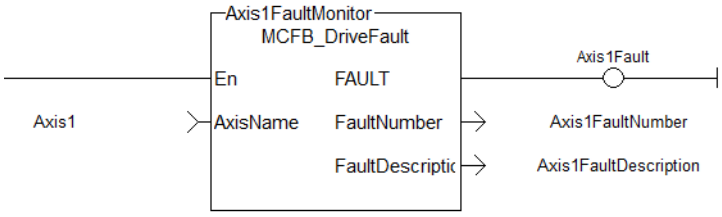
3.8.1.3.3.1 Usage

- Provide drive fault information the application program uses to determine next steps.
  - Example: Perform a machine-controlled stop or perform an immediate disable of the servo drives.
- In the application program, sends output fault information from this UDFB to the HMI for review by the machine operator.

3.8.1.3.4 FBD Language Example



3.8.1.3.5 FFLD Language Example



3.8.1.3.6 IL Language Example

Not available.

3.8.1.3.7 ST Language Example

```
//Execute and Read the Function Block
Inst_MCFB_DriveFault( Axis1 );
Axis1Fault := Inst_MCFB_DriveFault.FAULT;
Axis1FaultNumber := Inst_MCFB_DriveFault.FaultNumber;
Axis1FaultDescription := Inst_MCFB_DriveFault.FaultDescription;
```

Related Functions

- "FB_S700FltRpt" (→ p. 87)
- "MC_ReadStatus" (→ p. 442) (PLCopen Motion Engine)
- "MCFB_AKDFault" (→ p. 135)
- "MCFB_AKDFaultLookup" (→ p. 136)

3.8.2 MCFB_ECATRestart

PLCopen

Function Block - Reinitializes the EtherCAT network and the motion engine.

3.8.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
iEN	BOOL	0, 1	N/A	No default	Enables the Kollmorgen UDFB. Used in the FFLD editor only.
iRSTERR	BOOL	1, 256	0, 1	No default	Clears the motion engine and EtherCAT network errors in case of any faults.

3.8.2.2 Outputs

Output	Data Type	Range	Unit	Description
oOK	BOOL	0, 1	N/A	Function block activated status.
oDONE	BOOL	0, 1	N/A	Execution complete.
oERR	BOOL	0, 1	N/A	TRUE if the system initialization fails.

3.8.2.3 Remarks

- This function block clears motion engine errors, motion bus driver errors and EtherCAT network errors before reinitializing the motion engine, if requested to do so.
- This UDFB allows the EtherCAT and motion engines to restart without having to restart the entire project.

Examples:

- EtherCAT network wire is replaced or accidentally disconnected.
- Axis setup parameters defined by CreateAxis and/or InitAxis function need to be changed while the application is running.

This image shows the function or function block I/O.

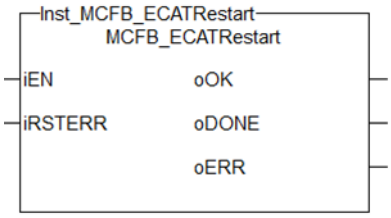
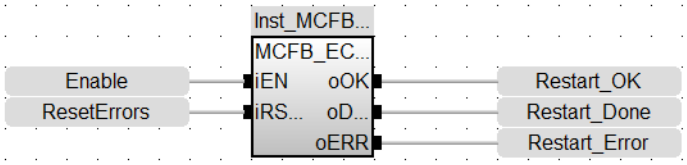
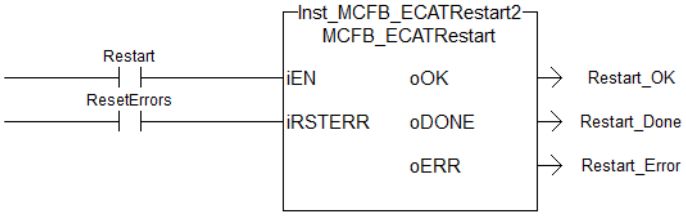


Figure 3-14: MCFB_ECATRestart

3.8.2.4 FBD Language Example



3.8.2.5 FFLD Language Example



3.8.2.6 IL Language Example

Not available.

3.8.2.7 ST Language Example

```
Inst_MCFB_ECATRestart( Restart, ResetErrors );
IF Inst_MCFB_ECATRestart.oDONE THEN
   RestartComplte:=1;
End_IF;
```

See Also

- "ClearCtrlErrors" (→ p. 652)
- "MLMotionInit" (→ p. 549)
- "MLMotionRstErr" (→ p. 551)
- "MLMotionStart" (→ p. 552)

3.8.3 MCFB_StepAbsSwitchFastInput

PLCopen

Function Block - Fast homing to a home switch.

3.8.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	Request the homing step procedure at the rising edge. Outputs are reset when execute input is FALSE.
AxisID	AXIS_REF	1, 256	N/A	No default	Structure for specified Axis designated to home. See AXIS_REF Structure for more information.
Direction	BOOL	0, 1	N/A	No default	Defines the axis homing direction. • 0 (zero) = Clockwise rotation. • 1 = Counterclockwise rotation.
SwitchMode	DINT	0, 3	N/A	No default	Switch state to complete homing. • 0 (zero) = Rising edge of sensor. • 1 = Falling edge of sensor. • 2 = Rising edge when traveling in positive direction but falling edge in negative direction. • 3 = Falling edge when traveling in negative direction but rising edge in positive direction.
Velocity	LREAL	No range	User unit/sec	No default	Commanded velocity for the homing move.
Acceleration	LREAL	No range	User unit/sec^2	No default	Commanded acceleration for the homing move.
Deceleration	LREAL	No range	User unit/sec^2	No default	Commanded deceleration for the homing move.
Jerk	LREAL	No range	User unit/sec^3	No default	Commanded jerk for the homing move. If 0 (zero), trapezoidal acc/dec is used.
TorqueLimit	LREAL	No range	User units	No default	Maximum torque applied for the homing move. • Entered in thousandths of maximum torque. • Example: 250 is 250/1000, or 25%.

Input	Data Type	Range	Unit	Default	Description
TimeLimit	TIME	No range	Sec	No default	Maximum time for homing move to complete. • If exceeded, the homing procedure errors out. • 0 (zero) = no time limit.
DistanceLimit	LREAL	No range	User units	No default	Maximum distance for homing move to complete. • If exceeded, the homing procedure errors out. • 0 (zero) = no distance limit.
FastInputNum	BOOL	0, 1	N/A	No default	• 0 (zero) = first fast input (X7 pin 10). • 1 = second fast input (X7 pin 9).
PosLimitSwitch	BOOL	0, 1	N/A	No default	The positive direction limit switch input I/O point.
NegLimitSwitch	BOOL	0, 1	N/A	No default	The negative direction limit switch input I/O point.
BufferMode	SINT	0, 5	N/A	No default	Define the homing move start action. • 0 = Abort. • 1 = Buffer. • 2 = Blend to active. • 3 = Blend to next. • 4 = Blend to low velocity. • 5 = Blend to high velocity.

3.8.3.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL		N/A	Indicates the move completed successfully. The Command Position has reached the endpoint.
Busy	BOOL		N/A	High from the moment the Execute input goes high until the time the move is ended.
Active	BOOL		N/A	Set when the function block is active.
CommandAborted	BOOL		N/A	Indicates the move was aborted.
Error	BOOL		N/A	Signals an error has occurred within the function block.

<table>
<tr><th>Output</th><th>Data Type</th><th>Range</th><th>Unit</th><th>Description</th></tr>
<tr><td>ErrorID</td><td>INT</td><td></td><td>N/A</td><td>Indicates the error if the Error output is set to TRUE.
Error identifier:
<table><tr><th>ErrorID</th><th>Description</th></tr><tr><td>1</td><td>TimeLimit exceeded.</td></tr><tr><td>2</td><td>DistanceLimit exceeded.</td></tr><tr><td>3</td><td>TorqueLimit exceeded.</td></tr><tr><td>4</td><td>Axis error stop state.</td></tr><tr><td>5</td><td>Axis not enabled.</td></tr><tr><td>6</td><td>Invalid inputs for Velocity-Acceleration-Deceleration.</td></tr></table></td></tr>
<tr><td>TriggerPosition</td><td>LREAL</td><td>No range</td><td>User units</td><td></td></tr>
</table>

3.8.3.3 Remarks

- Performs a homing function by searching for an absolute positioned external physical switch.
  - The switch must be connected to one of the two fast inputs on the Axis' AKD drive.
- An Absolute Switch has two Off (or On) areas.

This image shows the function or function block I/O.

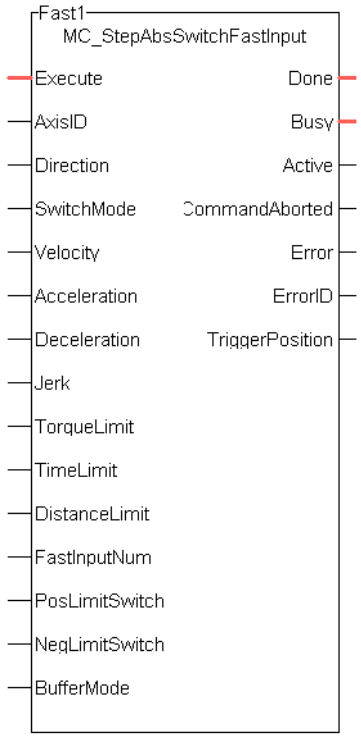
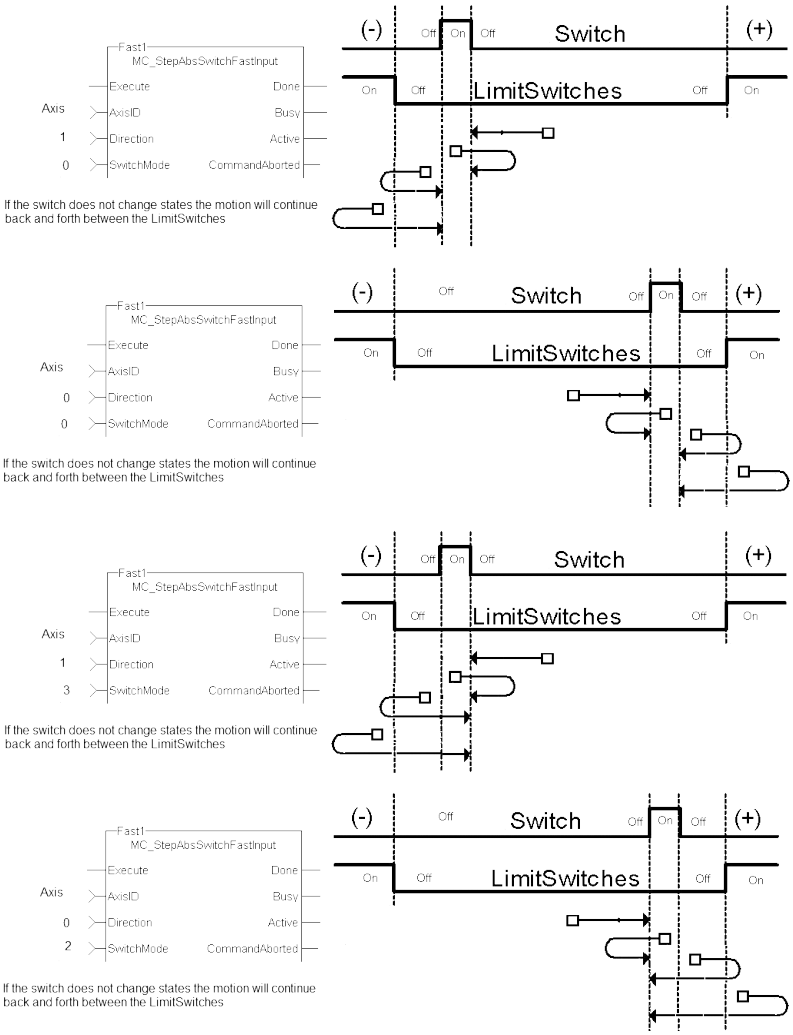


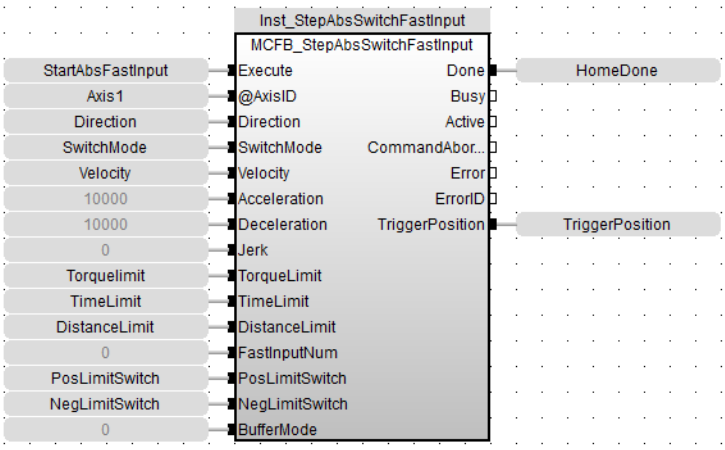
Figure 3-15: MCFB StepAbsSwitchFastInput

3.8.3.3.1 Usage

- The homing is commanded in the most likely direction where the sensor can be found.
- If any LimitSwitch is found during Homing (any of them), a special process is started in the opposite direction.
  - The AbsSwitch is searched to switch Off (or On, depending on SwitchMode setting).
- The Edge (passed by) and homing process is restarted in the original direction and with the same conditions.
  - This ensures the end conditions are always the same.

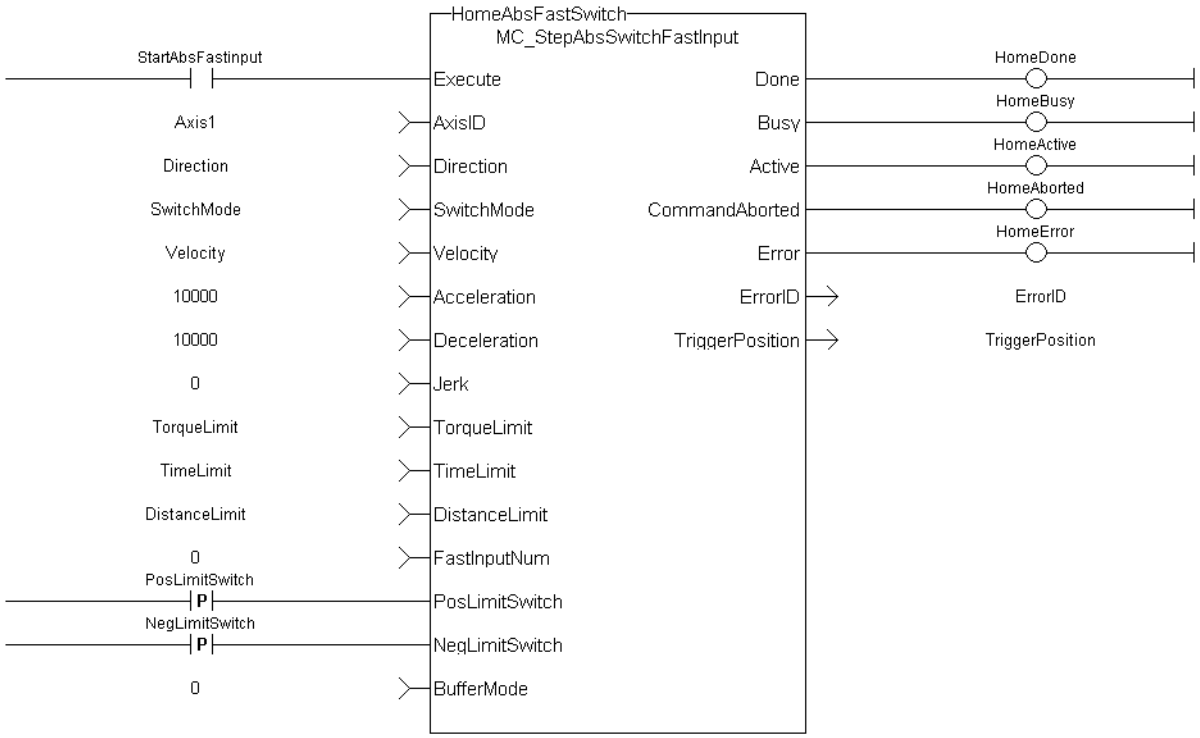


3.8.3.4 FBD Language Example



3.8.3.5 FFLD Language Example

PosLimitSwitch, NegLimitSwitch are declared I/O points.



3.8.3.6 IL Language Example

Not available.

3.8.3.7 ST Language Example

```
Execute_1 :=1;
(*Positive_Switch and Negitive_Switch are physical hardware in Dictionary. *)

Inst_MC_StepAbsSwitchFastInput( Execute_1, Axis1, 0, 0,
10000.0,Acceleration:=10000.0, 10000.0, 0, 0, 0, 0, 0, Positive_Switch ,
Negitive_Switch ,  0)

HomeComplete := Inst_MC_StepAbsSwitchFastInput.Done;
HomeBusy := Inst_MC_StepAbsSwitchFastInput.Busy;
HomeActive := Inst_MC_StepAbsSwitchFastInput.Active;
HomeAborted := Inst_MC_StepAbsSwitchFastInput.CommandAborted;
HomeError := Inst_MC_StepAbsSwitchFastInput.Error;
HomeErrorID := Inst_MC_StepAbsSwitchFastInput.ErrorID;
HomeTriggerPosition := Inst_MC_StepAbsSwitchFastInput.TriggerPosition;
```

See Also

"MCFB_StepLimitSwitchFastInput" (→ p. 146)

3.8.4 MCFB_StepLimitSwitchFastInput

PLCopen

Function Block - Fast homing to a limit switch.

3.8.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	Request the homing step procedure at the rising edge. Outputs are reset when execute input is FALSE.
AxisID	AXIS_REF	1, 256	N/A	No default	Structure for specified Axis designated to home. See AXIS_REF Structure for more information.
Direction	BOOL	0, 1	N/A	No default	Defines the axis homing direction. • 0 (zero) = Clockwise rotation. • 1 = Counterclockwise rotation.
LimitSwitchMode	DINT	0, 1	N/A	No default	Limit switch state to complete homing. • 0 (zero) = Rising edge of sensor. • 1 = Falling edge of sensor.
Velocity	LREAL	No range	User unit/sec	No default	Commanded velocity for the homing move.
Acceleration	LREAL	No range	User unit/sec^2	No default	Commanded acceleration for the homing move.
Deceleration	LREAL	No range	User unit/sec^2	No default	Commanded deceleration for the homing move.
Jerk	LREAL	No range	User unit/sec^3	No default	Commanded jerk for the homing move. If 0 (zero), trapezoidal acc/dec is used.
TorqueLimit	LREAL	No range	User units	No default	Maximum torque applied for the homing move. • Entered in thousandths of maximum torque. • Example: 250 is 250/1000, or 25%.
TimeLimit	TIME	No range	Sec	No default	Maximum time for homing move to complete. • If exceeded, the homing procedure errors out. • 0 (zero) = no time limit.

Input	Data Type	Range	Unit	Default	Description
DistanceLimit	LREAL	No range	User units	No default	Maximum distance for homing move to complete. • If exceeded, the homing procedure errors out. • 0 (zero) = no distance limit.
FastInputNum	BOOL	0, 1	N/A	No default	• 0 (zero) = first fast input (X7 pin 10). • 1 = second fast input (X7 pin 9).
BufferMode	SINT	0, 5	N/A	No default	Define the homing move start action. • 0 = Abort. • 1 = Buffer. • 2 = Blend to active. • 3 = Blend to next. • 4 = Blend to low velocity. • 5 = Blend to high velocity.

3.8.4.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL		N/A	Indicates the move completed successfully. The Command Position has reached the endpoint.
Busy	BOOL		N/A	High from the moment the Execute input goes high until the time the move is ended.
Active	BOOL		N/A	Set when the function block is active.
CommandAborted	BOOL		N/A	Indicates the move was aborted.
Error	BOOL		N/A	Signals an error has occurred within the function block.
ErrorID	INT		N/A	Indicates the error if the **Error** output is set to TRUE. Error identifier:

ErrorID	Description
1	TimeLimit exceeded.
2	DistanceLimit exceeded.
3	TorqueLimit exceeded.
4	Axis error stop state.
5	Axis not enabled.
6	Invalid inputs for Velocity-Acceleration-Deceleration.

3.8.4.3 Remarks

- Performs a homing function by searching for an external physical switch.
- The switch must be connected to one of the two fast inputs on the Axis' AKD drive.
- The Axis moves and, when a fast input is triggered, the triggered axis then performs an absolute move to the latched position.

This image shows the function or function block I/O.

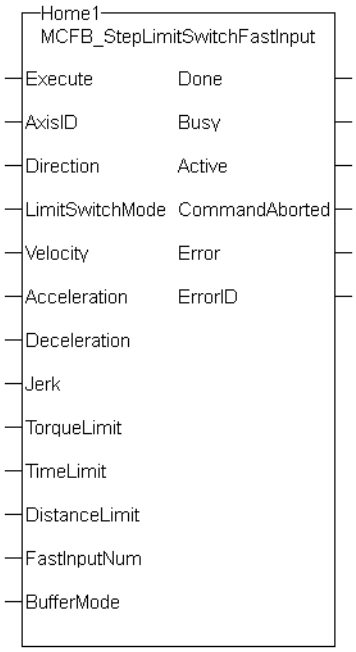
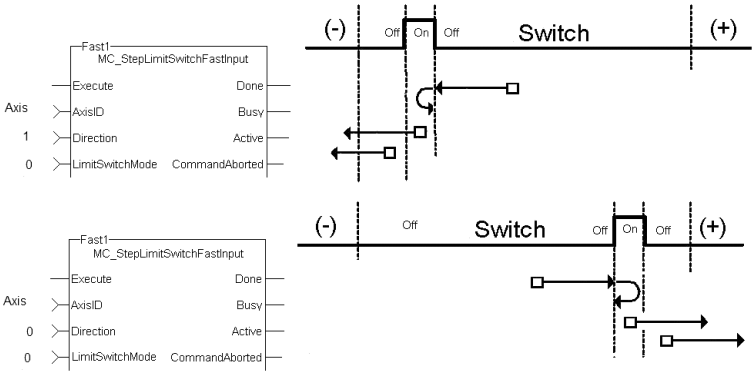


Figure 3-16: MCFB_StepLimitSwitchFastInput

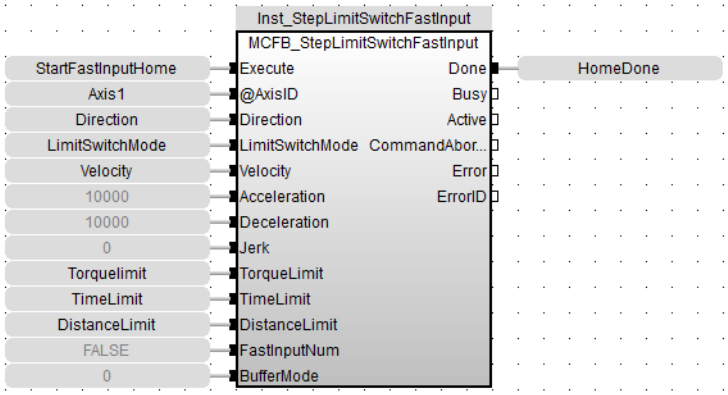
3.8.4.3.1 Usage

The homing is commanded in the most likely direction where the sensor can be found.

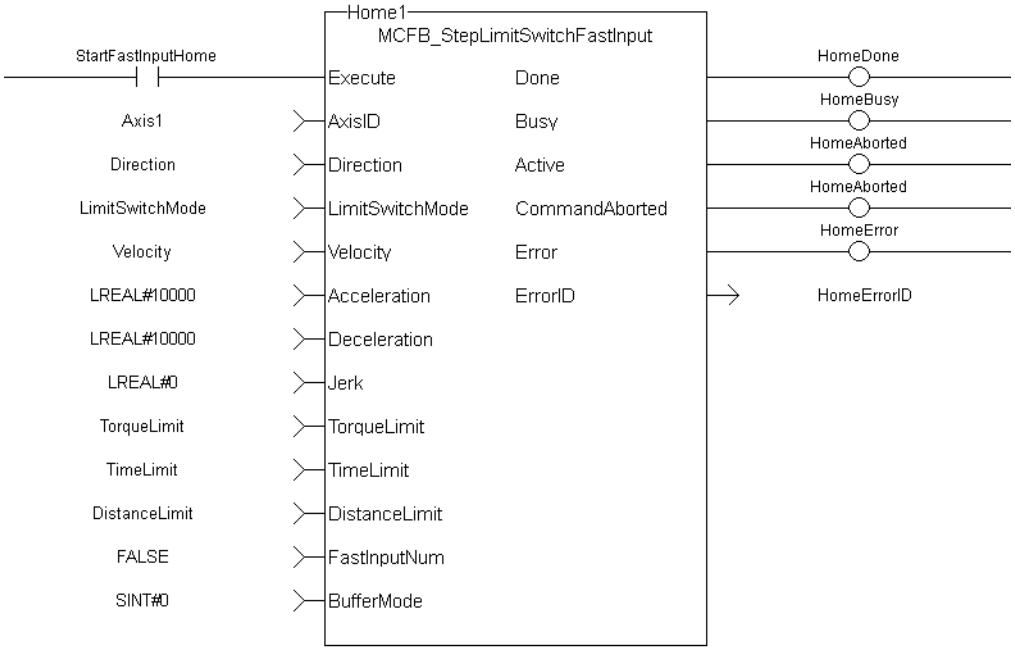
In this example (-).



3.8.4.4 FBD Language Example



3.8.4.5 FFLD Language Example



3.8.4.6 IL Language Example

Not available.

3.8.4.7 ST Language Example

```
Execute_1 :=1;

Inst_MCFB_StepLimitSwitchFastInput( Execute_1, Axis1, 0, 0, 10000.0, 10000.0,
10000.0, 0, 0, 0, 0, 0, 0);

HomeComplete := Inst_MCFB_StepLimitSwitchFastInput.Done;
HomeBusy := Inst_MCFB_StepLimitSwitchFastInput.Busy;
HomeActive := Inst_MCFB_StepLimitSwitchFastInput.Active;
HomeAborted := Inst_MCFB_StepLimitSwitchFastInput.CommandAborted;
HomeError := Inst_MCFB_StepLimitSwitchFastInput.Error;
HomeErrorID := Inst_MCFB_StepLimitSwitchFastInput.ErrorID;
```

See Also

"MCFB_StepAbsSwitchFastInput" (→ p. 141)

3.8.5 Homing - UDFB - Motion/PLCopen

These are the UDFB Motion/PLCopen Homing function blocks.

Name	Description
"MCFB_StepAbsolutes" (→ p. 151)	Homing by setting Actual Position to the position of the feedback.
"MCFB_StepAbsSwitch" (→ p. 153)	Homing to a home switch.
"MCFB_StepBlock" (→ p. 158)	Homing to a physical object, mechanically blocking the movement.
"MCFB_StepLimitSwitch" (→ p. 162)	Homing to a limit switch.
"MCFB_StepRefPulse" (→ p. 166)	Homing to a zero angle reference.

3.8.5.1 MCFB_StepAbsolutes

Function Block - Homing by setting Actual Position to the position of the feedback.

3.8.5.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	No range	N/A	No default	Enables execution. Used in the FFLD editor only.
Execute	BOOL	0, 1	N/A	No default	Request the homing step procedure at the rising edge.
AxisID	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function. See AXIS_REF Structure for more information.
Mode	BOOL	0, 1	N/A	No default	Define the actual position assignment source. • 0 (zero) = Use drive feedback position for actual position. • 1 = Use once per rev feedback position.

3.8.5.1.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL		N/A	Indicates the move completed successfully. The Command Position has reached the endpoint.
Busy	BOOL		N/A	High from the moment the Execute input goes high until the time the move is ended.
Active	BOOL		N/A	Indicates this move is the Active move.
CommandAborted	BOOL		N/A	Indicates the move was aborted.
Error	BOOL		N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.

Output	Data Type	Range	Unit	Description
ErrorID	INT		N/A	Indicates the error if the **Error** output is set to TRUE. 1 = Desired SetPosition is outside of Rollover period.

3.8.5.1.3 Remarks

- Performs a static homing function by setting Actual Position to the position of an absolute encoder.
- No physical motion is performed in this mode.
- Equivalent to MC_SetPosition is performed with SetPosition coming from absolute encoder reading but with the option of using the once per rev feedback value.

This image shows the function or function block I/O.

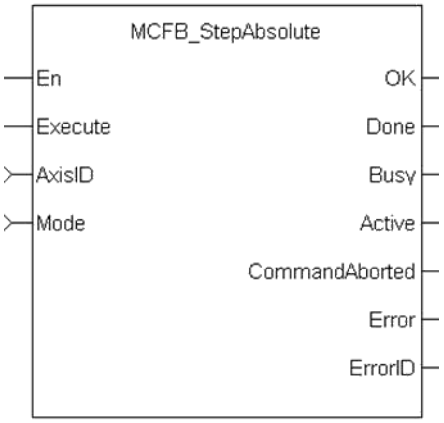
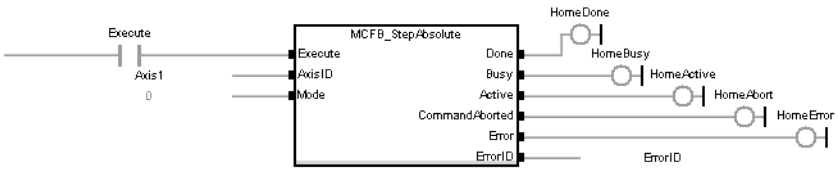
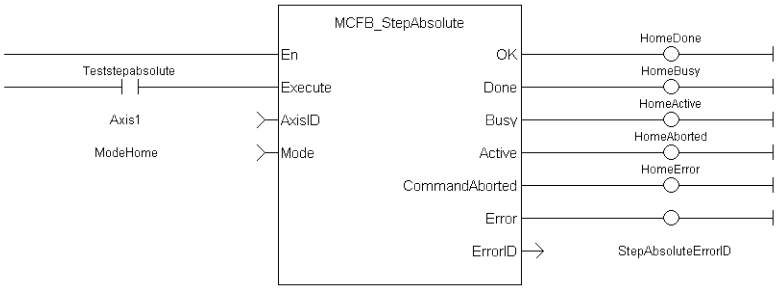


Figure 3-17: MCFB_StepAbsolute

3.8.5.1.4 FBD Language Example



3.8.5.1.5 FFLD Language Example



3.8.5.1.6 IL Language Example

Not available.

3.8.5.1.7 ST Language Example

```
//Write current once per rev feedback position to overall axis position
MCFB_StepAbsolute( ExecuteHome, Axis1, ModeHome );
```

See Also

- "MCFB_StepAbsSwitch" (→ p. 153)
- "MCFB_StepBlock" (→ p. 158)
- "MCFB_StepLimitSwitch" (→ p. 162)
- "MCFB_StepRefPulse" (→ p. 166)

3.8.5.2 MCFB_StepAbsSwitch

PLCopen

Function Block - Homing to a home switch.

3.8.5.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	Request the homing step procedure at the rising edge. Outputs are reset when execute input is FALSE.
AxisID	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function. See AXIS_REF Structure for more information.
Direction	DINT	0, 3	N/A	No default	Defines the axis homing direction. • 0 (zero) = Clockwise rotation. • 1 = Counterclockwise rotation. • 2 = Clockwise if AbsoluteSwitch starts Off and negative if switch starts On. • 3 = Counter-clockwise if AbsoluteSwitch starts On and positive if switch starts Off.
SwitchMode	DINT	0, 3	N/A	No default	Switch state to complete homing. • 0 (zero) = Switch is on. • 1 = Switch is off. • 2 = Rising edge of switch. • 3 = Falling edge of switch.
Velocity	LREAL	No range	User unit/sec	No default	Commanded velocity for the homing move.
Acceleration	LREAL	No range	User unit/sec^2	No default	Commanded acceleration for the homing move.
Deceleration	LREAL	No range	User unit/sec^2	No default	Commanded deceleration for the homing move.

Input	Data Type	Range	Unit	Default	Description
Jerk	LREAL	No range	User unit/sec^3	No default	Commanded jerk for the homing move. If 0 (zero), trapezoidal acc/dec is used.
TorqueLimit	LREAL	No range	User units	No default	Maximum torque applied for the homing move. • Entered in thousandths of maximum torque. • Example: 250 is 250/1000, or 25%.
TimeLimit	TIME	No range	Sec	No default	Maximum time for homing move to complete. • If exceeded, the homing procedure errors out. • 0 (zero) = no time limit.
DistanceLimit	LREAL	No range	User units	No default	Maximum distance for homing move to complete. • If exceeded, the homing procedure errors out. • 0 (zero) = no distance limit.
AbsoluteSwitch	BOOL	0, 1	N/A	No default	The absolute switch input I/O point.
PosLimitSwitch	BOOL	0, 1	N/A	No default	The positive direction limit switch input I/O point.
NegLimitSwitch	BOOL	0, 1	N/A	No default	The negative direction limit switch input I/O point.
SwitchMode	SINT	0, 5	N/A	No default	Switch state to complete homing. • 0 = Abort. • 1 = Buffer. • 2 = Blend to active. • 3 = Blend to next. • 4 = Blend to low velocity. • 5 = Blend to high velocity.

3.8.5.2.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL		N/A	Indicates the move completed successfully. The Command Position has reached the endpoint.
Busy	BOOL		N/A	High from the moment the Execute input goes high until the time the move is ended.
Active	BOOL		N/A	Indicates this move is the Active move.

Output	Data Type	Range	Unit	Description
CommandAborted	BOOL		N/A	Indicates the move was aborted.
Error	BOOL		N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
ErrorID	INT		N/A	Indicates the error if the **Error** output is set to TRUE. Error identifier:

ErrorID	Description
1	TimeLimit exceeded.
2	DistanceLimit exceeded.
3	TorqueLimit exceeded.
4	Axis error stop state.
5	Axis not enabled.
6	Invalid inputs for Velocity-Acceleration-Deceleration.

3.8.5.2.3 Remarks

- Performs a homing function by searching for an absolute positioned external physical switch.
- An Absolute Switch has two Off (or On) areas.

This image shows the function or function block I/O.

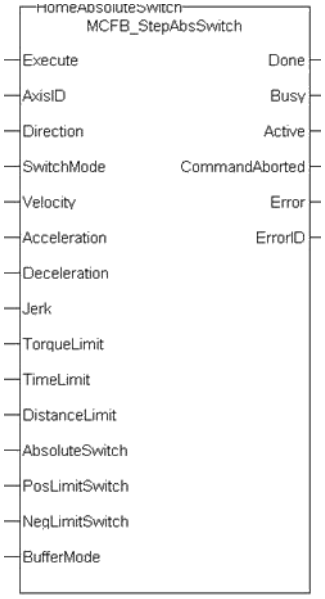


Figure 3-18: MCFB_StepAbsSwitch

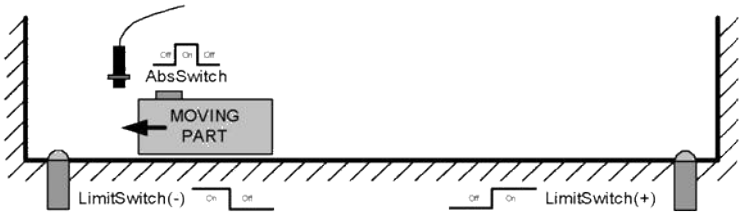
3.8.5.2.3.1 Usage

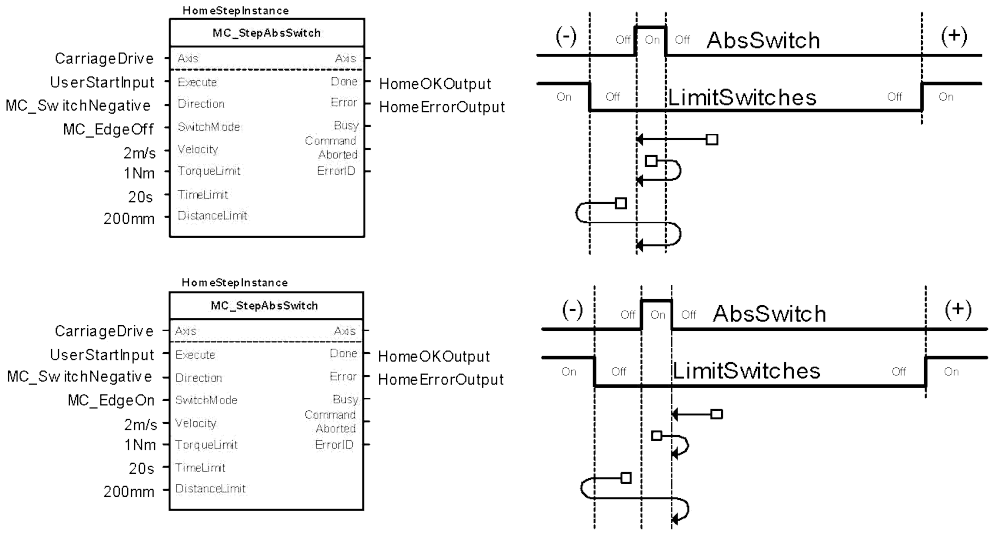
This physical layout has the risk that homing is started in the wrong direction (escaping the switch).

To support such case, it implements a special behavior when Limit Switches are found (or the AbsSwitch itself is On at Execute).

- This function block does not modify the actual position.
- The homing is commanded in the most likely direction where the sensor can be found.

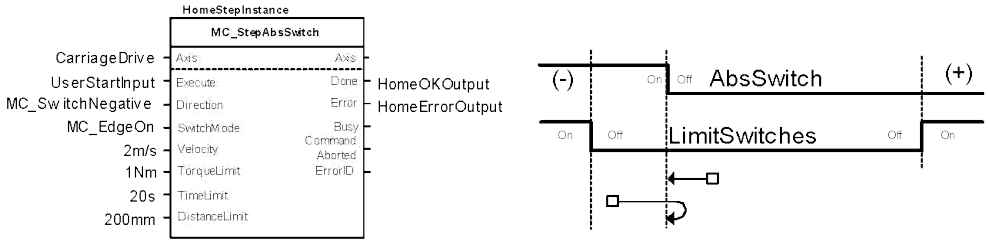
- The Velocity is defined by the input.
- The Torque is limited.
- Time and Distance Limits can cause an error if exceeded.
- If any LimitSwitch is found during Homing (any of them), a special process is started in the opposite direction.
  - The AbsSwitch is searched to switch Off (or On, depending on SwitchMode setting).
- The Edge (passed by) and homing process is restarted in the original direction and with the same conditions.
  - This ensures the end conditions are always the same.
- If the SwitchMode is either MC_SwitchNegative or MC_SwitchPositive, then the special process is also started in opposite direction depending from the switch state at Execute.
- The direction changes only when the specified Velocity is reached.





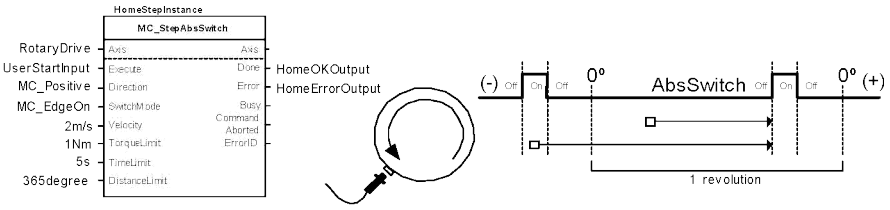
An overlapping switch configuration is also possible.

This has same the behavior as working on the limit switches:

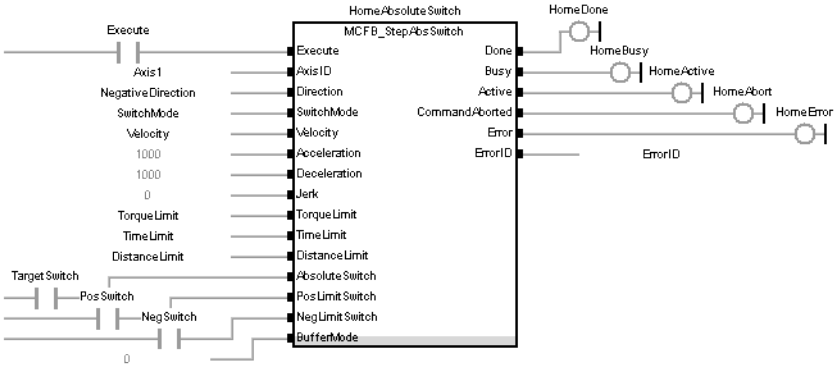


If the input Direction is set to a fixed direction (MC_Positive or MC_Negative), then the initial switch state is ignored.

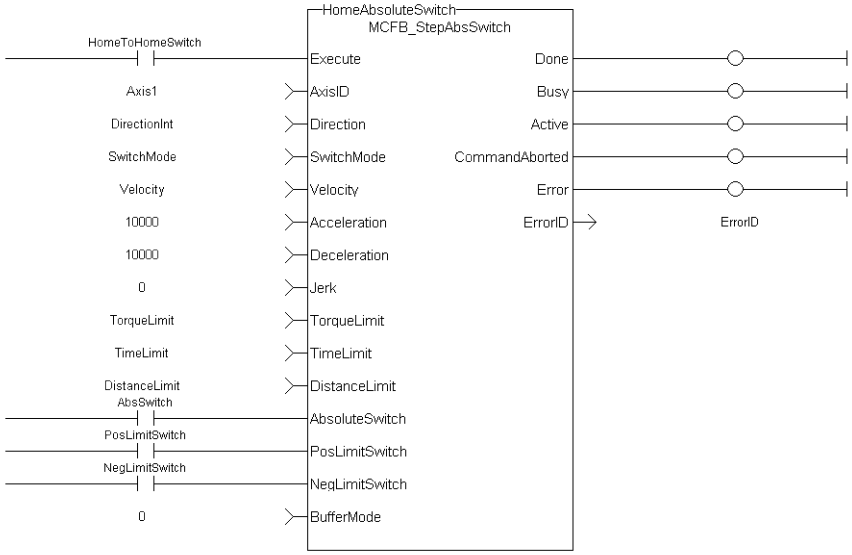
Used for example in rotary axis where only one sense of rotation is allowed:



3.8.5.2.4 FBD Language Example



3.8.5.2.5 FFLD Language Example



3.8.5.2.6 IL Language Example

Not available.

3.8.5.2.7 ST Language Example

```
NegativeDirection :=1;
RisingEdge :=2;
Velocity :=10000.0;
TorqueLimit :=50.0;
TimeLimit :=T#10s;
DistanceLimit :=10000.0;

Inst_MCFB_StepAbsSwitch( True, Axis1, NegativeDirection, RisingEdge,
Velocity, 1000, 1000, 0, TorqueLimit, TimeLimit, DistanceLimit,
```

```
AbsoluteSwitch, PosLimitSwitch, NegLimitSwitch, 0 );

HomeComplete :=Inst_MCFB_StepAbsSwitch.Done;
HomeBusy :=Inst_MCFB_StepAbsSwitch.Busy;
HomeActive :=Inst_MCFB_StepAbsSwitch.Active;
HomeAborted :=Inst_MCFB_StepAbsSwitch.CommandAborted;
HomeError :=Inst_MCFB_StepAbsSwitch.Error;
HomeErrorID :=Inst_MCFB_StepAbsSwitch.ErrorID;
(* AbsoluteSwitch, PosLimitSwitch, NegLimitSwitch are declared I/O points *)
```

See Also

- "MCFB_StepAbsolutes" (→ p. 151)
- "MCFB_StepBlock" (→ p. 158)
- "MCFB_StepLimitSwitch" (→ p. 162)
- "MCFB_StepRefPulse" (→ p. 166)

3.8.5.3 MCFB_StepBlock

PLCopen ✓

Function Block - Homing to a physical object, mechanically blocking the movement.

3.8.5.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	Request the homing step procedure at the rising edge. Outputs are reset when execute input is FALSE.
AxisID	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function. See AXIS_REF Structure for more information.
Direction	DINT	0, 1	N/A	No default	Defines the axis homing direction. • 0 (zero) = Clockwise rotation. • 1 = Counterclockwise rotation.
Velocity	LREAL	No range	User unit/sec	No default	Commanded velocity for the homing move.
Acceleration	LREAL	No range	User unit/sec^2	No default	Commanded acceleration for the homing move.
Deceleration	LREAL	No range	User unit/sec^2	No default	Commanded deceleration for the homing move.
Jerk	LREAL	No range	User unit/sec^3	No default	Commanded jerk for the homing move. If 0 (zero), trapezoidal acc/dec is used.

Input	Data Type	Range	Unit	Default	Description
TorqueLimit	LREAL	No range	User units	No default	Maximum torque applied for the homing move. • Entered in thousandths of maximum torque. • Example: 250 is 250/1000, or 25%.
TimeLimit	TIME	No range	Sec	No default	Maximum time for homing move to complete. • If exceeded, the homing procedure errors out. • 0 (zero) = no time limit.
DistanceLimit	LREAL	No range	User units	No default	Maximum distance for homing move to complete. • If exceeded, the homing procedure errors out. • 0 (zero) = no distance limit.
BufferMode	SINT	0, 5	N/A	No default	Switch state to complete homing. • 0 = Abort. • 1 = Buffer. • 2 = Blend to active. • 3 = Blend to next. • 4 = Blend to low velocity. • 5 = Blend to high velocity.

3.8.5.3.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL		N/A	Indicates the move completed successfully. The Command Position has reached the endpoint.
Busy	BOOL		N/A	High from the moment the Execute input goes high until the time the move is ended.
Active	BOOL		N/A	Indicates this move is the Active move.
CommandAborted	BOOL		N/A	Indicates the move was aborted.
Error	BOOL		N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
ErrorID	INT		N/A	Indicates the error if the **Error** output is set to TRUE. Error identifier:

ErrorID	Description
1	TimeLimit exceeded.
2	DistanceLimit exceeded.
3	TorqueLimit exceeded.
4	Axis error stop state.
5	Axis not enabled.
6	Invalid inputs for Velocity-Acceleration-Deceleration.

3.8.5.3.3 Remarks

- Performs homing against a physical object, mechanically blocking the movement.
- In this mode, there is no limit switch or Reference Pulse.
- Adequate torque limits are required for not damaging mechanics during homing process.
- The StepBlock condition is that we have reached the torque limit and real velocity falls below 5% of demanded.

This image shows the function or function block I/O.

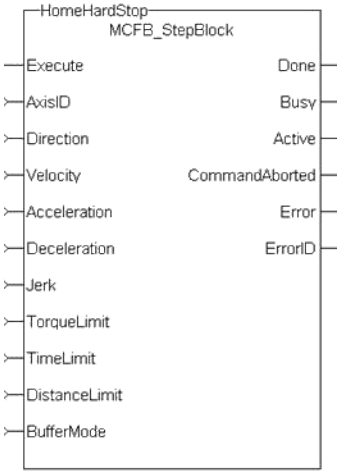
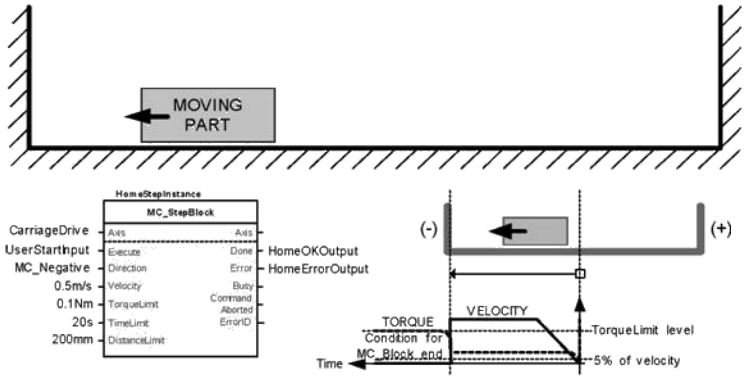


Figure 3-19: MCFB_StepBlock

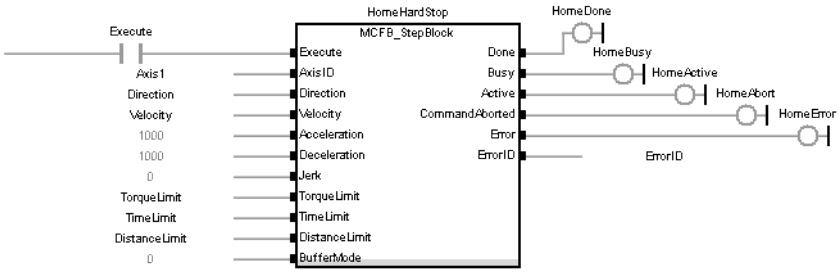
3.8.5.3.3.1 Usage

Homing against a physical object, mechanically blocking the movement require adequate torque limits for not damaging mechanics during homing process.

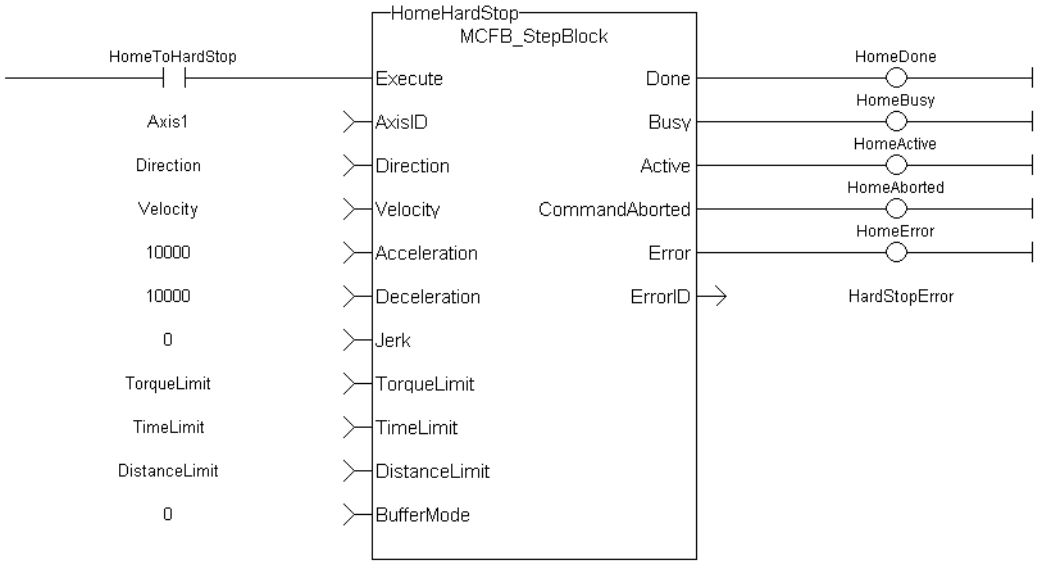
- This function block does not modify the actual position.
- Home is commanded by the user in the designated homing direction at the selected or programmed Velocity.
- The Torque is limited.
- Time and Distance Limits can cause an error if exceeded.
- Process is finished when Torque is in limit condition and real velocity is below 5% of selected Velocity.



3.8.5.3.4 FBD Language Example



3.8.5.3.5 FFLD Language Example



3.8.5.3.6 IL Language Example

Not available.

3.8.5.3.7 ST Language Example

```
PositiveDirection :=0;
Velocity :=10000.0;
TorqueLimit :=50.0;
TimeLimit :=T#10s;
DistanceLimit :=10000.0;

Inst_MCFB_StepBlock( True, Axis1, PositiveDirection, Velocity, 1000, 1000, 0,
TorqueLimit, TimeLimit, DistanceLimit, 0 );

HomeComplete :=Inst_MCFB_StepBlock.Done;
HomeBusy :=Inst_MCFB_StepBlock.Busy;
HomeActive :=Inst_MCFB_StepBlock.Active;
HomeAborted :=Inst_MCFB_StepBlock.CommandAborted;
HomeError :=Inst_MCFB_StepBlock.Error;
HomeErrorID :=Inst_MCFB_StepBlock.ErrorID;
```

See Also

- "MCFB_StepAbsolutes" (→ p. 151)
- "MCFB_StepAbsSwitch" (→ p. 153)
- "MCFB_StepLimitSwitch" (→ p. 162)
- "MCFB_StepRefPulse" (→ p. 166)

3.8.5.4 MCFB_StepLimitSwitch

Function Block - Homing to a limit switch.

3.8.5.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	Request the homing step procedure at the rising edge. Outputs are reset when execute input is FALSE.
AxisID	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function. See AXIS_REF Structure for more information.
Direction	BOOL	0, 1	N/A	No default	Defines the axis homing direction. • 0 (zero) = Clockwise rotation. • 1 = Counterclockwise rotation.
LimitSwitchMode	DINT	0, 3	N/A	No default	Limit switch state to complete homing. • 0 (zero) = Switch is on. • 1 = Switch is off. • 2 = Rising edge of switch. • 3 = Falling edge of switch.
Velocity	LREAL	No range	User unit/sec	No default	Commanded velocity for the homing move.
Acceleration	LREAL	No range	User unit/sec^2	No default	Commanded acceleration for the homing move.
Deceleration	LREAL	No range	User unit/sec^2	No default	Commanded deceleration for the homing move.
Jerk	LREAL	No range	User unit/sec^3	No default	Commanded jerk for the homing move. If 0 (zero), trapezoidal acc/dec is used.
TorqueLimit	LREAL	No range	User units	No default	Maximum torque applied for the homing move. • Entered in thousandths of maximum torque. • Example: 250 is 250/1000, or 25%.

Input	Data Type	Range	Unit	Default	Description
TimeLimit	TIME	No range	Sec	No default	Maximum time for homing move to complete. • If exceeded, the homing procedure errors out. • 0 (zero) = no time limit.
DistanceLimit	LREAL	No range	User units	No default	Maximum distance for homing move to complete. • If exceeded, the homing procedure errors out. • 0 (zero) = no distance limit.
LimitSwitch	BOOL	0, 1	N/A	No default	The limit switch input I/O point.
BufferMode	SINT	0, 5	N/A	No default	Define the homing move start action. • 0 = Abort. • 1 = Buffer. • 2 = Blend to active. • 3 = Blend to next. • 4 = Blend to low velocity. • 5 = Blend to high velocity.

3.8.5.4.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL		N/A	Indicates the move completed successfully. The Command Position has reached the endpoint.
Busy	BOOL		N/A	High from the moment the Execute input goes high until the time the move is ended.
Active	BOOL		N/A	Indicates this move is the Active move.
CommandAborted	BOOL		N/A	Indicates the move was aborted.
Error	BOOL		N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.

Output	Data Type	Range	Unit	Description
ErrorID	INT		N/A	Indicates the error if the **Error** output is set to TRUE. Error identifier:

ErrorID	Description
1	TimeLimit exceeded.
2	DistanceLimit exceeded.
3	TorqueLimit exceeded.
4	Axis error stop state.
5	Axis not enabled.
6	Invalid inputs for Velocity-Acceleration-Deceleration.

3.8.5.4.3 Remarks

- Performs a single-axis home to a limit switch.
- The limit switches are used for homing procedure.
  - They are always active once moving part working area has been surpassed.

This image shows the function or function block I/O.

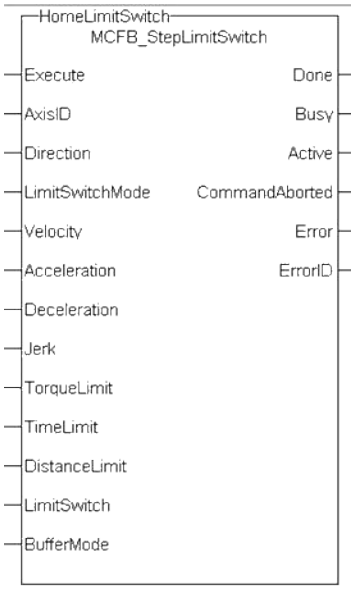


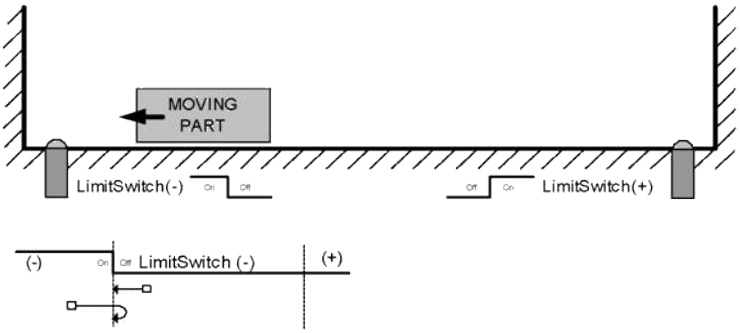
Figure 3-20: MCFB_StepLimitSwitch

3.8.5.4.3.1 Usage

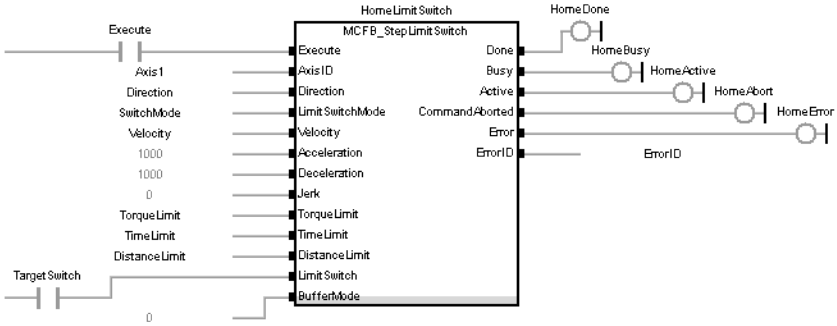
- This function block does not modify the actual position.
- Home is commanded by the user in the designated homing direction at the selected or programmed Velocity.
- This procedure performs a homing function searching for sensor using only LimitSwitches.
  - A LimitSwitch has one Off (or On) area.
- If **LimitSwitch** is found On with a rising Execute, the process is started in the opposite direction as specified.

- LimitSwitch is searched for Off Edge (released) and process is restarted again in original direction.
  - Or On, depending on LimitSwitchMode setting.
  - This ensures the end conditions are always the same.

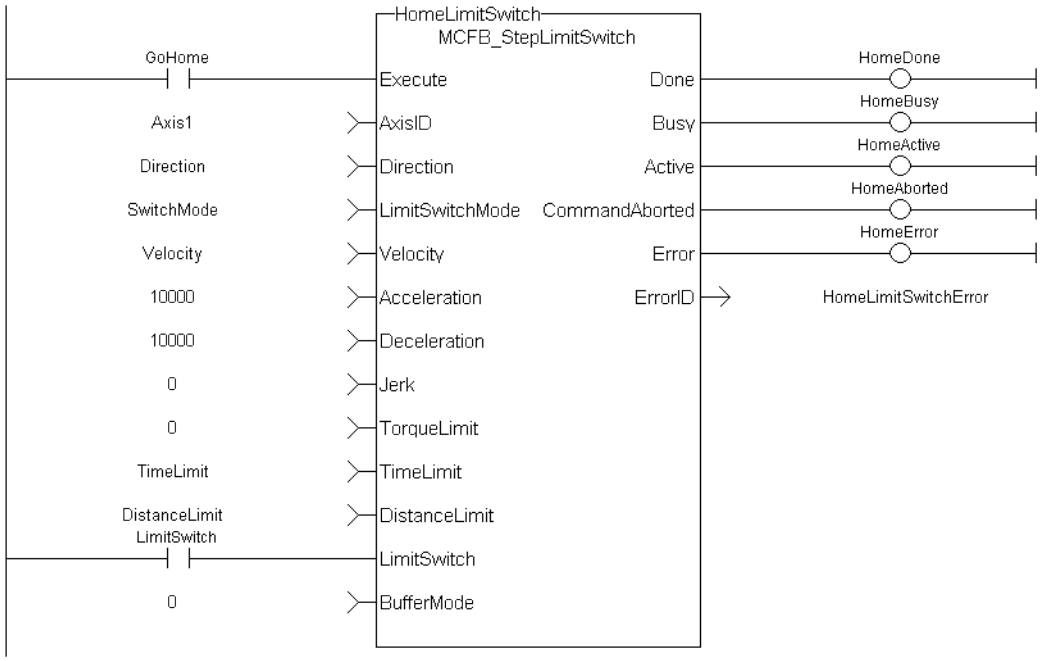
- The Torque is limited.
- Time and Distance Limits can cause an error if exceeded.
- The direction changes only when the specified Velocity is reached.
  - This ensures acceleration and deceleration spaces are fixed.



3.8.5.4.4 FBD Language Example



3.8.5.4.5 FFLD Language Example



3.8.5.4.6 IL Language Example

Not available.

3.8.5.4.7 ST Language Example

```
PositiveDirection :=0;
RisingEdge :=2;
Velocity :=10000.0;
TorqueLimit :=50.0;
TimeLimit :=T#10s;
DistanceLimit :=10000.0;

Inst_MCFB_StepLimitSwitch( True, Axis1, PositiveDirection, RisingEdge,
Velocity, 1000, 1000, 0, TorqueLimit, TimeLimit, DistanceLimit, LimitSwitch,
0 );

HomeComplete :=Inst_MCFB_StepLimitSwitch.Done;
HomeBusy :=Inst_MCFB_StepLimitSwitch.Busy;
HomeActive :=Inst_MCFB_StepLimitSwitch.Active;
HomeAborted :=Inst_MCFB_StepLimitSwitch.CommandAborted;
HomeError :=Inst_MCFB_StepLimitSwitch.Error;
HomeErrorID :=Inst_MCFB_StepLimitSwitch.ErrorID;

(* LimitSwitch is a declared I/O point *)
```

See Also

- "MCFB_StepAbsolutes" (→ p. 151)
- "MCFB_StepAbsSwitch" (→ p. 153)
- "MCFB_StepBlock" (→ p. 158)
- "MCFB_StepRefPulse" (→ p. 166)

3.8.5.5 MCFB_StepRefPulse

PLCopen

Function Block - Homing to a zero angle reference.

3.8.5.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	Request the homing step procedure at the rising edge.
AxisID	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function. See AXIS_REF Structure for more information.
Direction	BOOL	0, 1	N/A	No default	Defines the axis homing direction. • 0 (zero) = Clockwise rotation. • 1 = Counterclockwise rotation.

Input	Data Type	Range	Unit	Default	Description
SwitchMode	DINT	0, 3	N/A	No default	Switch state to complete homing. • 0 (zero) = Switch is on. • 1 = Switch is off. • 2 = Rising edge of switch. • 3 = Falling edge of switch.
Velocity	LREAL	No range	User unit/sec	No default	Commanded velocity for the homing move.
Acceleration	LREAL	No range	User unit/sec^2	No default	Commanded acceleration for the homing move.
Deceleration	LREAL	No range	User unit/sec^2	No default	Commanded deceleration for the homing move.
Jerk	LREAL	No range	User unit/sec^3	No default	Commanded jerk for the homing move. If 0 (zero), trapezoidal acc/dec is used.
SetPosition	LREAL	No range	User units	No default	Value of the absolute position to be set when the homing move is done.
TorqueLimit	LREAL	No range	User units	No default	Maximum torque applied for the homing move. • Entered in thousandths of maximum torque. • Example: 250 is 250/1000, or 25%.
TimeLimit	TIME	No range	Sec	No default	Maximum time for homing move to complete. • If exceeded, the homing procedure errors out. • 0 (zero) = no time limit.
DistanceLimit	LREAL	No range	User units	No default	Maximum distance for homing move to complete. • If exceeded, the homing procedure errors out. • 0 (zero) = no distance limit.
BufferMode	SINT	0, 5	N/A	No default	Define the homing move start action. • 0 = Abort. • 1 = Buffer. • 2 = Blend to active. • 3 = Blend to next. • 4 = Blend to low velocity. • 5 = Blend to high velocity.

3.8.5.5.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL		N/A	Indicates the move completed successfully. The Command Position has reached the endpoint.

Output	Data Type	Range	Unit	Description
Busy	BOOL		N/A	High from the moment the Execute input goes high until the time the move is ended.
Active	BOOL		N/A	Indicates this move is the Active move.
CommandAborted	BOOL		N/A	Indicates the move was aborted.
Error	BOOL		N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
ErrorID	INT		N/A	Indicates the error if the **Error** output is set to TRUE. Error identifier:

ErrorID	Description
1	TimeLimit exceeded.
2	DistanceLimit exceeded.
3	TorqueLimit exceeded.
4	Axis error stop state.
5	Axis not enabled.
6	Invalid inputs for Velocity-Acceleration-Deceleration.

3.8.5.5.3 Remarks

- This function modifies actual position and sets to the SetPosition input value at the end.
- Performs homing by searching for Zero pulse in encoder.
  - Also called Marker or Reference Pulse.

- The pulse appears once per encoder revolution.
- The advantage in using Reference Pulse for homing is the higher accuracy and precision achieved compared to traditional optical, mechanical, or magnetic sensors.

This image shows the function or function block I/O.

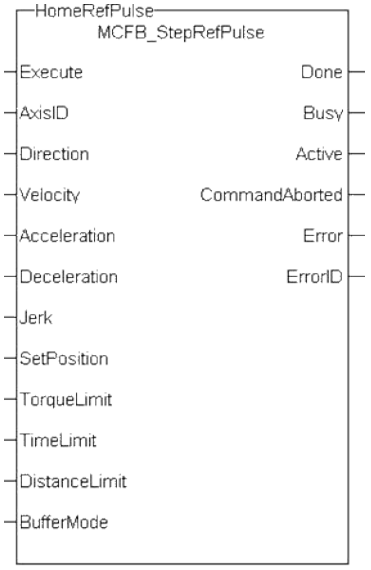
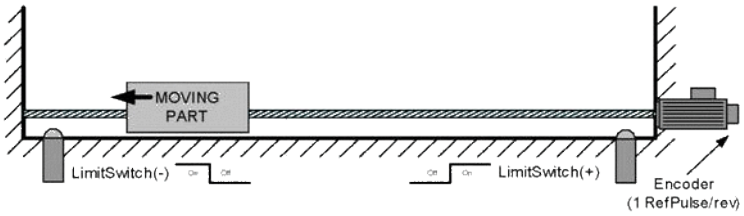
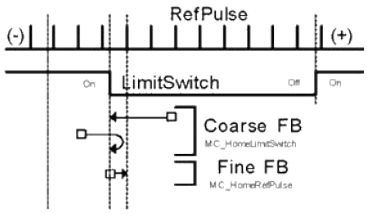


Figure 3-21: MCFB_StepRefPulse

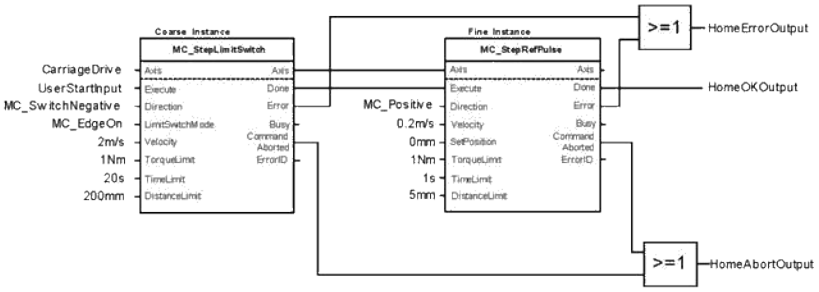
3.8.5.5.3.1 Usage

- Home is commanded by the user in the designated homing direction at the selected or programmed Velocity.
- First occurrence of the Reference Pulse, Homing is finished.
- The Torque is limited.
- Time and Distance Limits can cause an error if exceeded.

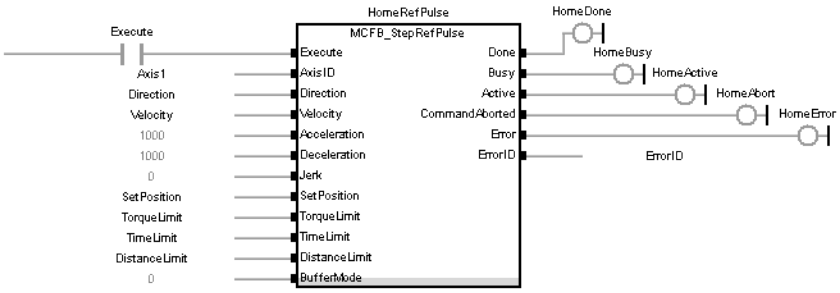




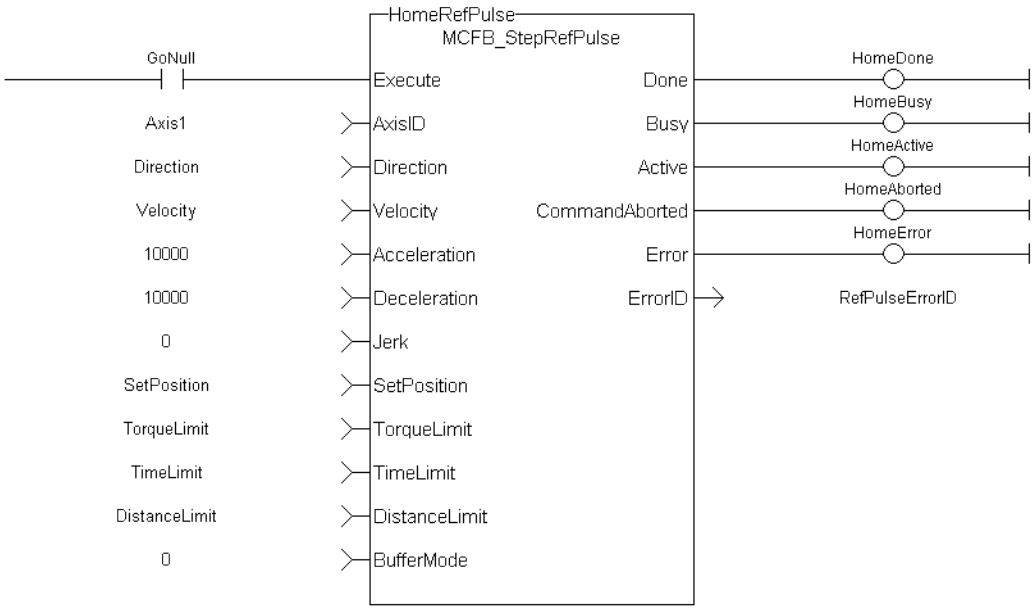
- It is common that a first approach is performed against a mechanical sensor at higher velocity, and after a Reference Pulse, at a lower velocity.
- This is a traditional 2-Step homing (Coarse by external Switch in reverse and Fine by Reference Pulse in forward).



3.8.5.5.4 FBD Language Example



3.8.5.5.5 FFLD Language Example



3.8.5.5.6 IL Language Example

Not available.

3.8.5.5.6.1 Structured Text

```
PositiveDirection :=0;
Velocity :=10000.0;
SetPosition :=0.0;
TorqueLimit :=50.0;
TimeLimit :=T#10s;
DistanceLimit :=10000.0;

Inst_MCFB_StepRefPulse( True, Axis1, PositiveDirection, Velocity, 1000, 1000,
0, SetPosition, TorqueLimit, TimeLimit, DistanceLimit, 0 );

HomeComplete :=Inst_MCFB_StepRefPulse.Done;
HomeBusy :=Inst_MCFB_StepRefPulse.Busy;
HomeActive :=Inst_MCFB_StepRefPulse.Active;
HomeAborted :=Inst_MCFB_StepRefPulse.CommandAborted;
HomeError :=Inst_MCFB_StepRefPulse.Error;
HomeErrorID :=Inst_MCFB_StepRefPulse.ErrorID;
```

See Also

- "MCFB_StepAbsolutes" (→ p. 151)
- "MCFB_StepAbsSwitch" (→ p. 153)
- "MCFB_StepBlock" (→ p. 158)
- "MCFB_StepLimitSwitch" (→ p. 162)

3.8.6 MCFB_Jog

PLCopen

Function - Define to jog an axis in the selected direction at a defined speed.

3.8.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	No range	N/A	No default	Enables execution. Used in the FFLD editor only.
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block. The **AxisID** selects the axis to jog.
JogPlus	BOOL	0, 1	N/A	No default	Enables a jog in the plus direction.
JogMinus	BOOL	0, 1	N/A	No default	Enables a jog in the minus direction.
Speed	LREAL	No range	User unit/sec	No default	Rate the axis moves.
Accel **Decel**	LREAL	No range	User unit/sec^2	No default	Linear Acc/Dec rate.

3.8.6.2 Outputs

Output	Data Type	Range	Unit	Description
InMotion	BOOL		N/A	Jogging is active when TRUE.

3.8.6.3 Remarks

- This function or function block can be used with PLCopen Motion engines.
- This function or function block is used to command motion in a designated direction at a defined rate.
  - This may be used where continuous motion is required (e.g., a conveyor system or in a setup mode for manually jogging the axis).
- The **JogPlus** and **JogMinus** inputs select the direction the motion occurs in.
  - Motion starts when the **JogPlus** or **JogMinus** input is TRUE.
    Motion stops when the input is FALSE.
  - Only one of these inputs should be enabled at a time.
  - If both are selected, the motion stops.
- If another motion is active when the jog is requested, that motion is aborted and the jog starts.
- The En input (FFLD editor only) must be high.
  - Typically wired to the rail.

This image shows the function or function block I/O.

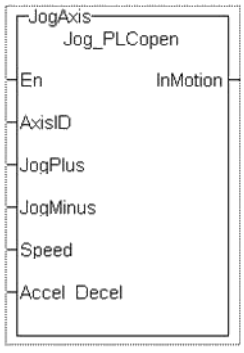
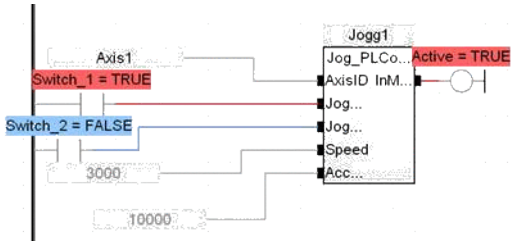
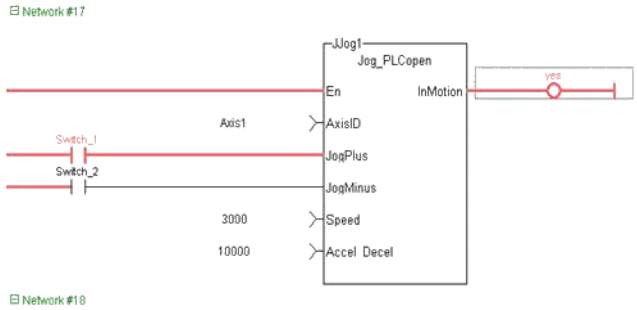


Figure 3-22: Jog for PLCopen

3.8.6.4 FBD Language Example



3.8.6.5 FFLD Language Example



3.8.6.6 IL Language Example

Not available.

3.8.6.7 ST Language Example

```
InMotion := Inst_Jog_PLCopen(Axis1, Switch_1, Switch_2, 600, 10000);
```

See Also

"MLAxisMoveVel" (→ p. 250)

3.8.7 MCFB_GearedWebTension

PLCopen

Function Block - Facilitates dancer and tension control in an electronic geared master/slave machine design.

3.8.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	Enables execution.
MasterID	AXIS_REF	N/A	N/A	No default	Master axis identifier.
SlaveID	AXIS_REF	N/A	N/A	No default	Slave axis identifier.
RatioNumerator	DINT	-2147483648 to 2147483647	N/A	No default	Numerator of master/slave ratio.

Input	Data Type	Range	Unit	Default	Description
RatioDenominator	DINT	-2147483648 to 2147483647	N/A	No default	Denominator of master/slave ratio.
Acceleration	LREAL	-1.7E308 to 1.7E308	N/A	No default	Trapezoidal: Acceleration rate. • S-curve: Maximum acceleration. • 14 to 15 significant digits of accuracy.
Deceleration	LREAL	-1.7E308 to 1.7E308	N/A	No default	Trapezoidal: Deceleration rate. • S-curve: Unused. • 14 to 15 significant digits of accuracy.
Jerk	LREAL	-1.7E308 to 1.7E308	N/A	No default	Trapezoidal: 0 (zero). • S-curve: Constant jerk. • 14 to 15 significant digits of accuracy.
Kp	LREAL	-1.7E308 to 1.7E308	N/A	No default	Proportional gain. 14 to 15 significant digits of accuracy.
Ti	LREAL	-1.7E308 to 1.7E308	N/A	No default	Integral gain. 14 to 15 significant digits of accuracy.
Td	LREAL	-1.7E308 to 1.7E308	N/A	No default	Derivative gain. 14 to 15 significant digits of accuracy.
DeviceFBValue	DINT	-2147483648 to 2147483647	N/A	No default	Analog input.
DeviceSetPoint	DINT	-2147483648 to 2147483647	N/A	No default	Analog set point.
ErrorDB	LREAL	-1.7E308 to 1.7E308	N/A	No default	Maximum or minimum error between DeviceFBValue and DeviceSetPoint before a change takes place. 14 to 15 significant digits of accuracy.
RatioLimitPercent	LREAL	-1.7E308 to 1.7E308	N/A	No default	Maximum and minimum master/slave ratio window. 14 to 15 significant digits of accuracy.

ErrorCalcMode	BOOL	0, 1	N/A	No default	• Not set:
					  • DeviceFBValue-DeviceSetPoint
					• Set:
					  • DeviceSetPoint-DeviceFBValue

3.8.7.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	0, 1	N/A	The output has power flow after the enable input has been energized.
NewRatio	REAL	-3.4E38 to 3.4E38	N/A	New master/slave ratio. 6 to 7 significant digits of accuracy.
Error	BOOL	0, 1	N/A	Function block error.
ErrorID	INT	-32768 to +32767	N/A	Function block error value.

3.8.7.3 Remarks

- This facilitation is done by using the analog feedback from an LVDT, tension transducer, potentiometer, encoder, resolver or some other similar device.
  - The analog feedback value is compared to a pre-determined analog set-point.
- The difference or error is used in a PID algorithm with the summed output driving changes to the master/slave gearing relationship.
  - This results in the slave axis either speeding up or slowing down to maintain desired tension.

This image shows the function or function block I/O.

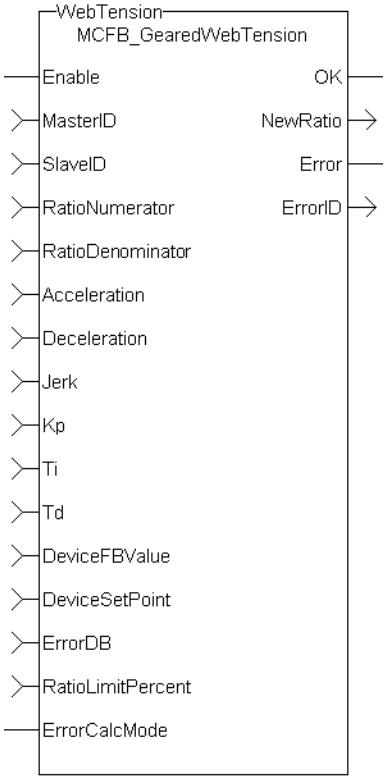


Figure 3-23: MCFB_GearedWebTension

3.8.7.3.1 Usage

- This UDFB is used in conjunction with the main ladder "MC_GearIn" (➜ p. 492) function.
  - It is assumed that the master/slave move is active.

- Internal to the UDFB is another call to the **MC_GearIn** function.
  - Therefore, the MasterID, SlaveID, RatioNumerator, RatioDenominator, Acceleration, Deceleration, and Jerk inputs are the same values as the main ladder **MC_GearIn** function input values, both with the Buffer input of 0 (zero).
  - This assures that the initial starting master/slave ratio will transition to the new Kollmorgen UDFB ratio smoothly.
- This UDFB changes the master/slave ratio defined by the **MC_GearIn** function based on the error between the analog input and the analog set-point.
  - The magnitude of the ratio and the rate of the ratio change is defined by the Kp, Ti, Td PID gain values.
  - The new ratio calculated is output at the NewRatio output.
- The **RatioLimitPercent** input is the maximum and minimum theoretical new ratio that can be changed.
  - This provides a +/- window limit around the running ratio to prevent unwanted motion in the event of a web break or analog feedback failure.

3.8.7.3.1.1 Example 1

NOTE

This example assumes the analog feedback device is located **after** (or downstream in the process) the feedroll axis.

```
RatioNumerator = 1
RatioDenominator = 2     Therefore the master/slave starting ratio is
0.5000000
ErrorCaclMode = 0
DeviceFBValue = 6
DeviceSetPoint = 4    Therefore error 6 - 4 = 2
Kp = 0.005
Ti = 0
Td= 0
```

From the equation:

New RatioDenominator = (RatioDenominator - Kp * error)

- The new RatioDenominator = (2 - 0.005*2) = 1.99.
- The new master/slave running ratio is 1 / 1.99 = 0.502512562.
- Since the master/slave ratio is greater than the previous ratio, the slave axis is going faster and the tension is reduced.

3.8.7.3.1.2 Example 2

NOTE

This example assumes the analog feedback device is located **before** (or upstream in the process) the feedroll axis.

```
RatioNumerator = 1
RatioDenominator = 2     Therefore the master/slave starting ratio is
0.5000000
ErrorCaclMode = 1
DeviceFBValue = 6
DeviceSetPoint = 4    Therefore error is 4 - 6 = -2
Kp = 0.005
Ti = 0
Td= 0
```

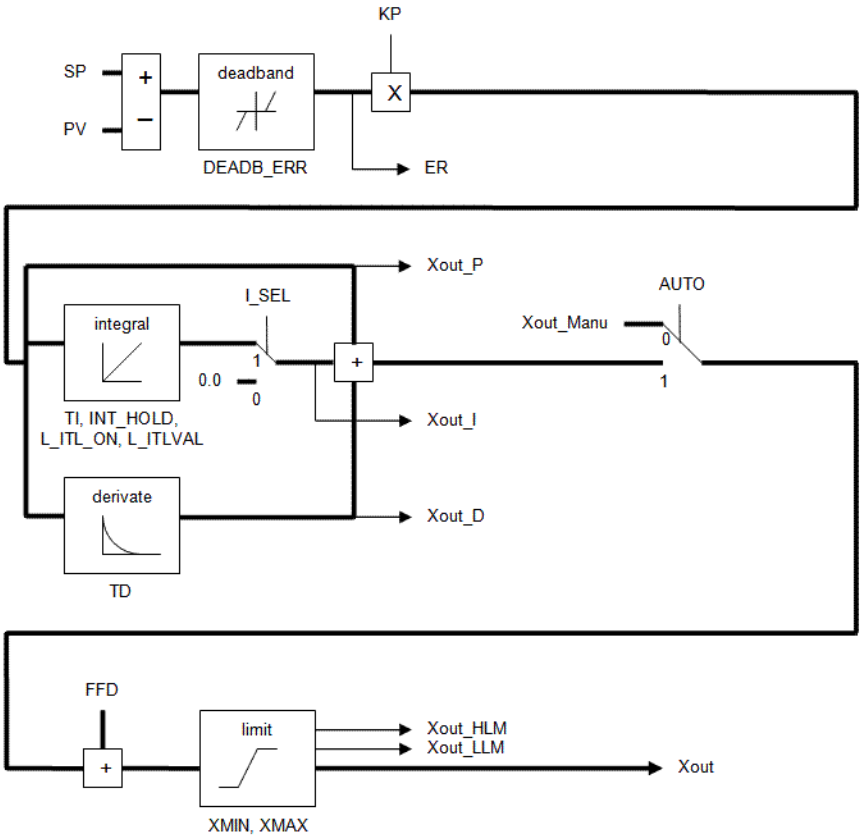
From the equation:

New RatioDenominator = (RatioDenominator – (Kp * error))

- The new RatioDenominator = (2 + 0.005*2) = 2.01.
- The new master/slave running ratio is 1 / 2.01 = 0.497512437.
- Since the master/slave ratio is less than the previous ratio, the slave axis is going slower and the tension is reduced.

3.8.7.3.2 PID Function in KAS

There is a PID function in KAS that could be used for the PID control section in the UDFB.



3.8.7.3.3 Programming Tips

- The "FB_FirstOrderDigitalFilter" (→ p. 180) UDFB can be used to decrease excess dither on the analog input.
  - The filtered analog value is then used at the DeviceFBValue input of the MCFB_ GearedWebTension UDFB.

- The assumption is an **MC_GearIn** function block is first called in the main ladder and these initial values are used at the inputs for the UDFB.
  - The resolution of the initial **MC_GearIn** the **RatioNumerator** and **RatioDenominator** inputs are directly related to the resolution of the calculated master/slave ratio (from the UDFB inputs) and may need to be scaled accordingly.

3.8.7.3.3.1 Example 1

No scaling
Initial MC_GearIn input RatioNumerator = 2.
Initial MC_GearIn input RatioDenominator = 1 then initial Master/Slave ratio = 2.

UDFB input RatioNumerator = 2.
UDFB input RatioDenominator = 1 then UDFB Master/Slave ratio = 2.

UDFB input DeviceFBValue = 4.
UDFB input DeviceFBSetpoint = 3 then Device PID error = 1 assume KP = 1, Ti and Td =0.

New UDFB RatioNumerator = Current RatioNumerator – PID error = 2 – 1 = 1 then new UDFB Master/Slave ratio = 1.

Resolution = Master/Slave ratio:PID Error ratio = 1:1.

- The resolution is so coarse that a change of 1 for the error output of the PID creates a Master/Slave ratio change of 1.
- This result is a significant change to the slave velocity that will probably cause excess slack or web breakage.

3.8.7.3.3.2 Example 2

Scaling value = 1000
Initial MC_GearIn input RatioNumerator = 2.
Initial MC_GearIn input RatioDenominator = 1 then initial Master/Slave ratio = 2.

UDFB input RatioNumerator = 2000
UDFB input RatioDenominator = 1000 then UDFB Master/Slave ratio = 2.

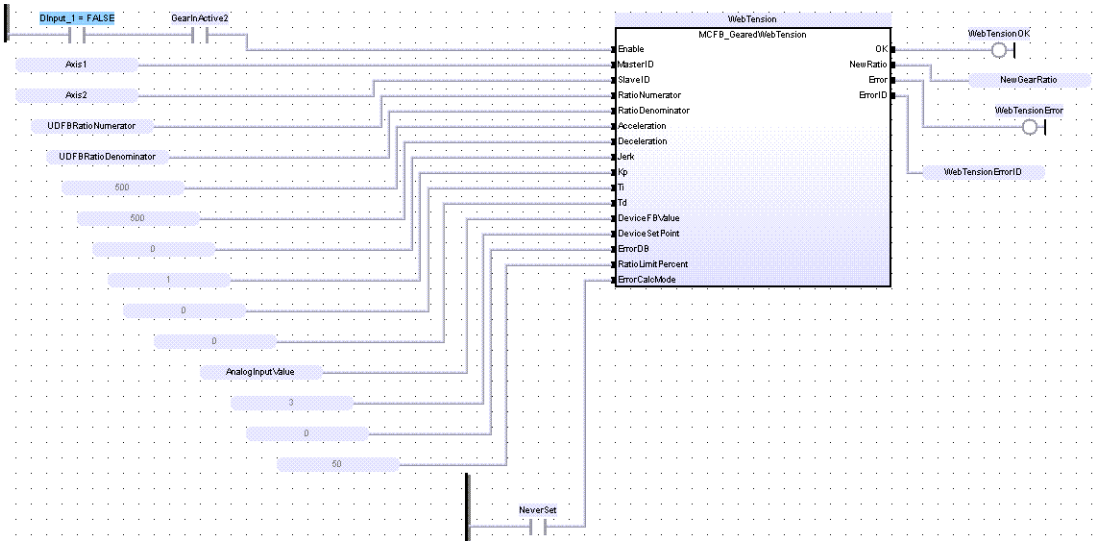
UDFB input DeviceFBValue = 4.
UDFB input DeviceFBSetpoint = 3 then Device PID error = 1 assume KP = 1, Ti and Td =0.

New UDFB RatioNumerator = Current RatioNumerator – PID error = 2000– 1 =1999 then new UDFB Master/Slave ratio = 1999.

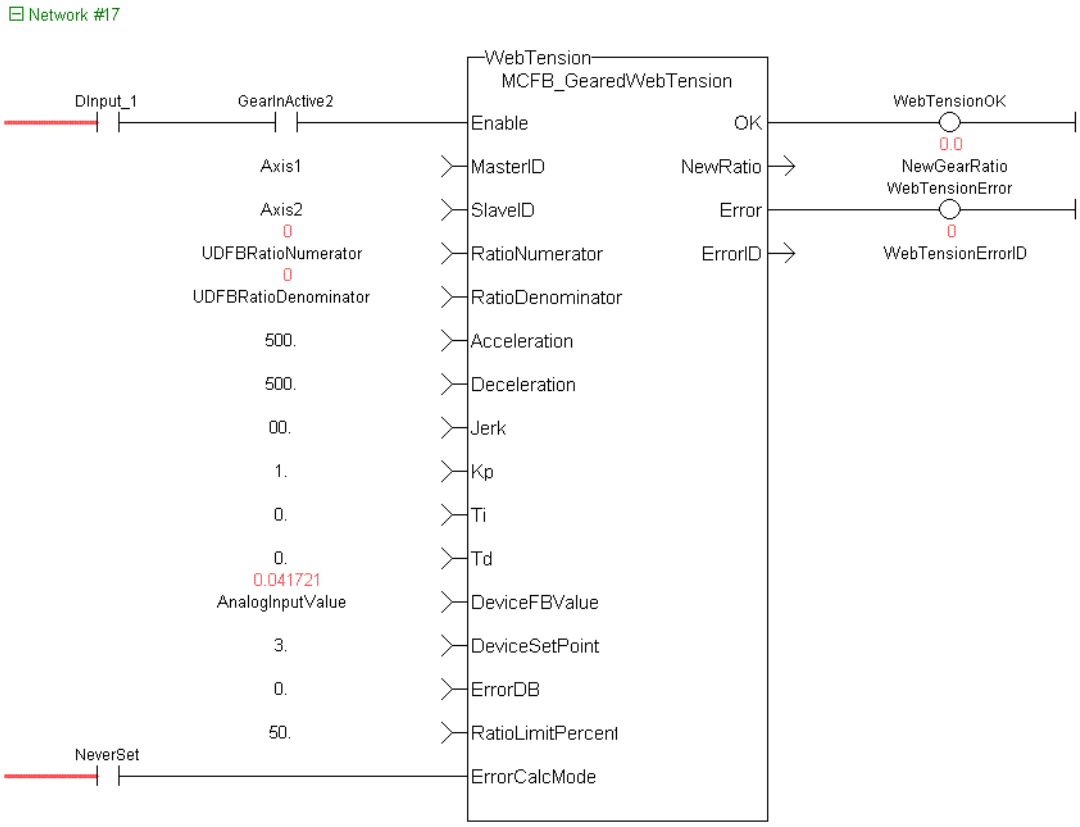
Resolution = Master/Slave ratio: PID Error ratio = 2000:1.

- This resolution is much finer than Example 1.
- For a change of 1 for the error output of the PID this creates a Master/Slave ratio change of 1999.
- This results is a slower rate of change to the slave velocity that is more suited to good tension in a machine process.

3.8.7.4 FBD Language Example



3.8.7.5 FFLD Language Example



3.8.7.6 IL Language Example

Not available.

3.8.7.7 ST Language Example

```
Inst_MCFB_GearedWebTension( DInput_1, Axis1, Axis2,
    ratioNumerator, ratioDenominator, 500.0, 500.0, 0.0, 1.0, 0.0, 0.0,
    AnalogInputValue, 3.0, 0.0, 50.0, NeverSet);
WebTensionOK       := Inst_MCFB_GearedWebTension.OK;
NewGearRatio       := Inst_MCFB_GearedWebTension.NewRatio;
WebTensionHasError := Inst_MCFB_GearedWebTension.Error;
WebTensionErrorID  := Inst_MCFB_GearedWebTension.ErrorID;
```

See Also

"FB_FirstOrderDigitalFilter" (→ p. 180)

3.9 UDFB - PLC

These are the Kollmorgen UDFB PLC function and function blocks.

Name	Description
FB_ElapseTime	Keeps track of the time (oTotalOnTime) a Boolean input variable is on.
FB_FirstOrderDigitalFilter	Defined to filter an Analog signal.
FB_PWDutyOutput	Accepts an input value between the minimum and maximum input range and converts this to a duty cycle percentage.

Name	Description
FB_ScaleInput - Scaling Analog IO	Converts un-scaled DINT values from Analog Inputs into user units of type LREAL.
FB_ScaleOutput - Scaling Analog IO	Converts un-scaled LREAL values from a PLC program into units of type DINT.
FB_TemperaturePID	Provides PID temperature control with auto tuning.
PipeNetwork_FFLD - Special Function	Calls the PNCode function block in FFLD POUs.
ProfilesCode_FFLD - Special Function	Calls the Profiles Code Function Block in FFLD POUs.

3.9.1 FB_ElapseTime

Function Block - Keeps track of the time (oTotalOnTime) a Boolean input variable is on.

3.9.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
iEN00	BOOL	FALSE, TRUE	N/A	FALSE	Enable for the block.
iVariable	BOOL	FALSE, TRUE	N/A	FALSE	The variable to be tracked.

3.9.1.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	FALSE, TRUE	N/A	Function block is OK. This output follows the state on iEN00 input.
oTotalOnTime	TIME	0ms - 24hr	ms	The amount of time the iVariable is turned on.
oInProcess	BOOL	FALSE, TRUE	N/A	The state of the block's execution whether or not it is still keeping track of time.
oDone	BOOL	FALSE, TRUE	N/A	The state of the block's execution whether or not it is completed.

3.9.1.3 Remarks

- Once the iEN00 enable input is high, the Kollmorgen UDFB keeps track of the total time iVariable is on.
- If iVariable changes to an off state while iEN00 is on, the oTotalOnTime stops.
  - oTotalOnTime will start to add again once iVariable changes state to high.
  - As long as the iEN00 input is on, iVariable can change states many times.
  - The oTotalOnTime will reflect only the total time that iVariable has been on.
- While iVariable is still TRUE, oInProcess is also TRUE and oDone is FALSE.
  - Once iVariable is FALSE, oInProcess is FALSE and oDone is TRUE.
- If the iEN00 input goes off, oTotalOnTime stops counting and the Kollmorgen UDFB execution stops.
  - To restart the timer turn iEN00 on again.
  - This resets oTotalOnTime to zero and counting begins once iVariable is also on.

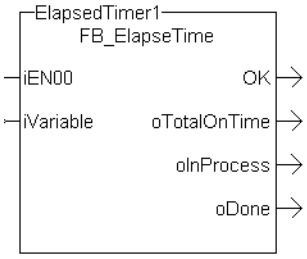


Figure 3-24: FB_ElapseTime

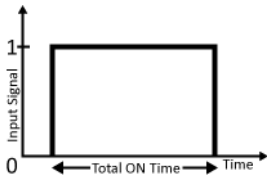
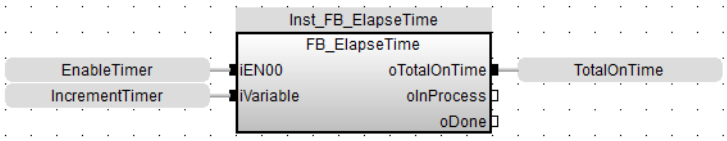


Figure 3-25: MFB_ElapseTime – Time Diagram

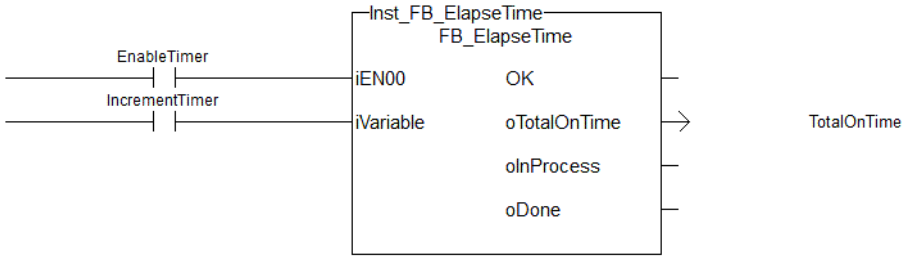
3.9.1.3.1 Usage

- Enable the block by setting iEN00 to TRUE
- Either manually set iVariable to TRUE or have the application set this variable to TRUE
- Once oDone returns TRUE, read the oTotalOnTime to find out how long iVariable was on.

3.9.1.4 FBD Language Example



3.9.1.5 FFLD Language Example



3.9.1.6 IL Language Example

Not available.

3.9.1.7 ST Language Example

```
//Keep track of total time that IncrememtTimer variable is TRUE while
EnableTimer variable is true
//Timer will be reset when EnableTimer variable is false
Inst_FB_ElapseTime( EnableTimer, IncrementTimer );
TotalOnTime := Inst_FB_ElapseTime.oTotalOnTime;
```

3.9.2 FB_FirstOrderDigitalFilter

Function Block - Defined to filter an Analog signal.

3.9.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	No range	N/A	No default	Enables execution. Used in the FFLD editor only.
AnalogInput	INT	No range	N/A	No default	Analog Input from transducer.
FilterGain	REAL	1 - 0.05	N/A	No default	Filter gain.

3.9.2.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	0, 1	N/A	Execution successful.
FilterOutput	REAL	0, 1	N/A	Filtered analog input value.

3.9.2.3 Remarks

In any control system with an analog feedback signal present, there is the risk of unwanted noise and jitter that can compromise the signal integrity yielding a less the desirable system.

- This Kollmorgen UDFB provides a digital first order filter of an analog feedback signal from an LVDT, tension transducer, potentiometer, encoder, resolver, or some other like device.
- The amount of filtering is based on a gain value and can provide no filter to full filter conditioning.

This image shows the function or function block I/O.

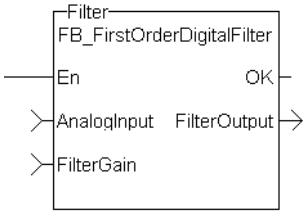


Figure 3-26: CBS First Order Digital Filter

3.9.2.3.1 Usage

When using this UDFB, the Enable (EN) input should always be energized to provide the desired filtering.

- The **AnalogInput** input is the unfiltered analog feedback signal from an LVDT, tension transducer, potentiometer, or some other like device.
- The **FilterGain** defines the amount of filtering to be used.
  - The range of the gain is from 1.0 or no filtering to 0.05 or the maximum filtering.
- The **FilterOutput** is the filtered analog input.
  - It is used as an input to some other function block or UDFB that has an analog input (e.g., the MCFB_GearedWebTension UDFB).
- The implementation of the digital first order filter is for PLCopen.
- The equation is defined as: `Input*Gain+Output*(1-Gain) = Output`.
- The steady state filter delay with a gain of 0.8 is shown in the "Filter Input Delay" (→ p. 181) table.

3.9.2.3.1.1 Filter Input Delay

Example: Filter Input Delay

FilterGain	FilterInput	FilterOutput
0.8	0	0
	100	80
	100	96
	100	99.2
	100	99.84
	100	99.968
	100	99.9936
	100	99.99872
	100	99.999744
	100	99.9999488
	100	99.99998976
	100	99.99999795
	100	99.99999959
	100	99.99999992
	100	99.99999998
	100	100
	100	100
	100	100
	100	100
	100	100
	100	100
	100	100
	100	100

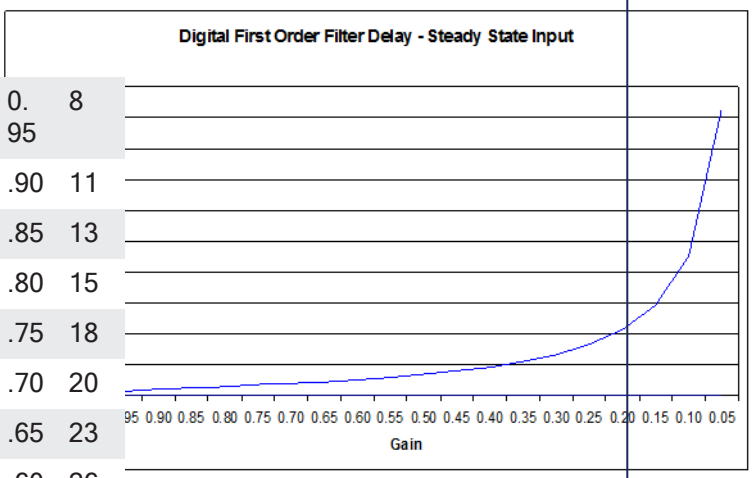
3.9.2.3.1.2 Filter Delay Tn

In this table:

- The numbers of filter delays for a steady state analog input at a given gain are listed.
- The range of the filter gain is between 1.00 and 0.05.
- For a filter gain of 0.8 there is a delay of 15 time constants with a time constant defined as the rate the UDFB is scanned or executed in the application.
  - Example: If the UDFB was executed every millisecond a gain of 0.8 would provide a filter delay of 15ms.
  - Conversely a gain of 1.00 provides zero filtering and the output signal follows the input signal, and a gain of 0.05 provides the most filtering for 463 ms.

Example Table: filter Delay TYn

Gain	Filter Delay Tn	Graph
0.95	8	
.90	11	
.85	13	
.80	15	
.75	18	
.70	20	
.65	23	
.60	26	
.55	30	
.50	35	
.45	40	
.40	47	
.35	56	
.30	66	
.25	83	
.20	107	
.15	146	
.10	226	
.05	463	



3.9.2.3.1.3 Example: Filter Input Lag - Random Input

A real world analog input is most always a varying feedback signal.

Example Table: Filter Input Lag - Random Input

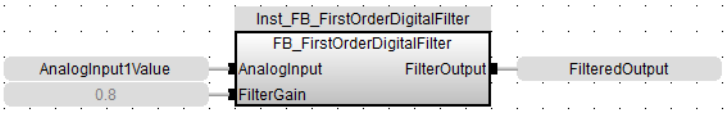
In this table, an initial input of 100, a gain of 0.8, and a random variability of 10%.Filter Input

Filter Input	Filter Current Output	Amount of Input Filtering	Random Filter % Variation
0	0	0	10%
100	80	-20	
97.38903813	93.9112305	-3.477807626	
92.67638093	92.92335084	0.246969915	
94.12988912	93.88858146	-0.241307655	
103.0835564	101.2445614	-1.838994993	

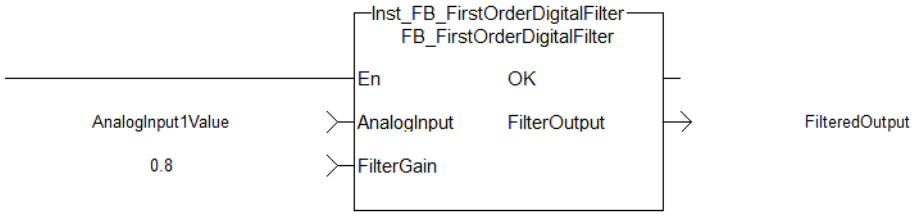
Filter Input	Filter Current Output	Amount of Input Filtering	Random Filter % Variation
91.16845433	93.18367575	2.015221422	
93.23936976	93.22823096	-0.011138803	
94.90272089	94.56782291	-0.334897986	
103.3070737	101.5592235	-1.747850153	
96.83149418	97.77704005	0.945545867	
96.35024002	96.63560002	0.285360007	
99.82417525	99.1864602	-0.637715045	
105.0792636	103.9007029	-1.178560685	
97.36988208	98.67604626	1.306164172	
107.82502	105.9952253	-1.829794752	
97.7886524	99.42996698	1.641314572	
108.2038024	106.4490353	-1.754767081	
91.58527607	94.55802792	2.972751845	
93.6783421	93.85427926	0.175937164	
102.8695349	101.0664838	-1.803051129	
93.95916817	95.3806313	1.421463121	
108.6579707	106.0025028	-2.655467871	
109.3425748	108.6745604	-0.668014397	
103.9066	104.8601921	0.953592077	
92.30112142	94.81293555	2.511814127	
109.4460726	106.5194452	-2.926627416	
94.88799896	97.21428821	2.326289251	
105.4738635	103.8219484	-1.651915057	
102.988167	103.1549233	0.166756284	
92.92925408	94.97438792	2.045133846	
95.58185568	95.46036213	-0.121493552	
109.414248	106.6234708	-2.790777178	
106.5661311	106.577599	0.011467953	
99.85857253	101.2023778	1.343805301	
107.865421	106.5328124	-1.332608643	
92.19683177	95.0640279	2.867196126	
104.8558146	102.8974573	-1.958357346	
104.5140236	104.1907104	-0.323313268	
104.3675014	104.3321432	-0.035358206	
109.2704266	108.2827699	-0.987656683	
101.4962729	102.8535723	1.35729941	

Filter Input	Filter Current Output	Amount of Input Filtering	Random Filter % Variation
92.19199163	94.32430776	2.132316128	
99.13065312	98.16938405	-0.961269073	
103.5068114	102.4393259	-1.067485466	
109.502983	108.0902516	-1.412731426	
99.05504822	100.8620889	1.80704068	
94.97711299	96.15410817	1.176995182	
107.1063597	104.9159094	-2.190450308	
91.12245188	93.88114339	2.758691504	
108.130314	105.2804799	-2.849834129	
104.2923832	104.4900025	0.197619344	
101.3775072	102.0000062	0.62249907	
100.5303014	**100.0399168**	**-0.490384645**	**Averages**

3.9.2.4 FBD Language Example



3.9.2.5 FFLD Language Example



3.9.2.6 IL Language Example

Not available.

3.9.2.7 ST Language Example

```
//Filter analog input signal with a gain of 0.8 to remove noise
FilteredOutput:= Inst_FB_FirstOrderDigitalFilter( AnalogInput1Value, 0.8 );
```

3.9.3 FB_PWDutyOutput

Function Block - Accepts an input value between the minimum and maximum input range and converts this to a duty cycle percentage.

3.9.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
uEN01	BOOL	0, 1	N/A	No default	Enable for the block.
uInputCommand	REAL	0 to --	N/A	No default	Signal Input. Sometimes the output of a PID block.
uPeriod	TIME	0 to --	N/A	No default	Period of the duty cycle.
uMaxInput	REAL	uMinInput to --	N/A	No default	uInputCommand at or above this number that sets DutyOutput =1.
uMinInput	REAL	0 (zero)to uMaxInput	N/A	No default	uInputCommand at or below this number set DutyOutput = 0 (zero).
uMaxTime	TIME	uMinTime to uPeriod	N/A	No default	Maximum on time for the output.
uMinTime	TIME	0 (zero)to uMaxTime	N/A	No default	Minimum on-time for the PW Duty Output.

3.9.3.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	0, 1	N/A	Function block is operational.
TM1	TIME	0 (zero) to uPeriod	N/A	On-time of the Output.
DutyOutput	BOOL	0, 1	N/A	PW signal (switching between 0 and 1). DutyOutput is set to 0 (zero) when the function block is not active (not eabled by the first input).

3.9.3.3 Remarks

The output is then cycled on and off over the period of the duty cycle at the duty cycle percentage.

- To have the output ON time range from 0 (zero) to the duty cycle period, the minimum should be set to 0 (zero) and the maximum to the duty cycle period.
- If the calculated duty cycle is:
  - Based on the input and the range values are less than the minimum ON time (MinTime), the output does not come on.
  - Between or equal to the range values the output is cycled by the duty cycle.
  - Greater than the maximum ON time (MaxTime) the output will remain on.

This image shows the function or function block I/O.

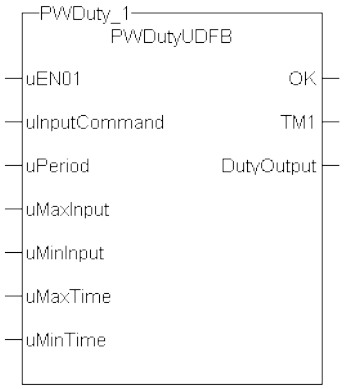


Figure 3-27: Pulse Width Duty Cycle

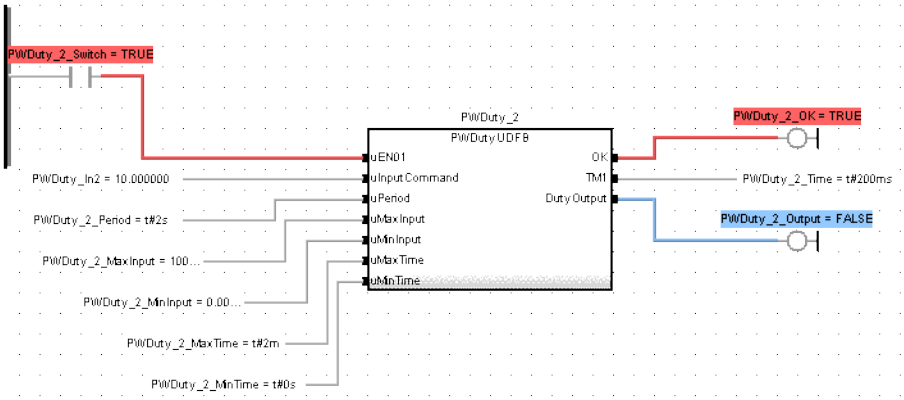
3.9.3.3.1 Usage

Flash a warning light for operators.

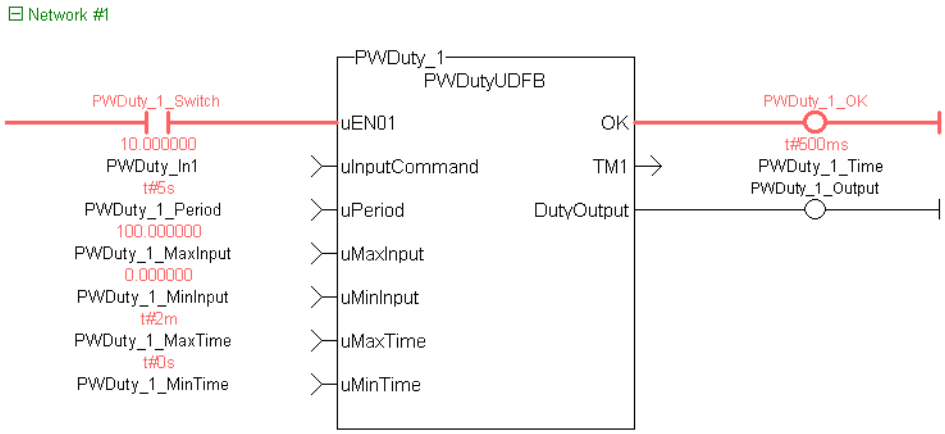
3.9.3.3.2 Function Block Calculations

```
IF (uInputCommand - uMinInput) < 0 then     //If Command less than MinInput
turn out put off
   DutyOutput := 0;
ELSIF (uInputCommand - uMaxInput) > 0 then     //If Command greater than
MaxInput turn out put on
   DutyOutput := 1;
ELSE
   DutyCycle := (uInputCommand - uMinInput)/(uMaxInput -
uMinInput);     //Calculate Duty Cycle
   ONTimeFromInput := DutyCycle * any_to_REAL(uPeriod) ;     //Calculate
Ontime

   IF  any_to_TIME(ONTimeFromInput)  < uMinTime then
      DutyOutput := 0;
   ELSIF any_to_TIME(ONTimeFromInput) > uMaxTime then
      DutyOutput := 1;
   ELSE
      TM1 := any_to_TIME(ONTimeFromInput) ;
      TM0 :=  uPERIOD - TM1;     //Calculate offtime
      DutyOutput := Inst_blinkA( 1 , TM0 , TM1 );     //Use BlinkA function
to set PW output
   END_IF ;
END_IF ;
```

3.9.3.3.3 Function Block Diagram



3.9.3.3.4 Ladder Diagram



3.9.3.4 IL Language Example

Not available.

3.9.3.5 ST Language Example

```
Inst_FB_PWDutyOutput( PWDuty_3_Switch, PWDuty_In3, PWDuty_3_Period, PWDuty_3_
MaxInput, PWDuty_3_MinInput, PWDuty_3_MaxTime, PWDuty_3_MinTime);
PWDuty_3_OK:=Inst_FB_PWDutyOutput.OK;
PWDuty_3_Time:=Inst_FB_PWDutyOutput.TM1;
PWDuty_3_Output:=Inst_FB_PWDutyOutput.DutyOutput;
```

See Also

Timers

3.9.4 FB_ScaleInput - Scaling Analog IO

Function Block - Converts un-scaled DINT values from Analog Inputs into user units of type LREAL.

3.9.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
InputSignal	DINT	0, 4	N/A	No default	Un-scaled input signal.
InputMin	DINT	0, 4	N/A	No default	Minimum value of accepted input signal range.
InputMax	DINT	0, 4	N/A	No default	Maximum value of accepted input signal range.
OutputMin	LREAL	0, 4	N/A	No default	Output value mapped to the InputMin.
OutputMax	LREAL	0, 4	N/A	No default	Output value mapped to the InputMax.

3.9.4.2 Outputs

Output	Data Type	Range	Unit	Description
OutputSignal	LREAL		N/A	Scaled value of the Input Signal with type converted to LREAL. Stays within specified Min/Max output values.
OutsideRange	BOOL		N/A	TRUE if the InputSignal is outside the range setup by min/max values, otherwise FALSE.

3.9.4.3 Remarks

- Scale s DINT to LREAL.
- The input signal is converted based on a linear mapping automatically calculated by two points entered.
- InputMin is mapped to OutputMin, InputMax is mapped to OutputMax, and all values in between are scaled automatically.
- If an input value is not between the selected Min/Max, the Boolean output OutsideRange turns TRUE.
  - The OutputSignal is set to the corresponding OutputMin or OutputMax value.

This image shows the function or function block I/O.

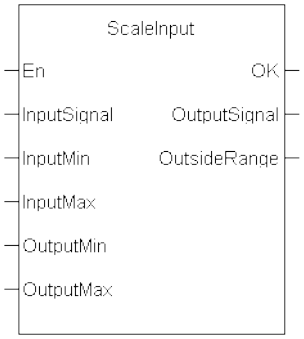
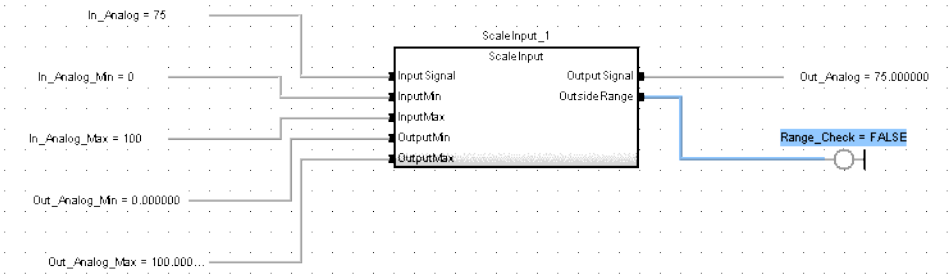


Figure 3-28: Scale Input

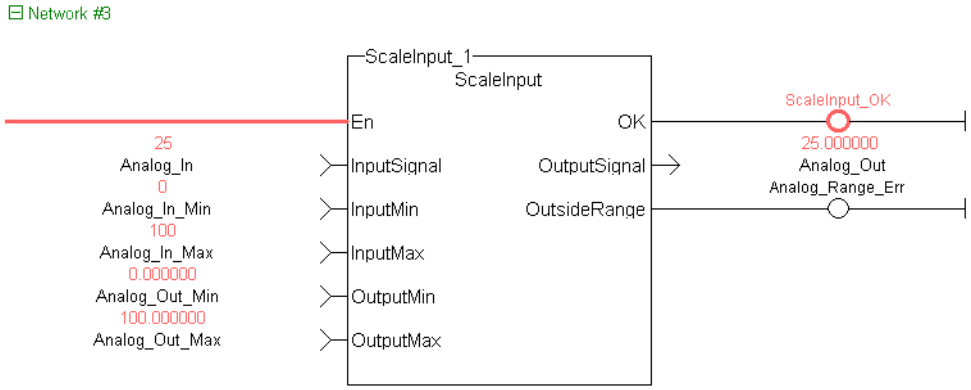
3.9.4.3.1 Usage

Scale an analog signal from a drive.

3.9.4.4 FBD Language Example



3.9.4.5 FFLD Language Example



3.9.4.6 IL Language Example

3.9.4.7 ST Language Example

```
//Scale an integer based analog input signal into floating point LREAL
variable
ScaleInput_1( Analog_In, Analog_In_Min, Analog_In_Max, LREAL_Out_Min, LREAL_
Out_Max );
LREAL_OutputSignal:= ScaleInput_1.OutputSignal;
Analog_Range_Err:= ScaleInput_1.OutsideRange;
```

See Also

"FB_ScaleOutput - Scaling Analog IO" (→ p. 190)

3.9.5 FB_ScaleOutput - Scaling Analog IO

Function Block - Converts un-scaled LREAL values from a PLC program into units of type DINT.

3.9.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
InputSignal	LREAL	0, 4	N/A	No default	Un-scaled input signal.
InputMin	LREAL	0, 4	N/A	No default	Minimum value of accepted input signal range.
InputMax	LREAL	0, 4	N/A	No default	Maximum value of accepted input signal range.
OutputMin	DINT	0, 4	N/A	No default	Output value mapped to the InputMin.
OutputMax	DINT	0, 4	N/A	No default	Output value mapped to the InputMax.

3.9.5.2 Outputs

Output	Data Type	Range	Unit	Description
OutputSignal	DINT		N/A	Scaled value of the Input Signal with type converted to DINT. Stays within specified Min/Max output values.
OutsideRange	BOOL		N/A	TRUE if the InputSignal is outside the range setup by min/max values, otherwise FALSE.

3.9.5.3 Remarks

- The scale LREAL to DINT can be mapped to an analog output.
- The input signal is converted based on a linear mapping automatically calculated by two points entered.
- InputMin is mapped to OutputMin, InputMax is mapped to OutputMax, and all values in between are scaled automatically.
- If an input value is not between the selected Min/Max, the Boolean output OutsideRange turns TRUE.
  - The OutputSignal is set to the corresponding OutputMin or OutputMax value.

This image shows the function or function block I/O.

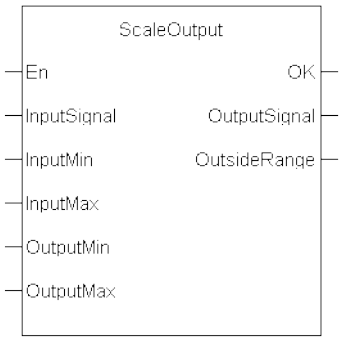
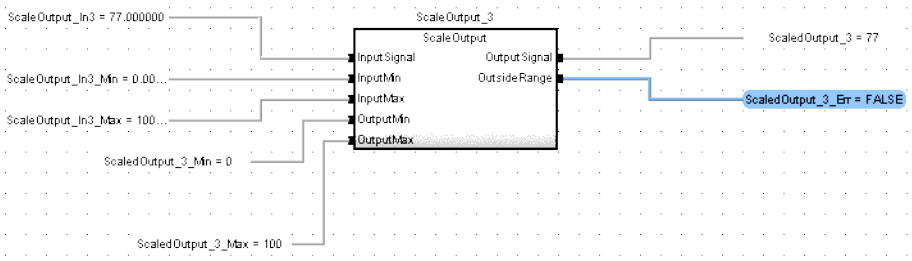


Figure 3-29: Scale Output

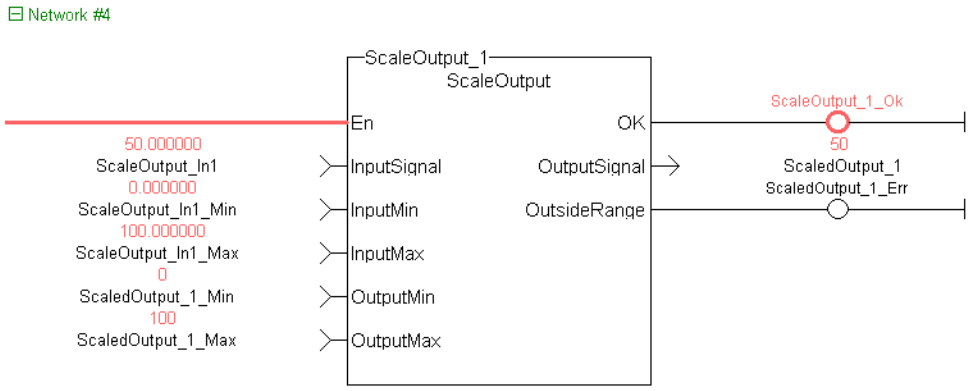
3.9.5.3.1 Usage

Scale an analog signal from a drive.

3.9.5.4 FBD Language Example



3.9.5.5 FFLD Language Example



3.9.5.6 IL Language Example

Not available.

3.9.5.7 ST Language Example

```
Inst_ScaleOutput1( ScaleOutput_In2, ScaleOutput_In2_Min,ScaleOutput_In2_Max,
ScaledOutput_2_Min, ScaledOutput_2_Max );
ScaledOutput_2:=Inst_ScaleOutput1.OutputSignal;
ScaledOutput_2_Err:=Inst_ScaleOutput1.OutsideRange;
```

See Also

"FB_ScaleInput - Scaling Analog IO" (→ p. 188)

3.9.6 FB_TemperaturePID

Function Block - Provides PID temperature control with auto tuning.

3.9.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
iEN	BOOL				Enable function.
iREFTEMP	LREAL				Reference temperature [°C].
iTEMP	LREAL				Actual temperature [°C].
iSTART	BOOL				Start PID or auto tuning.
iMODE	BOOL				• FALSE-automatic. • TRUE-tuning.
iKP	LREAL				PID Proportional Gain.
iKI	LREAL				PID Integral Gain.
iKD	LREAL				PID Derivative Gain.
iTC1S	BOOL				Sampling Time is 1s.
iTC5S	BOOL				Sampling Time is 5s.
iPROGCYCLE	LREAL				Execution time of the function [ms].

3.9.6.2 Outputs

Output	Data Type	Range	Unit	Description
oOK	BOOL			Function enabled.
oCHECKSTABLE	BOOL			TRUE when checking if ambient temperature is stable.
oTUNESTART	BOOL			Tuning is started.
oTUNEOK	BOOL			Tuning is completed.
oTAU	LREAL			System Time Constants.
oPWM	BOOL			PWM command for heater.
oERR	BOOL			Function error.
oERRID	INT			Function ID error (in case of oERR=TRUE).
oPOWER	LREAL			% of power requested from heater (100%=full power).
oTICK	BOOL			Pulse every sampling time.

3.9.6.3 Remarks

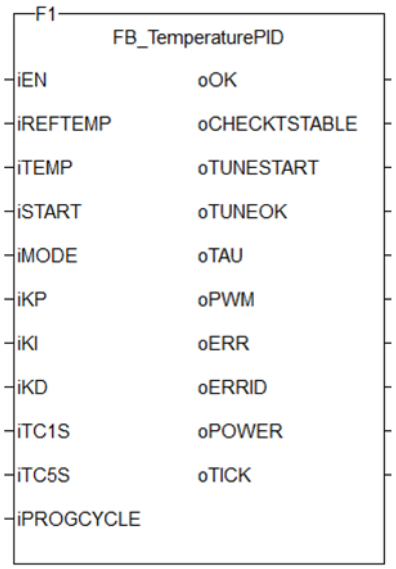
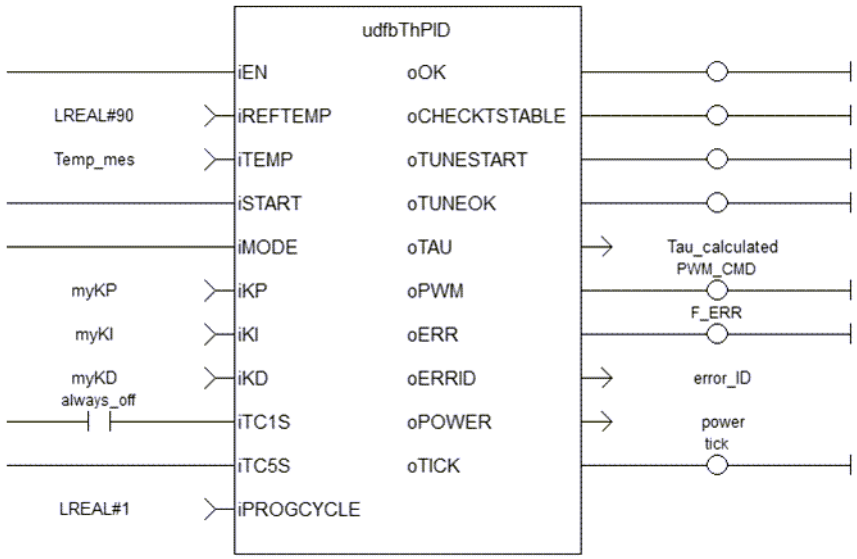


Figure 3-30: FB_TemperaturePID

3.9.6.3.1 Usage

3.9.6.3.1.1 Tuning Process



Tuning consists of three steps.

1. Verify the ambient temperature is stable.
   The measured `delta_temp=Tmax-Tmin` must be lower than `0.1*Tmax`.
   This step takes 10 cycles (`10*iTC5s` or `10*iTC1s`).
   The tuning fails (`oERR=TRUE, oERRID=1`) if the ambient temperature is greater than `0.1*Tmax`, otherwise `Tamb=(Tmax+Tmin)/2`.
2. Start tuning Phase1: output **oPWM** is kept TRUE until the final measured temperature **iTEMP** gets over `iREFTEMP/2`.
   After that, **oPWM** is kept LOW.
3. Start tuning Phase2.
   With **oPWM** kept LOW, the temperature gets down until the final value is lower than `[(iREFTEMP/2-Tamb)*0.368+Tamb]`.

After, PID gains are calculated as:

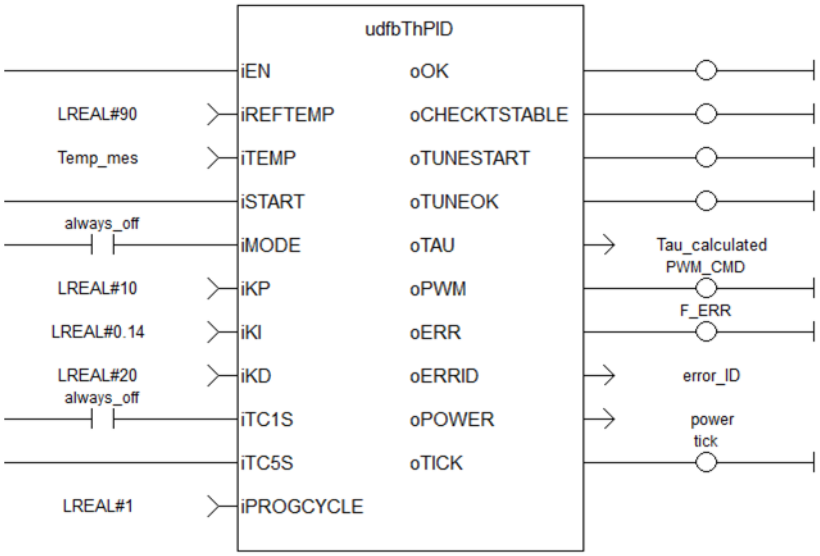
```
Kp=10
Ki=0.14
delta_time = time to complete Phase2

Kd=SQRT(delta_time)*7
```

The tuning is completed.

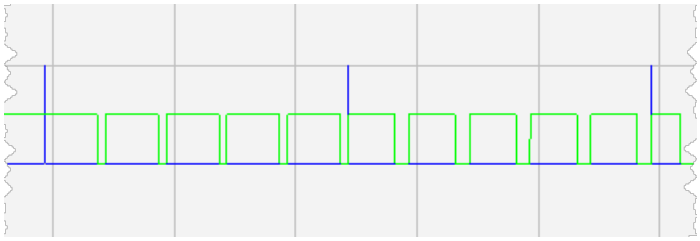
TIP

oTAU may be useful for setting the proper sampling time (1s or 5s).

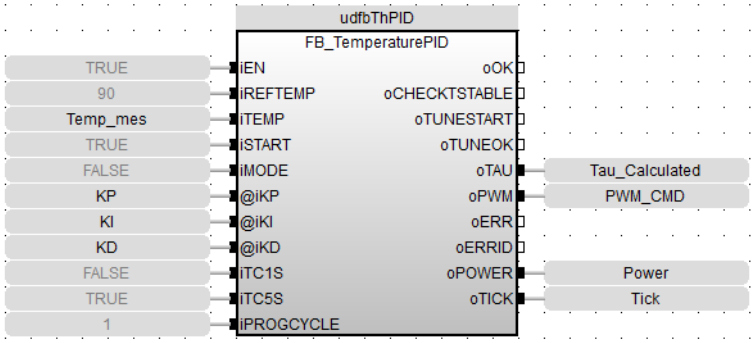
3.9.6.3.1.2 Start PID Controller



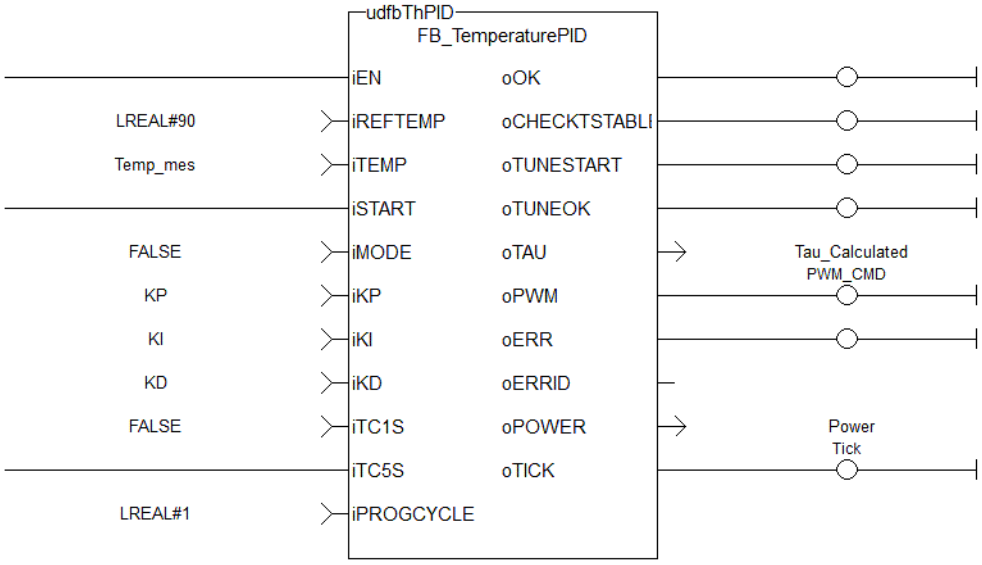
- When starting the PID controller:
  - The output **oPWM** is modulated five times within the sampling time (blue line is **oTICK**, green line is **oPWM**).
  - Each pulse length depends on output **oPOWER** (100%=full length).



3.9.6.4 FBD Language Example



3.9.6.5 FFLD Language Example



3.9.6.6 IL Language Example

Not available.

3.9.6.7 ST Language Example

```
//Run PID function with determined proportional, integral, and derivative
gains
//send PWM output to command heater
udfbThPID( TRUE, 90, Temp_mes, TRUE, FALSE, KP, KI, KD, FALSE, TRUE, 1);
Tau_Calculated := udfbThPID.oTAU;
PWM_CMD := udfbThPID.oPWM;
Power := udfbThPID.oPOWER;
Tick := udfbThPID.oTICK;
```

3.9.7 PipeNetwork_FFLD - Special FFLD Function

Pipe Network

Function - Calls the PNCode function block in FFLD POUs.

3.9.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Request to initialize the Pipe Network.

Input	Data Type	Range	Unit	Default	Description
cmdID	DINT	N/A	N/A	No default	Commands used to start and initialize the Pipe Network. • MLPN_CREATE_OBJECTS – Create Pipe Network • MLPN_POWER_ON – Power on all axes • MLPN_POWER_OFF – Power off all axes • MLPN_ACTIVATE – Activate the pipes • MLPN_CONNECT – Connect the axes to the pipes • MLPN_DEACTIVATE – Deactivate the pipes

3.9.7.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL		N/A	Returns TRUE when the function successfully executes.

3.9.7.3 Remarks

- It starts and initializes the Pipe Network, based on the command specified by `cmdID`.
  - Internally this function calls the Function Block PNCode.
- This is a special function.
  - It should only be used in Pipe Network applications that contain FFLD POUs that call PNCode.
  - To use this function, Pipe Network must be declared as a global variable in the dictionary.
- Calling this function instead of PNCode in FFLD POUs eliminates this compile error that occurs after modifying the Pipe Network using the Pipe Network editor:

```
Controller:PLC:Main: NW1(1,14): PNCode: Invalid block height
```

- The compile error is generated because the number of outputs on PNCode can vary.
  - This occurs after modifying the original Pipe Network using the Pipe Network editor.
- The new PNCode Function Block is not automatically updated in any FFLD POUs, reflecting the new outputs.
  - Manually update each PNCode function block call in any FFLD POUs to correct this problem.

TIP

The Pipe Network FFLD Application Template is a good example of how to use this Function.
See Pipe Network 2-Axes Template - FFLD.

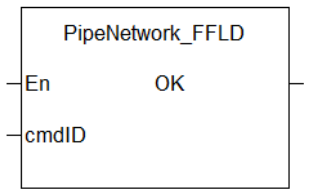
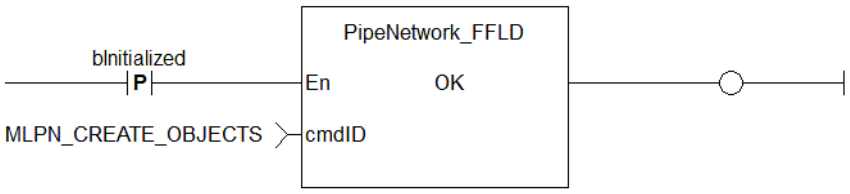


Figure 3-31: PipeNetwork_FFLD

3.9.7.4 FBD Language Example

Not available.

3.9.7.5 FFLD Language Example



3.9.7.6 IL Language Example

Not available.

3.9.7.7 ST Language Example

Not available.

See Also

- Design Motion with Pipe Network
- Initialize and Start a Pipe Network
- PLCopen 2-Axes Template - FFLD

3.9.8 ProfilesCode_FFLD - Special FFLD Function

Pipe Network

Function - Calls the Profiles Code Function Block in FFLD POUs.

3.9.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Request to initialize the Pipe Network.
cmdID	DINT	N/A	N/A	No default	Commands used to start and initialize the Pipe Network. • MLPR_CREATE_PROFILES - Creation and initialization of profiles.

3.9.8.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL		N/A	Returns TRUE when the function successfully executes.

3.9.8.3 Remarks

- Internally, this function calls the Function Block ProfilesCode.
- This is a special function.
  - It should only be used in applications that contain FFLD POUs that call ProfilesCode.
  - To use this function, Profiles must be declared as a global variable in the dictionary.
- Calling this function instead of ProfilesCode in FFLD POUs eliminates this compile error that occurs after adding a new Profile to the project tree.

```
Controller:PLC:Main: NW1(1,14): ProfilesCode: Invalid block height
```

- The compile error is generated because the number of outputs on ProfilesCode can vary.
  - This occurs after adding a new profile to the project tree.
- The ProfilesCode Function Block is not automatically updated in any FFLD POUs, reflecting the new outputs.
  - You must manually update each ProfilesCode Function Block call in any FFLD POUs to correct this problem.
  - If you use this function instead, you no longer need to manually update each ProfilesCode Function Block in FFLD.

TIP

The Pipe Network and PLCopen 2 Axis FFLD Application Templates are two examples of how to use this function.
See Pipe Network 2-Axes Template - FFLD or PLCopen 2-Axes Template - FFLD.

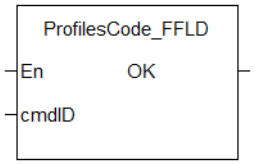
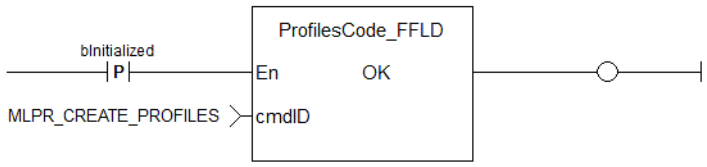


Figure 3-32: ProfilesCode_FFLD

3.9.8.4 FBD Language Example

Not available.

3.9.8.5 FFLD Language Example



3.9.8.6 IL Language Example

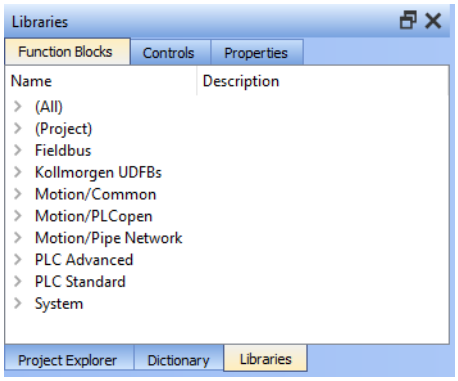
Not available.

3.9.8.7 ST Language Example

Not available.

4 Motion Library

The Motion Library (ML) (for Pipe Network Concept and PLCopen®) is in the Function Blocks tab of the Library toolbox.



- The KAS function library contains ML function blocks used to integrate motion in a PLC program.
- ML function blocks can be used in four of the IEC 61131-3 languages: ST, FBD, FFLD and IL.
- For Sequential Function Chart (SFC) programs, ML function blocks are used as part of SFC Steps or SFC Transitions defined with FBD, FFLD, IL, or ST languages.

4.1 Motion / Pipe Network

The KAS-IDE function library contains Motion Library (ML) function blocks (FBs) used to integrate motion from a Pipe Network in a PLC program.

These are the ML function blocks:

Function	Description
Block	Manage the blocks: Create / activate.
Motion	Prepare the physical motion part: init, reset, start, stop.
Pipe Block	Manage each specific Pipe Block: Read / write parameters.
Pipe Network	Manage the Pipe Network: Create / activate.

IMPORTANT

Pipe Network code is generated automatically by the compiler.
Do not try to modify it.

4.1.1 Adder

Name	Description
MLAddInit	Initializes an Adder Pipe Block for use in a PLC Program with user-defined settings.
MLAddReadOff1	Returns the offset value of the first entry of an Adder block.
MLAddReadOff2	Returns the offset value of the second entry of an Adder block.
MLAddReadRatio1	Returns the ratio value of the first entry of an Adder block.
MLAddReadRatio2	Returns the ratio value of the second entry of an Adder block.
MLAddWriteInput	Sets the source of an input of an Adder Pipe Block.

Name	Description
MLAddWriteOff1	Set the offset value of the first entry of the Adder block.
MLAddWriteOff2	Set the offset value of the second entry of the Adder block.
MLAddWriteRat1	Set the ratio value of the first entry of the Adder block.
MLAddWriteRat2	Set the ratio value of the second entry of the Adder block.

4.1.1.1 MLAddInit

Function - Initializes an Adder Pipe Block for use in a PLC Program with user-defined settings.

4.1.1.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of a created Pipe Block.
Offset1	LREAL	No range	N/A	No default	Sets the Offset value of the first entry of an Adder object.
Offset2	LREAL	No range	N/A	No default	Sets the Offset value of the second entry of an Adder object.
Ratio1	LREAL	No range	N/A	No default	Sets the Ratio value of the first entry of an Adder object.
Ratio2	LREAL	No range	N/A	No default	Sets the Ratio value of the second entry of an Adder object.

4.1.1.1.2 Outputs

```
Adder Block Output = Ratio1*Input1 + Offset1 + Ratio2*Input2 + Offset2
```

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the Adder Pipe Block is initialized.

4.1.1.1.3 Remarks

NOTE

Adder objects are normally created in the Pipe Network using the graphical engine.
You do not have to add **MLAddInit** function blocks to their programs.
Parameters are entered directly in pop-up windows and the code is automatically added to the current project.

- Function block is automatically called if an Adder Block is added to the Pipe Network.
  - User-defined settings are entered in the **Pipe Blocks Properties** screen.
- The Pipe Block is assigned ratios and offsets for both inputs.
  - After an Adder block is initialized, the inputs need to be selected using the "MLAddWriteInput" (→ p. 206) function block or graphically using the Pipe Network.

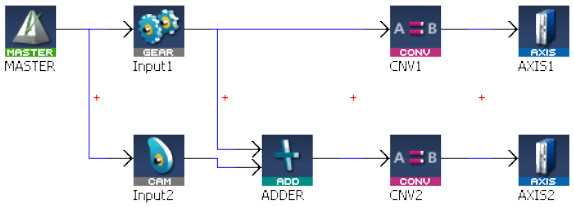
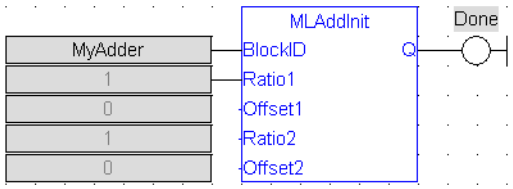
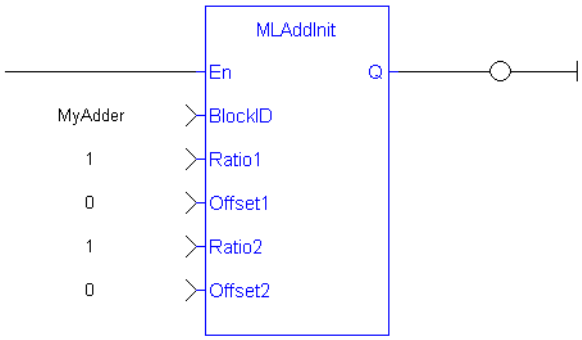


Figure 4-1: MLAddInit

4.1.1.1.4 FBD Language Example



4.1.1.1.5 FFLD Language Example



4.1.1.1.6 IL Language Example

Not available.

4.1.1.1.7 ST Language Example

```
//Create and Initiate a Trigger object
MyAdder := MLBlkCreate( 'MyAdder', 'ADDER' );
MLAddInit( MyAdder, 1.0, 0.0, 1.0, 0.0 );
```

See Also

- "MLAddReadOff1" (→ p. 201)
- "MLAddReadRatio1" (→ p. 204)
- "MLAddWriteInput" (→ p. 206)
- "MLBlkCreate" (→ p. 212)

4.1.1.2 MLAddReadOff1

Pipe Network

Function - Returns the offset value of the first entry of an Adder block.

4.1.1.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Adder object.

4.1.1.2.2 Outputs

```
Adder Block Output = Ratio1*Input1 + Offset1 + Ratio2*Input2 + Offset2
```

Output	Data Type	Range	Unit	Description
Offset	LREAL			Returns the offset value of the first entry of an Adder block.

4.1.1.2.3 Remarks

- Offset1 shifts the value of the first input to the block before its added to the second input.
- Can change the offset value with "MLAddWriteOff1" (→ p. 207) function block.

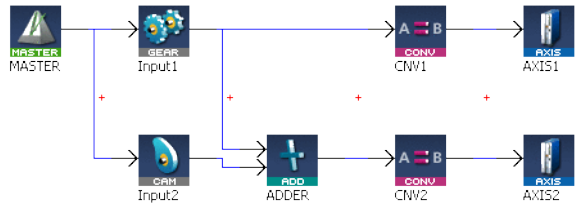
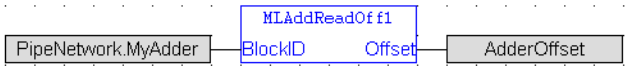
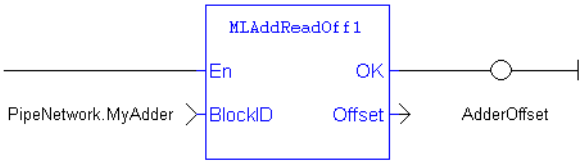


Figure 4-2: MLAddReadOff1

4.1.1.2.4 FBD Language Example



4.1.1.2.5 FFLD Language Example



4.1.1.2.6 IL Language Example

Not available.

4.1.1.2.7 ST Language

```
//Save the offset value of first entry to the Adder block
AdderOffset := MLAddReadOff1( PipeNetwork.MyAdder );
```

See Also

- "MLAddReadOff2" (→ p. 202)
- "MLAddReadRatio1" (→ p. 204)
- "MLAddWriteOff1" (→ p. 207)
- "MLAddWriteRat1" (→ p. 210)

4.1.1.3 MLAddReadOff2

Pipe Network

Function - Returns the offset value of the second entry of an Adder block.

4.1.1.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Adder object.

4.1.1.3.2 Outputs

```
Adder Block Output = Ratio1*Input1 + Offset1 + Ratio2*Input2 + Offset2
```

Output	Data Type	Range	Unit	Description
Offset	LREAL			Returns the offset value of the second entry of an Adder block.

4.1.1.3.3 Remarks

- Offset2 shifts the value of the second input to the block before its added to the first input.
- Can change the offset value with "MLAddWriteOff2" (→ p. 208) function block.

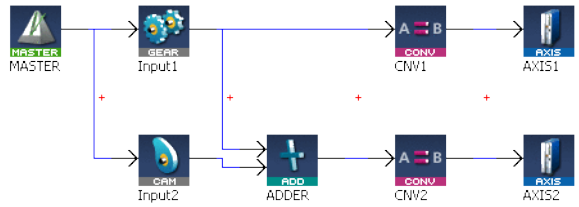
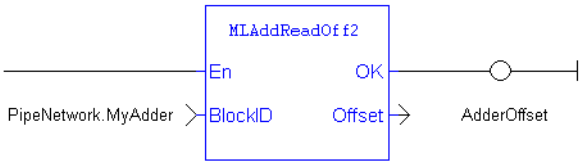


Figure 4-3: MLAddReadOff2

4.1.1.3.4 FBD Language Example

4.1.1.3.5 FFLD Language Example



4.1.1.3.6 IL Language Example

Not available.

4.1.1.3.7 ST Language Example

```
//Save the offset value of second entry to the Adder block
AdderOffset := MLAddReadOff2( PipeNetwork.MyAdder );
```

See Also

- "MLAddReadOff1" (→ p. 201)
- "MLAddReadRatio2" (→ p. 205)
- "MLAddWriteOff2" (→ p. 208)
- "MLAddWriteRat2" (→ p. 211)

4.1.1.4 MLAddReadRatio1

Pipe Network ✓

Function - Returns the ratio value of the first entry of an Adder block.

4.1.1.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Adder object.

4.1.1.4.2 Outputs

```
Adder Block Output = Ratio1*Input1 + Offset1 + Ratio2*Input2 + Offset2
```

Output	Data Type	Range	Unit	Description
Ratio	LREAL			Returns the ratio value of the first entry of an Adder block.

4.1.1.4.3 Remarks

- Ratio1 amplifies the value of the first input to the block before its added to the second input.
- Can change the ratio value with "MLAddWriteRat1" (→ p. 210) function block.

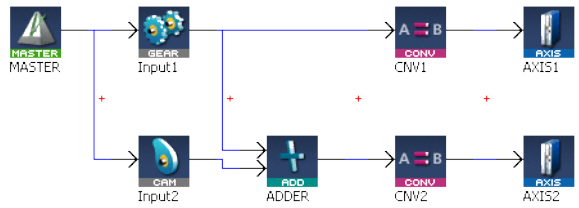
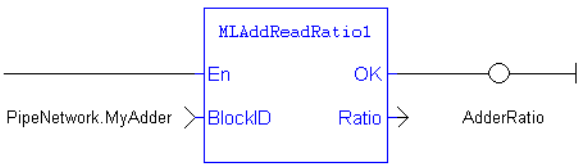


Figure 4-4: MLAddReadRatio1

4.1.1.4.4 FBD Language Example

4.1.1.4.5 FFLD Language Example



4.1.1.4.6 IL Language Example

Not available.

4.1.1.4.7 ST Language Example

```
//Save the ratio value of first entry to the Adder block
AdderRatio := MLAddReadRatio1( PipeNetwork.MyAdder );
```

See Also

- "MLAddReadOff1" (→ p. 201)
- "MLAddReadOff2" (→ p. 202)
- "MLAddReadRatio2" (→ p. 205)
- "MLAddWriteRat1" (→ p. 210)

4.1.1.5 MLAddReadRatio2

Pipe Network ✓

Function - Returns the ratio value of the second entry of an Adder block.

4.1.1.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Adder object.

4.1.1.5.2 Outputs

```
Adder Block Output = Ratio1*Input1 + Offset1 + Ratio2*Input2 + Offset2
```

Output	Data Type	Range	Unit	Description
Ratio	LREAL	No range	N/A	Returns the ratio value of the second entry of an Adder block.

4.1.1.5.3 Remarks

- Ratio2 amplifies the value of the second input to the block before its added to the first input.
- Can change the ratio value with "MLAddWriteRat2" (→ p. 211) function block.

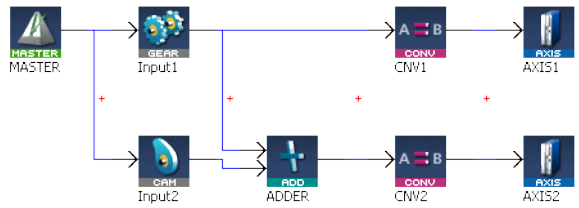


Figure 4-5: MLAddReadRatio2

4.1.1.5.4 FBD Language Example

4.1.1.5.5 FFLD Language Example

4.1.1.5.6 IL Language Example

Not available.

4.1.1.5.7 ST Language Example

```
//Save the ratio value of second entry to the Adder block
AdderRatio := MLAddReadRatio2( PipeNetwork.MyAdder );
```

See Also

- "MLAddReadOff1" (→ p. 201)
- "MLAddReadOff2" (→ p. 202)
- "MLAddReadRatio1" (→ p. 204)
- "MLAddWriteRat2" (→ p. 211)

4.1.1.6 MLAddWriteInput

Pipe Network ✓

Function - Sets the source of an input of an Adder Pipe Block.

4.1.1.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Adder object.
InputID	DINT	1, 2	N/A	No default	Select first or second input to the Adder object.
InputBlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Pipe Block. This is an input to the Adder object.

4.1.1.6.2 Outputs

```
Adder Block Output = Ratio1*Input1 + Offset1 + Ratio2*Input2 + Offset2
```

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the input to the Adder object is set.

4.1.1.6.3 Remarks

- Function block is automatically called if an Adder Block is connected to other blocks in the Pipe Network.

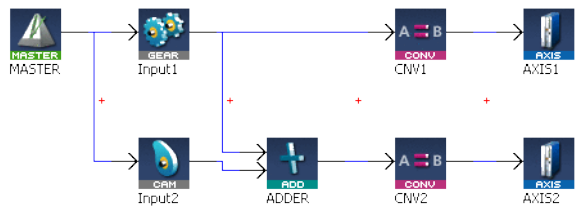


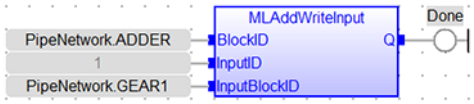
Figure 4-6: MLAddWriteInput

NOTE

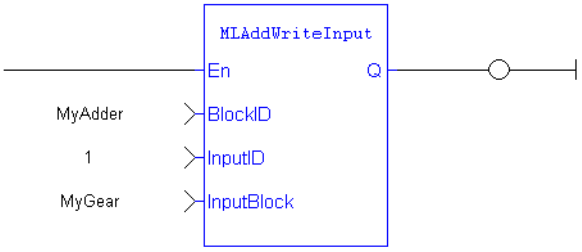
Adder objects are normally created in the Pipe Network using the graphical engine.
You do not have to add **MLAddWriteInput** function blocks to their programs.

Blocks are connected with lines in the Pipe Network.
The code is automatically added to the current project.

4.1.1.6.4 FBD Language Example



4.1.1.6.5 FFLD Language Example



4.1.1.6.6 IL Language Example

Not available.

4.1.1.6.7 ST Language Example

```
//Set the first input of an Adder pipeblock to be connected to the output of
GEAR1 pipeblock
MLAddWriteInput( PipeNetwork.ADDER, 1, PipeNetwork.GEAR1 );
```

See Also

- "MLAddInit" (→ p. 200)
- "MLAddReadOff1" (→ p. 201)
- "MLAddReadRatio1" (→ p. 204)
- "MLBlkCreate" (→ p. 212)

4.1.1.7 MLAddWriteOff1

Pipe Network

Function - Set the offset value of the first entry of the Adder block.

4.1.1.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Adder object.
InputID	LREAL	No range	N/A	No default	Desired new value for the Adder Object's Offset1.

4.1.1.7.2 Outputs

```
Adder Block Output = Ratio1*Input1 + Offset1 + Ratio2*Input2 + Offset2
```

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the Offset value for input one is set.

4.1.1.7.3 Remarks

IMPORTANT

Changes made to the Offset of an Adder block are executed immediately and can cause an axis position to jump.

- Offset1 shifts the value of the first input to the block before its added to the second input.

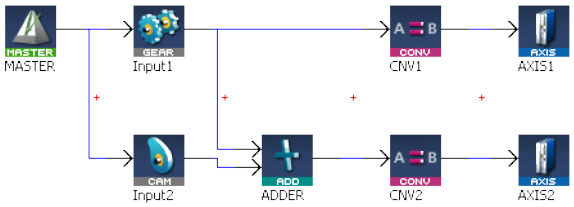
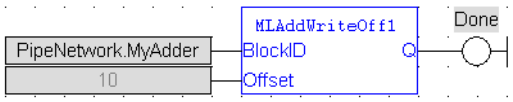
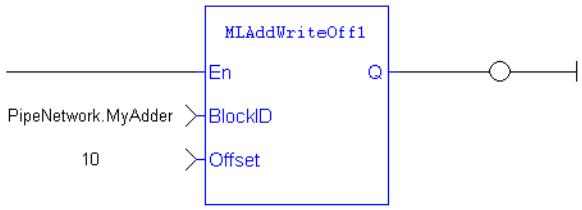


Figure 4-7: MLAddWriteOff1

4.1.1.7.4 FBD Language Example



4.1.1.7.5 FFLD Language Example



4.1.1.7.6 IL Language Example

Not available.

4.1.1.7.7 ST Language Example

```
//Change the offset value of first entry to the Adder block to 10
MLAddWriteOff1( PipeNetwork.MyAdder, 10 );
```

See Also

- "MLAddReadOff1" (→ p. 201)
- "MLAddReadRatio1" (→ p. 204)
- "MLAddWriteOff2" (→ p. 208)
- "MLAddWriteRat1" (→ p. 210)

4.1.1.8 MLAddWriteOff2

Pipe Network

Function - Set the offset value of the second entry of the Adder block.

4.1.1.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Adder object.
InputID	LREAL	No range	N/A	No default	Desired new value for the Adder Object's Offset2.

4.1.1.8.2 Outputs

```
Adder Block Output = Ratio1*Input1 + Offset1 + Ratio2*Input2 + Offset2
```

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the Offset value for input two is set.

4.1.1.8.3 Remarks

IMPORTANT

Changes made to the Offset of an Adder block are executed immediately and can cause an axis position to jump.

- Offset2 shifts the value of the second input to the block before its added to the first input.

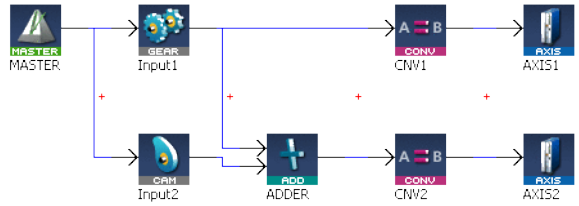
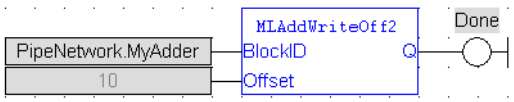
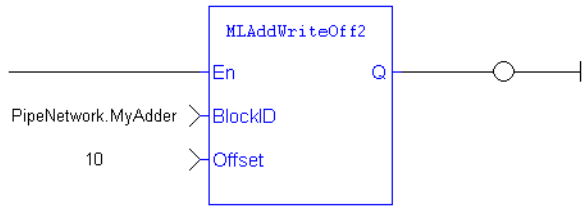


Figure 4-8: MLAddWriteOff2

4.1.1.8.4 FBD Language Example



4.1.1.8.5 FFLD Language Example



4.1.1.8.6 IL Language Example

Not available.

4.1.1.8.7 ST Language Example

```
//Change the offset value of second entry to the Adder block to 10
MLAddWriteOff2( PipeNetwork.MyAdder, 10 );
```

See Also

- "MLAddReadOff2" (→ p. 202)
- "MLAddReadRatio2" (→ p. 205)
- "MLAddWriteOff1" (→ p. 207)
- "MLAddWriteRat2" (→ p. 211)

4.1.1.9 MLAddWriteRat1

Pipe Network

Function - Set the ratio value of the first entry of the Adder block.

4.1.1.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Adder object.
Ratio	LREAL	No range	N/A	No default	Designated new value for the Adder Object's Ratio1.

4.1.1.9.2 Outputs

```
Adder Block Output = Ratio1*Input1 + Offset1 + Ratio2*Input2 + Offset2
```

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the Ratio value for input one is set.

4.1.1.9.3 Remarks

IMPORTANT

Changes made to the Ratio of an Adder block are executed immediately and can cause an axis position to jump.

- Ratio1 amplifies the value of the first input to the block before its added to the second input.

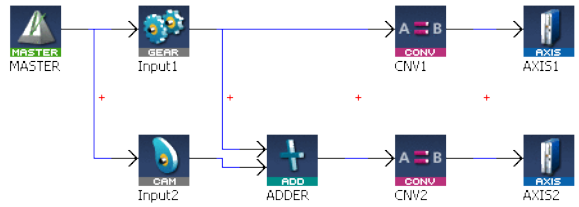
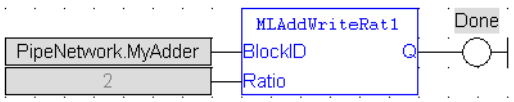
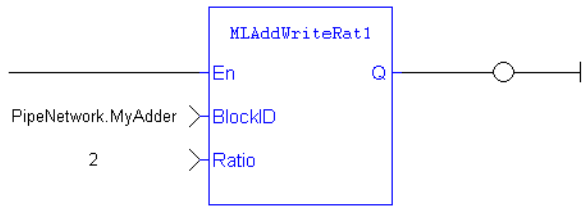


Figure 4-9: MLAddWriteRat1

4.1.1.9.4 FBD Language Example



4.1.1.9.5 FFLD Language Example



4.1.1.9.6 IL Language Example

Not available.

4.1.1.9.7 ST Language Example

```
//Change the ratio value of first entry to the Adder block to 2
MLAddWriteRat1( PipeNetwork.MyAdder, 2 );
```

See Also

- "MLAddReadOff1" (→ p. 201)
- "MLAddReadRatio1" (→ p. 204)
- "MLAddWriteOff1" (→ p. 207)
- "MLAddWriteRat2" (→ p. 211)

4.1.1.10 MLAddWriteRat2

Pipe Network ✓

Function - Set the ratio value of the second entry of the Adder block.

4.1.1.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Adder object.
Ratio	LREAL	No range	N/A	No default	Designated new value for the Adder Object's Ratio2.

4.1.1.10.2 Outputs

```
Adder Block Output = Ratio1*Input1 + Offset1 + Ratio2*Input2 + Offset2
```

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the Ratio value for input two is set.

4.1.1.10.3 Remarks

IMPORTANT

Changes made to the Ratio of an Adder block are executed immediately and can cause an axis position to jump.

- Ratio2 amplifies the value of the second input to the block before its added to the first input.

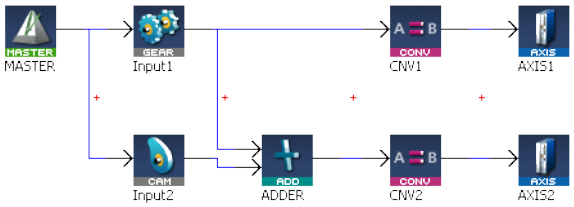
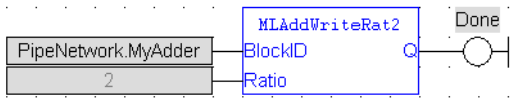
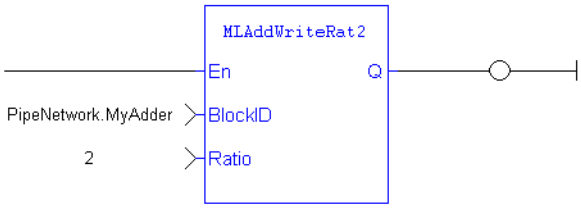


Figure 4-10: MLAddWriteRat2\

4.1.1.10.4 FBD Language Example



4.1.1.10.5 FFLD Language Example



4.1.1.10.6 IL Language Example

Not available.

4.1.1.10.7 ST Language Example

```
//Change the ratio value of second entry to the Adder block to 2
MLAddWriteRat2 ( PipeNetwork.MyAdder, 2 );
```

See Also

- "MLAddReadOff2" (→ p. 202)
- "MLAddReadRatio2" (→ p. 205)
- "MLAddWriteOff2" (→ p. 208)
- "MLAddWriteRat1" (→ p. 210)

4.1.2 Block

Name	Description
MLBlkCreate	Creates a new Pipe Block object.
MLBlkIsReady	Verifies if a Pipe Block currently has a function running.
MLBlkReadModPos	Gets the value of the period of a block in user units.
MLBlkReadOutVal	Gets the output value of a selected Pipe Block.
MLBlkWriteModPos	Sets the value of the period of a block in user units.

4.1.2.1 MLBlkCreate

Function - Creates a new Pipe Block object.

4.1.2.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Name	STRING	No range	N/A	No default	Designated name for the newly created Pipe Block.
Type	STRING	No range	N/A	No default	The type of Pipe Block to create. Examples: MASTER, GEAR, PHASER, etc.

4.1.2.1.2 Outputs

Output	Data Type	Range	Unit	Description
ID	DINT			Assigned ID number of the created Block.

4.1.2.1.3 Remarks

- Before a Pipe Block is Initialized, the function needs to be created and assigned an ID number.
- The **MLBlkCreate** function is automatically called if a Block is added to the Pipe Network.

NOTE

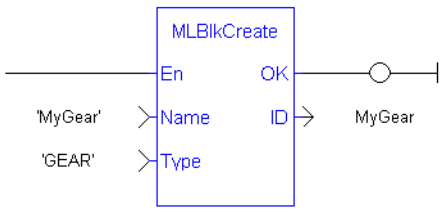
Pipe Blocks are normally created in the Pipe Network using the graphical engine.
You do not have to add **MLBlkCreate** function blocks to their programs.
The code with **MLBlkCreate** commands are automatically generated and called in a program with **Pipe Network(MLPN_CREATE_OBJECTS)**.

TIP

This function should be called after "MLMotionInit" (→ p. 549) is called and before "MLMotionStart" (→ p. 552) is called.

4.1.2.1.4 FBD Language Example

4.1.2.1.5 FFLD Language Example



4.1.2.1.6 IL Language Example

Not available.

4.1.2.1.7 ST Language Example

```
//Create a new  GEAR Pipe Block named "MyGear"
MyGear := MLBlkCreate( 'MyGear', 'GEAR' );
```

See Also

- "MLAxisInit" (→ p. 246)
- See Step 3 in Initialize and Start a Pipe Network.

4.1.2.2 MLBlkIsReady

Pipe Network

Function - Verifies if a Pipe Block currently has a function running.

4.1.2.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of a created Pipe Block.

4.1.2.2.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	N/A	N/A	Returns TRUE if no function of a specified Pipe Block is running.

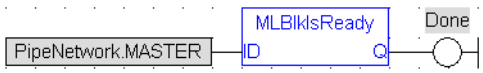
4.1.2.2.3 Remarks

- Returns TRUE if **no function** of a specified Pipe Block is running.
- Returns FALSE if the selected Pipe Block has a function running.
- Same return value as the **.Q** output of a specific function itself.

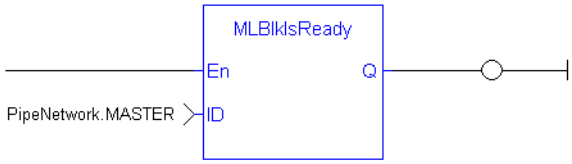
NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

4.1.2.2.4 FBD Language Example



4.1.2.2.5 FFLD Language Example



4.1.2.2.6 IL Language Example

Not available.

4.1.2.2.7 ST Language Example

```
//Check if the MST Pipe Block named "MASTER" has a function running
IsReady := MLBlkIsReady( PipeNetwork.MASTER );
```

See Also

- "MLBlkReadModPos" (→ p. 214)
- "MLBlkReadOutVal" (→ p. 215)

4.1.2.3 MLBlkReadModPos

Pipe Network

Function - Gets the value of the period of a block in user units.

4.1.2.3.1 Inputs

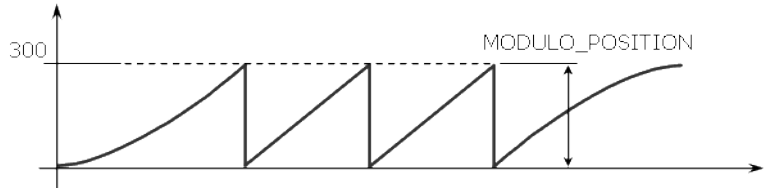
Input	Data Type	Range	Unit	Default	Description
ID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of a created Pipe Block.

4.1.2.3.2 Outputs

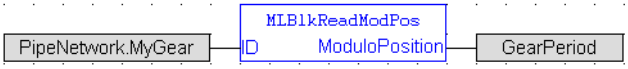
Output	Data Type	Range	Unit	Description
ModuloPosition	LREAL		User units	Current period value for selected Pipe Block.

4.1.2.3.3 Remarks

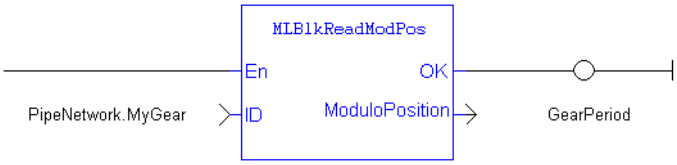
- The output value of a block is reset each time it reaches its period value.



4.1.2.3.4 FBD Language Example



4.1.2.3.5 FFLD Language Example



4.1.2.3.6 IL Language Example

Not available.

4.1.2.3.7 ST Language Example

```
//Return and save the Period of a Pipe Block
GearPeriod := MLBlkReadModPos( PipeNetwork.MyGear );
```

See Also

- "MLBlkCreate" (→ p. 212)
- "MLBlkReadOutVal" (→ p. 215)
- "MLBlkWriteModPos" (→ p. 216)

4.1.2.4 MLBlkReadOutVal

Pipe Network

Function - Gets the output value of a selected Pipe Block.

4.1.2.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of a created Pipe Block.

4.1.2.4.2 Outputs

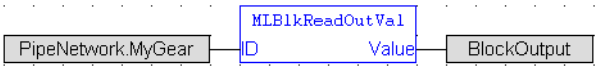
Output	Data Type	Range	Unit	Description
Value	LREAL			Current output value of the selected Pipe Block.

4.1.2.4.3 Remarks

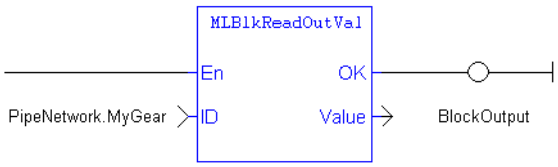
NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

4.1.2.4.4 FBD Language Example



4.1.2.4.5 FFLD Language Example



4.1.2.4.6 IL Language Example

Not available.

4.1.2.4.7 ST Language Example

```
//Save the output of a Gear Pipe Block
BlockOutput := MLBlkReadOutVal( PipeNetwork.MyGear );
```

See Also

- "MLBlkCreate" (→ p. 212)
- "MLBlkReadModPos" (→ p. 214)

4.1.2.5 MLBlkWriteModPos

Pipe Network

Function - Sets the value of the period of a block in user units.

4.1.2.5.1 Inputs

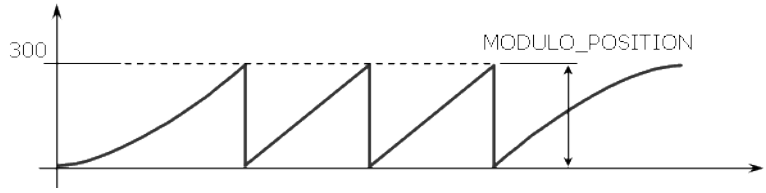
Input	Data Type	Range	Unit	Default	Description
ID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of a created Pipe Block.
ModuloPosition	LREAL	No range	User units	No default	Designated new Period Value for selected Pipe Block.

4.1.2.5.2 Outputs

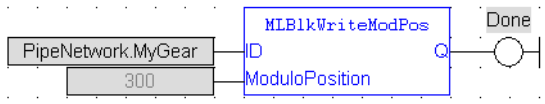
Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the function block is executed.

4.1.2.5.3 Remarks

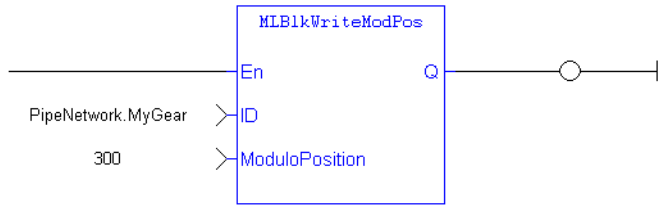
- The output value of a block is reset each time it reaches its period value.



4.1.2.5.4 FBD Language Example



4.1.2.5.5 FFLD Language Example



4.1.2.5.6 IL Language Example

Not available.

4.1.2.5.7 ST Language Example

```
//Set the Period of a Pipe Block to 300
MLBlkWriteModPos( PipeNetwork.MyGear, 300 );
```

See Also

- "MLBlkCreate" (→ p. 212)
- "MLBlkReadModPos" (→ p. 214)
- "MLBlkReadOutVal" (→ p. 215)

4.1.3 Pipe

Name	Description
MLPipeAct	Activates a pipe.
MLPipeAddBlock	Adds a Pipe Block to a pipe.
MLPipeCreate	Create a new pipe object.
MLPipeDeact	Deactivates a pipe.

4.1.3.1 MLPipeAct

Pipe Network

Function - Activates a pipe.

4.1.3.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
PipeID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of a created Pipe object.

4.1.3.1.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE if the Pipe is activated.

4.1.3.1.3 Remarks

A Pipe contains an Input Pipe Block (Master, PMP, or Sampler), a Converter Output Pipe Block, and any Transformation Pipe Block that can be in between.

- The "MLPipeAct" (→ p. 218) image shows two Pipes, both with the same Master Input Pipe Block.
  - The first ends with the first converter and has a Gear Pipe Block to transform the input values from the Master.
  - The second ends with the second converter and has a CAM Pipe Block to modify the input values from the Master.

- Once a Pipe is activated, history on the values in the Pipe's Blocks are saved and updated each program cycle.
  - A Converter object connected to a destination Axis object cannot send updated position values unless its Pipe is activated.

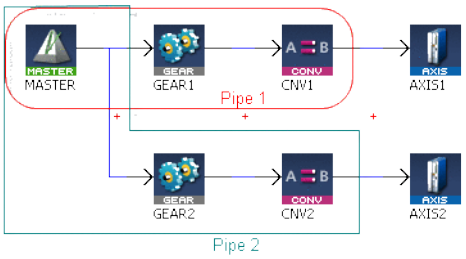
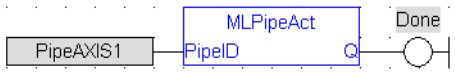


Figure 4-11: MLPipeAct

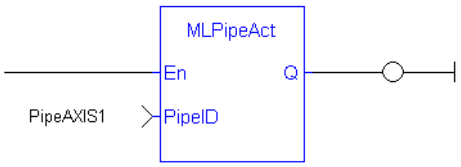
NOTE

All Pipes in the Pipe Network can be activated at once with the command **PipeNetwork(MLPN_ACTIVATE)**.
This calls automatically generated code with MLPipeAct commands for each Pipe object.
In a multi-pipe program, only one command can be used to activate Pipes instead of writing code for each Pipe separately.

4.1.3.1.4 FBD Language Example



4.1.3.1.5 FFLD Language Example



4.1.3.1.6 IL Language Example

Not available.

4.1.3.1.7 ST Language Example

```
//Activate a Pipe
MLPipeAct( PipeAXIS1 );
```

See Also

- "MLCNVConnect" (→ p. 303)
- "MLPipeAddBlock" (→ p. 219)
- "MLPipeDeact" (→ p. 221)
- Pipe Network Functions for the PLC - MLPN_ACTIVATE

4.1.3.2 MLPipeAddBlock

Pipe Network

Function - Adds a Pipe Block to a pipe.

4.1.3.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
PipeID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of a created Pipe object.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of a created Pipe object to add to the selected Pipe.

4.1.3.2.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE if the Pipe Block is added to the Pipe.

4.1.3.2.3 Remarks

A Pipe contains an Input Pipe Block (Master, PMP, or Sampler), a Converter Output Pipe Block, and any Transformation Pipe Block that can be in between.

- The "MLPipeAddBlock" (→ p. 220) image shows two Pipes, both with the same Master Input Pipe Block.
- If Pipe 1 is created without using the Graphical Engine, these commands are used once a Pipe and the Pipe Blocks have been created:
  - `MLPipeAddBlock( PipeAXIS1, MASTER);`
  - `MLPipeAddBlock( PipeAXIS1, MyGear);`
  - `MLPipeAddBlock( PipeAXIS1, CNV1);`

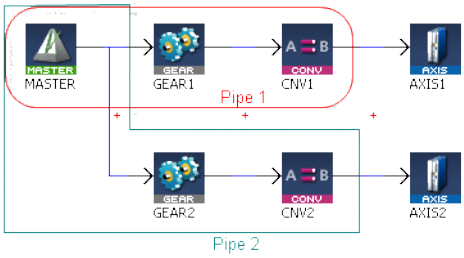
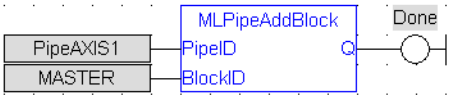


Figure 4-12: MLPipeAddBlock

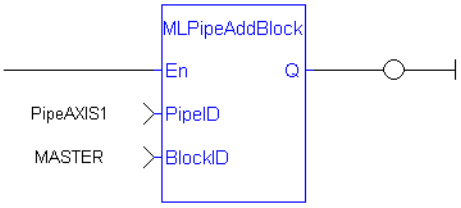
NOTE

All Blocks in the Pipe Network are added to a Pipe automatically.
Code with **MLPipeAddBlock** commands are automatically generated and called in a program with PipeNetwork(MLPN_CREATE_OBJECTS).
When using the Pipe Network graphical engine to create Pipe Blocks, the user does not have to manually add MLPipeAddBlock commands to the Project.

4.1.3.2.4 FBD Language Example



4.1.3.2.5 FFLD Language Example



4.1.3.2.6 IL Language Example

Not available.

4.1.3.2.7 ST Language Example

```
//Add a block to a pipe
MLPipeAddBlock( PipeAXIS1, MyGear );
```

See Also

- "MLCNVConnect" (→ p. 303)
- "MLPipeAct" (→ p. 217)
- "MLPipeCreate" (→ p. 220)
- "MLPipeDeact" (→ p. 221)
- Pipe Network Functions for the PLC - MLPN_CREATE_OBJECTS

4.1.3.3 MLPipeCreate

Pipe Network

Function - Create a new pipe object.

4.1.3.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Name	STRING	No range	N/A	No default	Name for the newly created Pipe.

4.1.3.3.2 Outputs

Output	Data Type	Range	Unit	Description
ID	DINT			Assigned ID number of the created Pipe.

4.1.3.3.3 Remarks

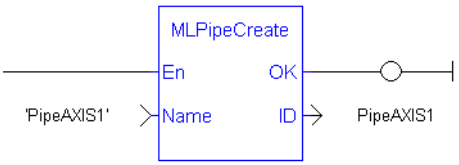
A Pipe contains an Input Pipe Block (Master, PMP, or Sampler), a Converter Output Pipe Block, and any Transformation Pipe Block that can be in between.

NOTE

Pipes are created in the Pipe Network using the graphical engine.
You do not have to add MLPipeCreate function blocks to their programs.
The code with **MLPipeCreate** commands are automatically generated and called in a program with PipeNetwork(MLPN_CREATE_OBJECTS).

4.1.3.3.4 FBD Language Example

4.1.3.3.5 FFLD Language Example



4.1.3.3.6 IL Language Example

Not available.

4.1.3.3.7 ST Language Example

```
//Create a new pipe
PipeAXIS1 := MLPipeCreate( 'PipeAXIS1' );
```

See Also

- "MLPipeAct" (→ p. 217)
- "MLPipeAddBlock" (→ p. 219)
- "MLPipeDeact" (→ p. 221)
- Pipe Network Functions for the PLC - MLPN_CREATE_OBJECTS

4.1.3.4 MLPipeDeact

Pipe Network

Function - Deactivates a pipe.

4.1.3.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
PipeID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of a created Pipe object.

4.1.3.4.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE if the Pipe is deactivated.

4.1.3.4.3 Remarks

A Pipe contains an Input Pipe Block (Master, PMP, or Sampler), a Converter Output Pipe Block, and any Transformation Pipe Block that can be in between.

- The "MLPipeDeact" (→ p. 222) image shows two Pipes, both with the same Master Input Pipe Block.
  - The first ends with the first converter and has a Gear Pipe Block to transform the input values from the Master.
  - The second ends with the second converter and has a CAM Pipe Block to modify the input values from the Master.
- Once a Pipe is activated, history on the values in the Pipe's Blocks are lost and no longer updated.
  - A Converter object connected to a destination Axis object cannot send updated position values once its Pipe is deactivated.

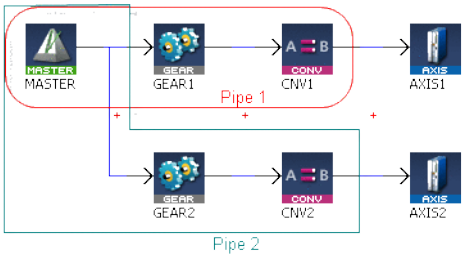


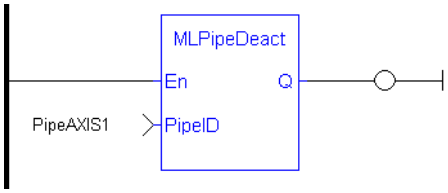
Figure 4-13: MLPipeDeact

NOTE

All Pipes in the Pipe Network can be deactivated at once with the command PipeNetwork(MLPN_DEACTIVATE).
This calls automatically generated code with **MLPipeDeact** commands for each Pipe object.
In a multi-pipe program, only one command can be used to deactivate Pipes instead of writing code for each Pipe separately.

4.1.3.4.4 FBD Language Example

4.1.3.4.5 FFLD Language Example



4.1.3.4.6 IL Language Example

Not available.

4.1.3.4.7 ST Language Example

```
//Deactivate a Pipe
MLPipeDeact( PipeAXIS1 );
```

See Also

- "MLCNVDisconnect" (→ p. 306)
- "MLPipeAct" (→ p. 217)
- "MLPipeAddBlock" (→ p. 219)
- "MLPipeCreate" (→ p. 220)
- Pipe Network Functions for the PLC - MLPN_DEACTIVATE

4.1.4 Axis

TIP

- See "Examples of Axis Functions" (→ p. 278) for more information.

4.1.4.1 Function by Types

Power Stage	Motion Control	Inquiry Functions	Position setting
"MLAxisPower" (→ p. 252)	"MLAxisAbs" (→ p. 226)	"MLAxisGenPos" (→ p. 239)	"MLAxisWritePos" (→ p. 274)
"MLAxisPowerDOff" (→ p. 253)	"MLAxisAdd" (→ p. 229)	"MLAxisPipePos" (→ p. 251)	"MLAxisReAlign" (→ p. 263)
	"MLAxisMoveVel" (→ p. 250)	"MLAxisCmdPos" (→ p. 234)	
	"MLAxisRel" (→ p. 264)	"MLAxisReadActPos" (→ p. 255)	
	"MLAxisStop" (→ p. 269)	"MLAxisFBackPos" (→ p. 236)	
		"MLAxisStatus" (→ p. 267)	
		"MLAxisReadGenStatus" (→ p. 257)	
		"MLAxisGenIsRdy" (→ p. 238)	
		"MLAxisTimeStamp" (→ p. 271)	
		"MLAxisDriveNumber" (→ p. 235)	

4.1.4.2 Functions in Alphabetical Order

Name	Description
"MLAxisAbs" (→ p. 226)	Performs a move to an absolute position.
"MLAxisAdd" (→ p. 229)	A selected Axis performs a move for a specified distance relative to the endpoint of the previous move.
"MLAxisAddress" (→ p. 231)	Returns the motion bus address of the axis.

Name	Description
"MLAxisAddTq" (→ p. 232)	Set the additive torque value to the drive output (torque feed-forward).
"MLAxisCfgFastIn" (→ p. 233)	Configures the Fast Input for the axis by writing the expected settings in the Latch Control Word.
"MLAxisCmdPos" (→ p. 234)	Returns the reference position of the axis.
"MLAxisDriveNumber" (→ p. 235)	Returns the drive number associated with the axis or -1 if the function block failed.
"MLAxisFBackPos" (→ p. 236)	Returns the feedback position of the axis.
"MLAxisGenEN" (→ p. 237)	Enables or disables the internal TMP generator of the axis.
"MLAxisGenIsEN" (→ p. 238)	Verifies if the internal TMP generator of the axis is enabled.
"MLAxisGenIsRdy" (→ p. 238)	Verifies if an axis is ready.
"MLAxisGenPos" (→ p. 239)	Returns the generator position of the axis.
"MLAxisGenReadAcc" (→ p. 240)	Gets the acceleration of the internal generator of an axis.
"MLAxisGenReadDec" (→ p. 241)	Gets the deceleration of the internal generator of an axis.
"MLAxisGenReadSpd" (→ p. 242)	Gets the speed of the internal generator of an axis.
"MLAxisGenWriteAcc" (→ p. 243)	Sets the acceleration of the internal generator of an axis.
"MLAxisGenWriteDec" (→ p. 244)	Sets the deceleration of the internal generator of an axis.
"MLAxisGenWriteSpd" (→ p. 245)	Sets the speed of the internal generator of an axis.
"MLAxisInit" (→ p. 246)	Initializes an axis object.
"MLAxisIsCnctd" (→ p. 248)	Verifies if a pipe is currently connected to the axis.
"MLAxisIsTrigged" (→ p. 249)	Verifies if the axis received a trigger event.
"MLAxisMoveVel" (→ p. 250)	Jog at the specified speed.
"MLAxisPipePos" (→ p. 251)	Returns the pipe position of the axis.
"MLAxisPower" (→ p. 252)	Powers up or down the axis. Enables or disables a Servo or Stepper drive mapped to the axis.
"MLAxisPowerDOff" (→ p. 253)	This function has been deprecated.

Name	Description
"MLAxisRatedTq" (→ p. 253)	Allows conversion of drive torque values from rated torque units (1000 = rated motor continuous torque) to N.m (Newton meter).
"MLAxisReadActPos" (→ p. 255)	Returns the actual position of the axis.
"MLAxisReadFBUnit" (→ p. 256)	Gets the feedback units per revolution value of the axis.
"MLAxisReadFEUU" (→ p. 257)	Returns the difference between the reference position and the actual position of the drive mapped to the specified axis.
"MLAxisReadGenStatus" (→ p. 257)	Returns the status of the internal generator of the axis.
"MLAxisReadModPos" (→ p. 259)	Gets the value period of the axis.
"MLAxisReadTq" (→ p. 259)	Returns the actual torque applied by the drive mapped to the specified axis.
"MLAxisReadUUnits" (→ p. 260)	Gets the User units per revolution value of the axis.
"MLAxisReadVel" (→ p. 261)	Returns the actual velocity of the axis based on the data provided by the drive's feedback device.
"MLAxisReAlgnRdy" (→ p. 262)	Verifies if an axis is ready.
"MLAxisReAlign" (→ p. 263)	Realigns the actual position with the reference position by moving the axis by the specified delta position.
"MLAxisRel" (→ p. 264)	A selected Axis performs a move for a specified distance relative to the current position.
"MLAxisResetErrors" (→ p. 266)	Clears errors of the specified axis.
"MLAxisRstFastIn" (→ p. 266)	Writes in the Latch Control Word to reset the Fast Input.
"MLAxisStatus" (→ p. 267)	Returns the status of the axis.
"MLAxisStop" (→ p. 269)	Stop with the specified deceleration. This does NOT remove the input source, but to stop the drive from continuing to move.
"MLAxisTimeStamp" (→ p. 271)	Returns the timestamp of the triggered axis.
"MLAxisWriteModPos" (→ p. 272)	Sets the value period of the axis.
"MLAxisWritePipPos" (→ p. 273)	Forces the pipe position internal value. **IMPORTANT** This function works only when no pipe is connected.
"MLAxisWritePos" (→ p. 274)	Sets a new value to an axis' current location.
"MLAxisWriteUUnits" (→ p. 276)	Set the user units per revolution value of the axis.

Name	Description
"MLPNAxisCreate" (→ p. 277)	Creates a new axis object.

4.1.4.3 MLAxisAbs

Function - Performs a move to an absolute position.

4.1.4.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	-2147483648 to 2147483647	N/A	No default	ID Name of the Axis block.
Position	LREAL	No range	User units	No default	Sets the value of the absolute destination position. • See "Position with Modulo On" (→ p. 226) for more information. • See Set the Axis Block Position Units for more information.

4.1.4.3.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

4.1.4.3.3 Remarks

- Returns TRUE if the function succeeded.
- See "Examples of Axis Functions" (→ p. 278) for more information.

4.1.4.3.3.1 Position with Modulo On

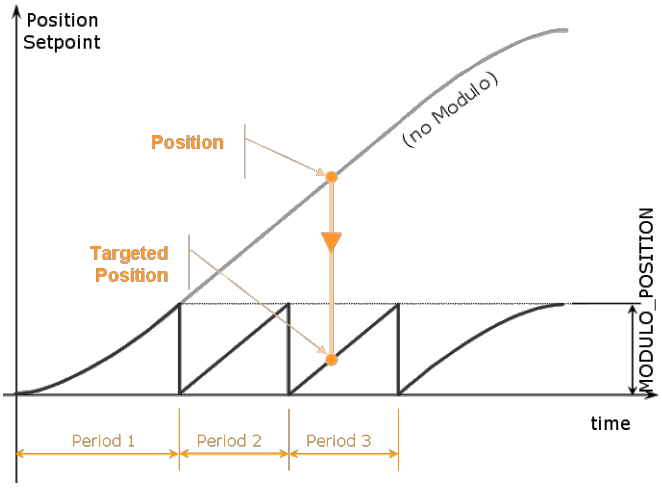
Position with Modulo On

NOTE

This information applies to both "MLMstAbs" (→ p. 332) and "MLAxisAbs" (→ p. 226).
For simplicity, the term Axis Block also refers to Master Block.

When the Modulo is turned on, the Axis Block moves to the targeted position during the corresponding period and is calculated as:

- If the Position input is between 0 (zero) and the Modulo position, the Axis Block moves within the **current** period (no position rollover).
- If the Position input is greater than the Modulo position, the Axis Block moves during one of the **next** period (positive position rollover).



The Axis Block works similarly for negative positions.

If the Position input is less than 0 (zero), the Axis Block moves during one of the **previous** period (negative position rollover).

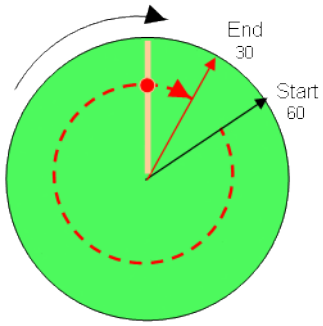
4.1.4.4 Forcing the Direction of Rotation

In some applications, the direction of rotation for the axis is forced in one direction only.

As a consequence, the motor movement goes to the next or previous modulo in these situations:

4.1.4.4.0.1 End Position is Less Than Start Position

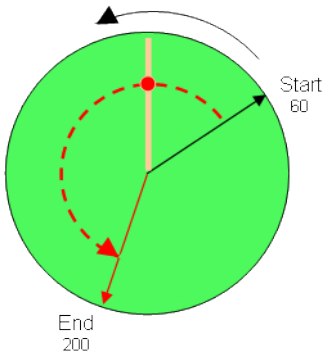
If the **End Position** is less than the **Start Position** and the direction of rotation for the axis is forced to be clockwise, the red point shows when the Modulo position is reached.



See Row 2 of the "MLAxisAbs" (→ p. 226).

4.1.4.4.0.2 End Position is Greater Than Start Position

If the **End Position** is greater than the **Start Position** and the direction of rotation for the axis is forced to be counter clockwise.



See Row 4 of the "MLAxisAbs" (→ p. 226).

Example Table

Start Position	End Position	Direction of Rotation	Cross Modulo	Position Input to MLAxisAbs (1)	RelativeDistance Moved (2)	
60	200	Clockwise	No	200	140	(i.e., 200 - 60 + 0)
60	30	Clockwise	Yes	390	330	(i.e., 30 - 60 + 360)
60	30	Counter clockwise	No	30	-30	(i.e., 30 - 60 - 0)
60	200	Counter clockwise	Yes	-160	-220	(i.e., 200 - 60 - 360)

With:

(1) **Position Input** = End Position (+ Modulo * Direction of rotation)

(2) **Relative Distance Moved** = End Position - Start Position (+ Modulo * Direction of rotation)

Where:

Direction of rotation = 1 when clockwise and -1 when anti-clockwise.

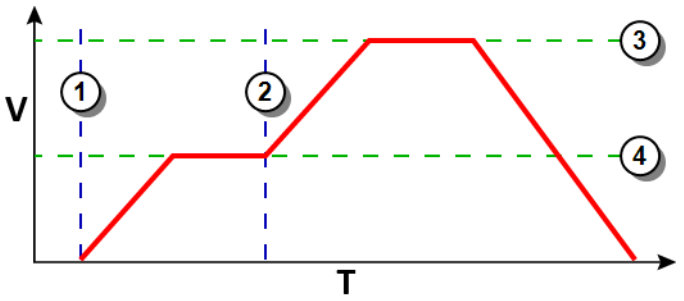
4.1.4.4.0.3 Travel Speed Update with MLAxisAbs

The travel speed of the generator can be updated using the function block "MLAxisGenWriteSpd" (→ p. 245).

Depending on the state of the generator, this speed is directly reflected on the current move or a future move.

- If **MLAxisAbs** is not currently being executed, the new travel speed is applied for the trajectory calculation for a future **MLAxisAbs** command.
- If **MLAxisAbs** is currently being executed, and a new MLAxisAbs with the **same** target position is called, the new travel speed is taken into account only if the current state of the TMP profile is the constant velocity or acceleration.
- If the axis was decelerating to stop at the goal position the new travel speed is not taken into account.
- If a **MLAxisAbs** is currently being executed, and a new MLAxisAbs with a **different** target position is called, the new travel speed is taken into account.

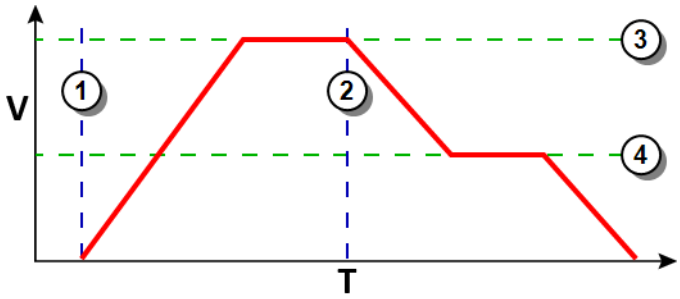
Example 1



1. First MLAxisAbs
2. Second MLAxisAbs
3. Initial Speed
4. New Speed

Figure 4-14: Initial speed is smaller than the new speed.

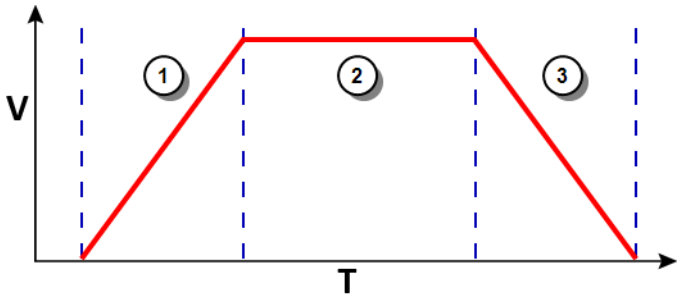
Example 2



1. First MLAxisAbs
2. Second MLAxisAbs
3. Initial Speed
4. New Speed

Figure 4-15: Initial speed is bigger than the new speed.

Example 3

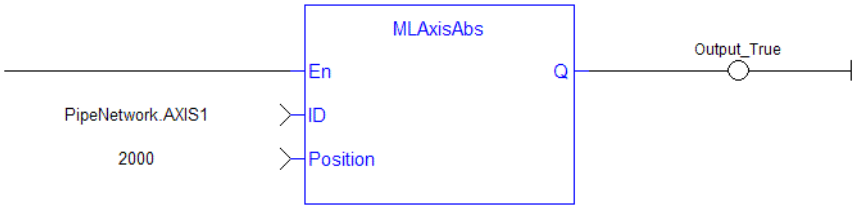


1. Acceleration
2. Constant Velocity
3. Deceleration

Figure 4-16: The speed update is taken into account only if the second MLAxisAbs is triggered during acceleration or constant velocity.

4.1.4.4.1 FBD Language Example

4.1.4.4.2 FFLD Language Example



4.1.4.4.3 IL Language Example

Not available.

4.1.4.4.4 ST Language Example

```
MLAxisAbs( PipeNetwork.Axis1, 2000 ) ;
```

See Also

- "MLAxisGenWriteAcc" (→ p. 243)
- "MLAxisGenWriteDec" (→ p. 244)
- "MLAxisGenWriteSpd" (→ p. 245)

4.1.4.5 MLAxisAdd

Pipe Network ✓

Function - A selected Axis performs a move for a specified distance relative to the endpoint of the previous move.

4.1.4.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.
DeltaPosition	LREAL	No range	User units	No default	Sets the Axis Delta Position to add to the endpoint of the previous move. See Set the Axis Block Position Units for more information.

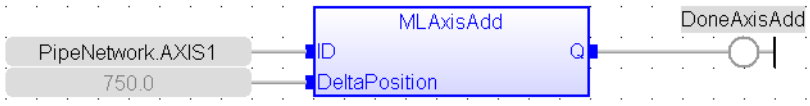
4.1.4.5.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE when the function successfully executes. • TRUE is after the motion profile is complete. •

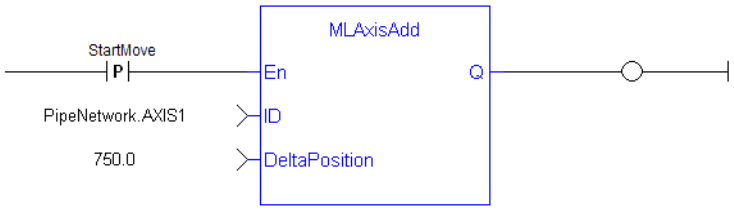
4.1.4.5.3 Remarks

- See "Examples of Axis Functions" (→ p. 278) for more information.
- The **DeltaPosition** input is signed so:
  - The move can be in the positive or negative direction.
  - The Axis moves this distance in User units.
- The travel speed, acceleration, deceleration, and User units of the move are values inherited from the selected Axis.
- The default settings are entered when an Axis is created and initiated, and can be changed with other MLAxis commands.

4.1.4.5.4 FBD Language Example



4.1.4.5.5 FFLD Language Example



NOTE

A pulse contact is required to start the FB.

4.1.4.5.6 IL Language Example

Not available.

4.1.4.5.7 ST Language Example

```
MLAxisAdd(PipeNetwork.Axis1, LREAL#750.0 ) ;
```

See Also

- "MLAxisGenWriteAcc" (→ p. 243)
- "MLAxisGenWriteDec" (→ p. 244)
- "MLAxisGenWriteSpd" (→ p. 245)
- "MLAxisWriteUUnits" (→ p. 276)

4.1.4.6 MLAxisAddress

Pipe Network

Function - Returns the motion bus address of the axis.

4.1.4.6.1 Inputs

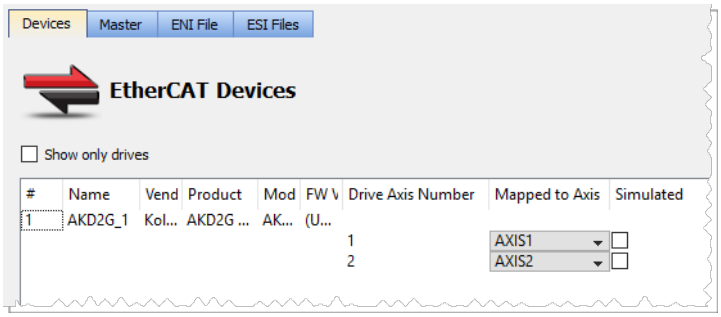
Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.

4.1.4.6.2 Outputs

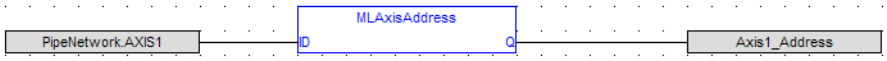
Output	Data Type	Range	Unit	Description
OK	BOOL			Returns TRUE when the function successfully executes.
Default (.Q)	DINT			Returns the motion bus address of the axis.

4.1.4.6.3 Remarks

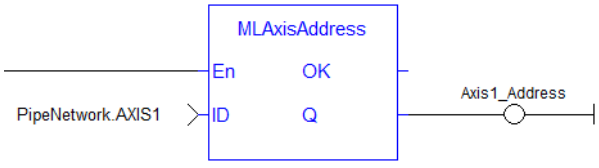
- Axes have the same address when they are mapped to the same multi-axis drive (e.g., the dual-axis AKD2G drive).
  - Example: If Axis1 is mapped to an AKD2G's Drive Axis Number 1 and Axis2 is mapped to the AKD2G's Drive Axis Number 2, both axes return the same address.



4.1.4.6.4 FBD Language Example



4.1.4.6.5 FFLD Language Example



4.1.4.6.6 IL Language Example

Not available.

4.1.4.6.7 ST Language Example

```
Axis1_Address := MLAxisAddress(PipeNetwork.AXIS1);
```

4.1.4.7 MLAxisAddTq

Pipe Network

Function - Set the additive torque value to the drive output (torque feed-forward).

4.1.4.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	Pipe network identifier of the axis block.
Torque	LREAL	No range	Rated torque units as used in the drive. Example: Rated motor continuous torque X the Torque factor.	No default	Requested additive torque value in N.m (Newton meter).

4.1.4.7.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE when the function successfully executes.

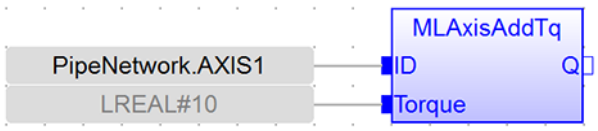
4.1.4.7.3 Remarks

NOTE

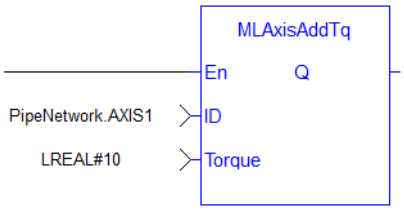
This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- This function is only active after the "MLAxisRatedTq" (→ p. 253) function has been invoked.
- Using the PDO, it also requires the IL.KBUSFF value to be set to 1 in the drive.

4.1.4.7.4 FBD Language Example



4.1.4.7.5 FFLD Language Example



4.1.4.7.6 IL Language Example

Not available.

4.1.4.7.7 ST Language Example

```
MLAxisAddTq(PipeNetwork.Axis1, LREAL#10 )  ;
```

See Also

"MLAxisRatedTq" (→ p. 253)

4.1.4.8 MLAxisCfgFastIn

Pipe Network ✓

Function - Configures the Fast Input for the axis by writing the expected settings in the Latch Control Word.

4.1.4.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	No range	N/A	No default	Enables execution.
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block.
InputID	DINT	0, 1	N/A	No default	ID of the FastInput of an axis (e.g., IN1 and IN2 on S300). **InputID** INT Range = 0, 1 • 0 = Touch Probe 1 / Capture Engine 0 • 1 = Touch Probe 2 / Capture Engine 1
Mode	DINT	0, 2	N/A	No default	Configures the Fast Inputs as: • 0= Disabled • 1=Rising edge • 2=Falling edge

4.1.4.8.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL	No range	N/A	• Returns TRUE when the function successfully executes. • Returns FALSE if the fast input could not be configured because of an invalid PDO mapping in the .XML file.

4.1.4.8.3 Remarks

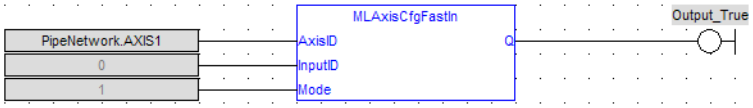
Fast Input can be armed on falling or rising edge.

4.1.4.8.3.1 Fast Homing - Inputs

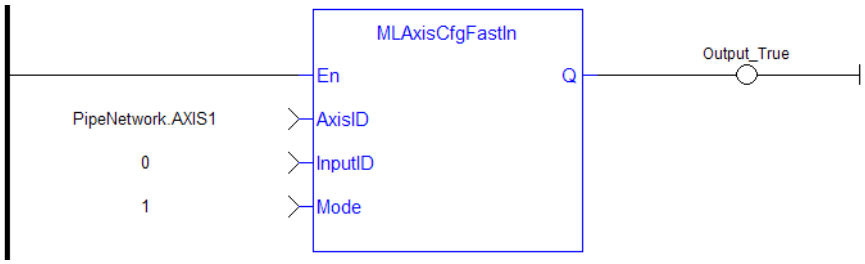
See these topics for more information:

- Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block
- Fast Homing Example with the PLCopen Motion Engine
- Fast Inputs with Pipe Network Motion
- Pipe Network Registration and Fast Homing
- Registration Position Capture Example with Pipe Network Trigger Block

4.1.4.8.4 FBD Language Example



4.1.4.8.5 FFLD Language Example



4.1.4.8.6 IL Language Example

Not available.

4.1.4.8.7 ST Language Example

```
MLAxisCfgFastIn( PipeNetwork.Axis1, 0, 1 ) ;
```

See Also

- "MLAxisIsTrigged" (→ p. 249)
- "MLAxisRstFastIn" (→ p. 266)

4.1.4.9 MLAxisCmdPos

Pipe Network

Function - Returns the reference position of the axis.

4.1.4.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.

4.1.4.9.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Returns TRUE when the function successfully executes.
Position	LREAL			Returns the Axis reference position. See Set the Axis Block Position Units for more information.

4.1.4.9.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- Previous function name: MLAxisRefPos

4.1.4.9.4 FBD Language Example

4.1.4.9.5 FFLD Language Example

4.1.4.9.6 IL Language Example

Not available.

4.1.4.9.7 ST Language Example

```
Axis1_ReferencePosition := MLAxisCmdPos(PipeNetwork.AXIS1);
```

See Also

- "MLAxisFBackPos" (→ p. 236)
- "MLAxisGenPos" (→ p. 239)
- "MLAxisPipePos" (→ p. 251)
- "MLAxisReadActPos" (→ p. 255)
- "MLAxisWritePipPos" (→ p. 273)

4.1.4.10 MLAxisDriveNumber

Pipe Network

Function Block - Returns the drive number associated with the axis or -1 if the function block failed.

4.1.4.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block.

4.1.4.10.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Returns TRUE when the function successfully executes.
DriveAxisNumber	INT	-1 or 1,32767	N/A	Drive number associated with the axis, or -1 if the function block failed.

4.1.4.10.3 Remarks

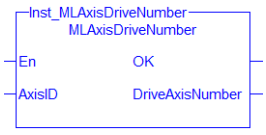
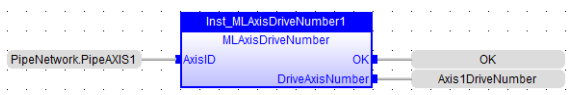


Figure 4-17: ML_AxisDriveNumber

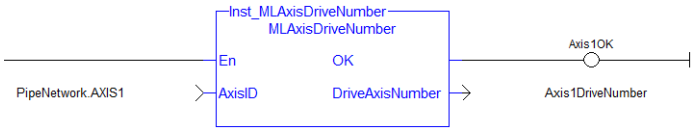
TIP

"MLPNAxisCreate" (→ p. 277) assigns the drive axis number.

4.1.4.10.4 FBD Language Example



4.1.4.10.5 FFLD Language Example



4.1.4.10.6 IL Language Example

Not available.

4.1.4.10.7 ST Language Example

```
Inst_MLAxisDriveNumber( AxisID)
IF Inst_MLAxisDriveNumber.OK Then
     Axis1DriveNumber     := Inst_MLAxisDriveNumber.DriveAxisNumber
End_IF;
```

4.1.4.11 MLAxisFBackPos

Pipe Network

Function - Returns the feedback position of the axis.

4.1.4.11.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.

4.1.4.11.2 Outputs

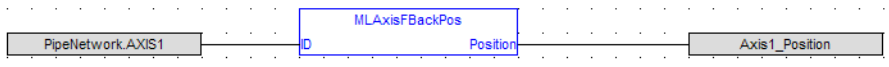
Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Returns TRUE when the function successfully executes.
Position	LREAL	No range	User units	Returns the feedback position of the axis. See Set the Axis Block Position Units for more information.

4.1.4.11.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

4.1.4.11.4 FBD Language Example



4.1.4.11.5 FFLD Language Example

4.1.4.11.6 IL Language Example

Not available.

4.1.4.11.7 ST Language Example

```
Axis1_Position := MLAxisFBackPos( PipeNetwork.Axis1 ) ;
```

See Also

- "MLAxisCmdPos" (→ p. 234)
- "MLAxisGenPos" (→ p. 239)
- "MLAxisPipePos" (→ p. 251)
- "MLAxisReadActPos" (→ p. 255)
- "MLAxisWritePipPos" (→ p. 273)

4.1.4.12 MLAxisGenEN

Pipe Network ✓

Function - Enables or disables the internal TMP generator of the axis.

4.1.4.12.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.
Enable	BOOL	0, 1	N/A	No default	Boolean switch to activate the generator.

4.1.4.12.2 Outputs

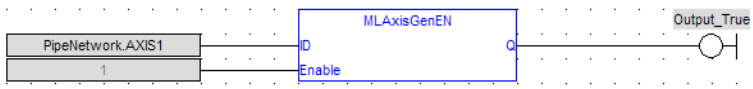
Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE when the function successfully executes.

4.1.4.12.3 Remarks

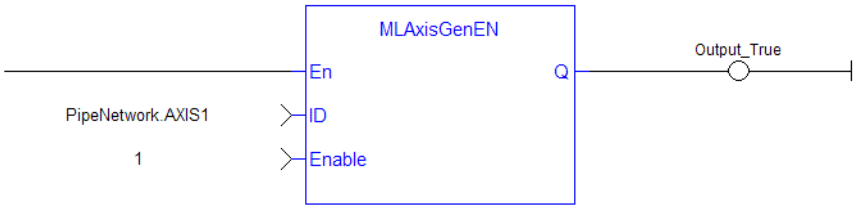
NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

4.1.4.12.4 FBD Language Example



4.1.4.12.5 FFLD Language Example



4.1.4.12.6 IL Language Example

Not available.

4.1.4.12.7 ST Language Example

```
MLAxisGenEN( PipeNetwork.Axis1, true) ;
```

See Also

"MLAxisGenIsEN" (→ p. 238)

4.1.4.13 MLAxisGenIsEN

Pipe Network

Function - Verifies if the internal TMP generator of the axis is enabled.

4.1.4.13.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.

4.1.4.13.2 Outputs

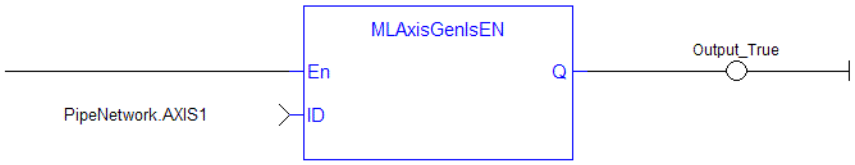
Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE when the function successfully executes.

4.1.4.13.3 Remarks

- Returns TRUE if the internal generator is enabled.

4.1.4.13.4 FBD Language Example

4.1.4.13.5 FFLD Language Example



4.1.4.13.6 IL Language Example

Not available.

4.1.4.13.7 ST Language Example

```
MLAxisGenIsEN(PipeNetwork.Axis1 ) ;
```

See Also

"MLAxisGenIsRdy" (→ p. 238)

4.1.4.14 MLAxisGenIsRdy

Pipe Network

Function - Verifies if an axis is ready.

4.1.4.14.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.

4.1.4.14.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE when the function successfully executes.

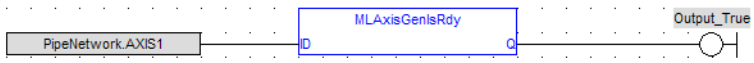
4.1.4.14.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- Returns TRUE if the internal generator axis is ready.
- See "Examples of Axis Functions" (→ p. 278) for more information.

4.1.4.14.4 FBD Language Example



4.1.4.14.5 FFLD Language Example

4.1.4.14.6 IL Language Example

Not available.

4.1.4.14.7 ST Language Example

```
MLAxisGenIsRdy(PipeNetwork.Axis1 );
```

See Also

- "MLAxisGenIsEN" (→ p. 238)
- "MLAxisStatus" (→ p. 267)

4.1.4.15 MLAxisGenPos

Pipe Network

Function - Returns the generator position of the axis.

4.1.4.15.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.

4.1.4.15.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE when the function successfully executes.
Position	LREAL			Returns the Axis generator position value. See Set the Axis Block Position Units for more information.

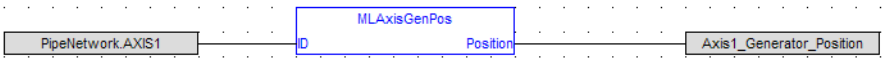
4.1.4.15.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- Returns TRUE if the internal generator axis is ready.

4.1.4.15.4 FBD Language Example



4.1.4.15.5 FFLD Language Example

4.1.4.15.6 IL Language Example

Not available.

4.1.4.15.7 ST Language Example

```
Axis1_Generator_Position := MLAxisGenPos(PipeNetwork.Axis1 ) ;
```

See Also

MLAxisReadActPos
MLAxisFBackPos
MLAxisPipePos
MLAxisCmdPos
MLAxisWritePipPos

4.1.4.16 MLAxisGenReadAcc

Pipe Network

Function - Gets the acceleration of the internal generator of an axis.

4.1.4.16.1 Inputs

Input	Data Type	Range	Unit	Default	Description
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block.

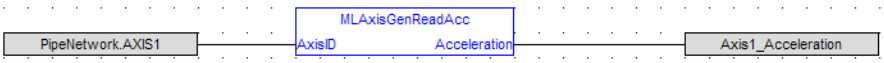
4.1.4.16.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Returns TRUE when the function successfully executes.
Acceleration	LREAL	No range	User unit/sec^2	Returns the Axis Acceleration value. See Set the Axis Block Position Units for more information.

4.1.4.16.3 Remarks

None

4.1.4.16.4 FBD Language Example



4.1.4.16.5 FFLD Language Example

4.1.4.16.6 IL Language Example

Not available.

4.1.4.16.7 ST Language Example

```
Axis1_Acceleration := MLAxisGenReadAcc( PipeNetwork.Axis1 );
```

See Also

- "MLAxisGenReadDec" (→ p. 241)
- "MLAxisGenReadSpd" (→ p. 242)

4.1.4.17 MLAxisGenReadDec

Pipe Network

Function - Gets the deceleration of the internal generator of an axis.

4.1.4.17.1 Inputs

Input	Data Type	Range	Unit	Default	Description
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block.

4.1.4.17.2 Outputs

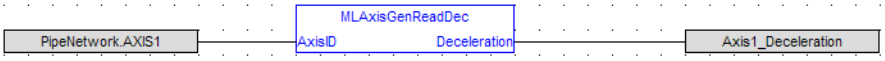
Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Returns TRUE when the function successfully executes.

Output	Data Type	Range	Unit	Description
Deceleration	LREAL	No range	User unit/sec^2	Returns the Axis Deceleration value. See Set the Axis Block Position Units for more information.

4.1.4.17.3 Remarks

None

4.1.4.17.4 FBD Language Example



4.1.4.17.5 FFLD Language Example

4.1.4.17.6 IL Language Example

Not available.

4.1.4.17.7 ST Language Example

```
Axis1_Deceleration := MLAxisGenReadDec( PipeNetwork.Axis1 );
```

See Also

- "MLAxisGenReadAcc" (→ p. 240)
- "MLAxisGenReadSpd" (→ p. 242)

4.1.4.18 MLAxisGenReadSpd

Pipe Network

Function - Gets the speed of the internal generator of an axis.

4.1.4.18.1 Inputs

Input	Data Type	Range	Unit	Default	Description
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block.

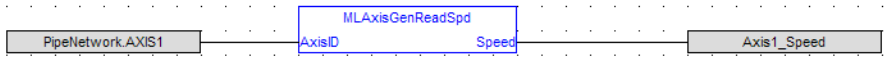
4.1.4.18.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Returns TRUE when the function successfully executes.
Speed	LREAL	No range	User unit/sec	Returns the Axis Speed value. See Set the Axis Block Position Units for more information.

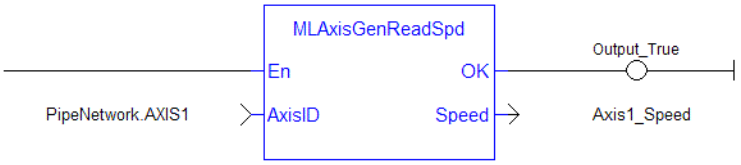
4.1.4.18.3 Remarks

None

4.1.4.18.4 FBD Language Example



4.1.4.18.5 FFLD Language Example



4.1.4.18.6 IL Language Example

Not available.

4.1.4.18.7 ST Language Example

```
Axis1_Speed := MLAxisGenReadSpd( PipeNetwork.Axis1 ) ;
```

See Also

- "MLAxisGenReadAcc" (→ p. 240)
- "MLAxisGenReadDec" (→ p. 241)

4.1.4.19 MLAxisGenWriteAcc

Pipe Network

Function - Sets the acceleration of the internal generator of an axis.

4.1.4.19.1 Inputs

Input	Data Type	Range	Unit	Default	Description
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block.
Acceleration	LREAL	No range	User unit/sec^2	No default	Sets the generator Acceleration value. See Set the Axis Block Position Units for more information.

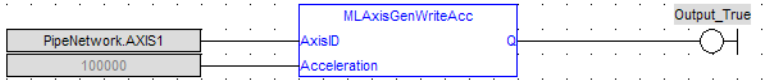
4.1.4.19.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE when the function successfully executes.

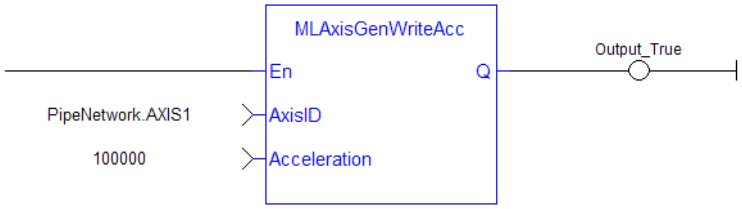
4.1.4.19.3 Remarks

- Returns TRUE if the internal generator axis is ready.

4.1.4.19.4 FBD Language Example



4.1.4.19.5 FFLD Language Example



4.1.4.19.6 IL Language Example

Not available.

4.1.4.19.7 ST Language Example

```
MLAxisGenWriteAcc(PipeNetwork.Axis1, 100000 )  ;
```

See Also

- "MLAxisGenWriteDec" (→ p. 244)
- "MLAxisGenWriteSpd" (→ p. 245)

4.1.4.20 MLAxisGenWriteDec

Pipe Network

Function - Sets the deceleration of the internal generator of an axis.

4.1.4.20.1 Inputs

Input	Data Type	Range	Unit	Default	Description
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block.
Deceleration	LREAL	No range	User unit/sec^2	No default	Sets the generator Deceleration value. • The axis deceleration rate is limited so the velocity cannot change by more than the value of the declared velocity limit in a single iteration. • The Pipe Network Axis block uses the **TRAVEL_SPEED** parameter to scale this limit. • The maximum deceleration is affected by the Pipe Network Axis Block parameter **TRAVEL_SPEED** and the axis update rate. • See Set the Axis Block Position Units for more information.

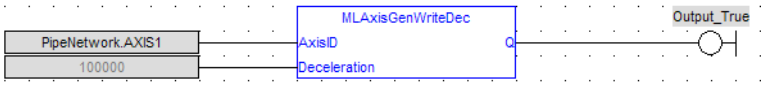
4.1.4.20.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE when the function successfully executes.

4.1.4.20.3 Remarks

- Returns TRUE if the internal generator axis is ready.

4.1.4.20.4 FBD Language Example



4.1.4.20.5 FFLD Language Example

4.1.4.20.6 IL Language Example

Not available.

4.1.4.20.7 ST Language Example

```
MLAxisGenWriteDec(PipeNetwork.Axis1, 100000 )  ;
```

See Also

- "MLAxisGenWriteAcc" (→ p. 243)
- "MLAxisGenWriteSpd" (→ p. 245)

4.1.4.21 MLAxisGenWriteSpd

Pipe Network

Function - Sets the speed of the internal generator of an axis.

4.1.4.21.1 Inputs

Input	Data Type	Range	Unit	Default	Description
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block.
Speed	LREAL	No range	User units	No default	Sets the generator Speed value. See Set the Axis Block Position Units for more information.

4.1.4.21.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE when the function successfully executes.

4.1.4.21.3 Remarks

- This function or function block does not generate any motion.

4.1.4.21.4 FBD Language Example

4.1.4.21.5 FFLD Language Example

4.1.4.21.6 IL Language Example

Not available.

4.1.4.21.7 ST Language Example

```
MLAxisGenWriteSpd(PipeNetwork.Axis1, 500 )  ;
```

See Also

- "MLAxisGenWriteAcc" (→ p. 243)
- "MLAxisGenWriteDec" (→ p. 244)

4.1.4.22 MLAxisInit

Pipe Network

Function - Initializes an axis object.

4.1.4.22.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Acceleration	LREAL	No range	User unit/sec^2	No default	Sets the Axis Acceleration value. See Set the Axis Block Position Units for more information.
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block.
Deceleration	LREAL	No range	User unit/sec^2	No default	Sets the Axis Deceleration value. See Set the Axis Block Position Units for more information.
FeedbackUnitPerTurn	DINT	No range	User units	No default	

Input	Data Type	Range	Unit	Default	Description
InitialPosition	LREAL	No range	User units	No default	Initial position value expressed in user logical units. • Used only at the pipe activation to initialize the position starting point. • See Set the Axis Block Position Units for more information.
Modulo	BOOL	0, 1	N/A	No default	Define the mode which can be Modulo (True) or not (False).
ModuloPosition	LREAL	No range	User units	No default	The value of the period of a cyclic system expressed in User units. The parameter is defined to correctly manage the periodicity (modulo) of the input values.
Speed	LREAL	No range	User units	No default	Sets the Axis Speed. See Set the Axis Block Position Units for more information.
UserUnitPerTurn	LREAL	No range	User units	No default	Define the unit equivalent to one revolution of the physical motor.

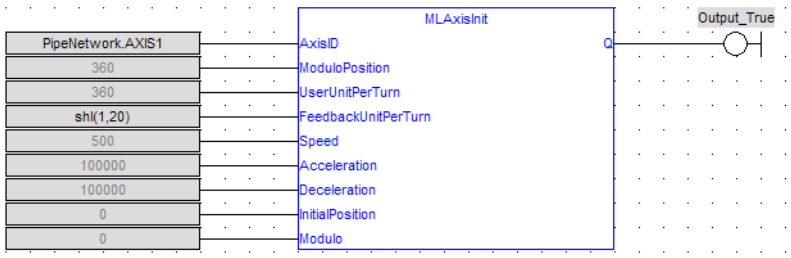
4.1.4.22.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE when the function successfully executes.

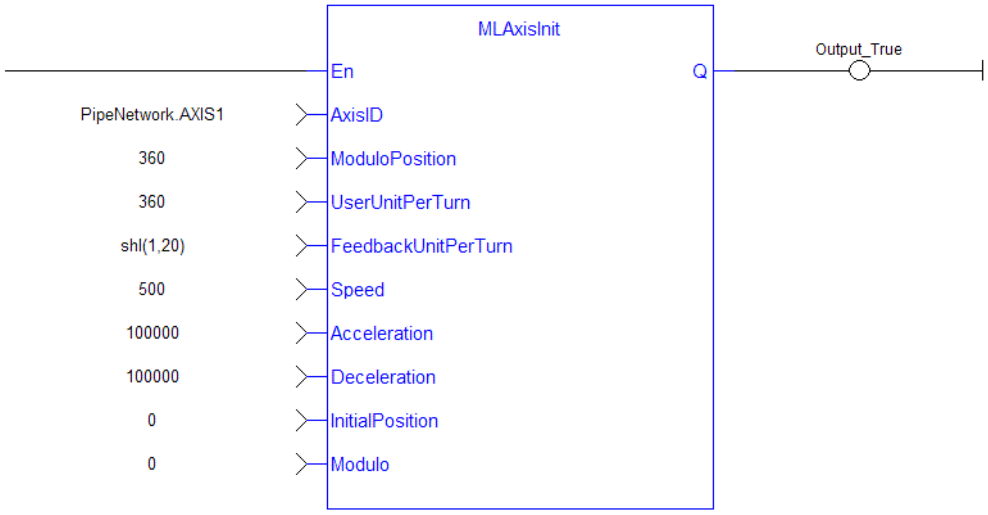
4.1.4.22.3 Remarks

- Returns TRUE if the function succeeded.
- The axis object can be mapped to servo or stepper drives.

4.1.4.22.4 FBD Language Example



4.1.4.22.5 FFLD Language Example



4.1.4.22.6 IL Language Example

Not available.

4.1.4.22.7 ST Language Example

```
MLAxisInit( PipeNetwork.Axis1, 360.0, 360.0, SHL(1,20), 500.0, 100000.0,
100000.0, 0.0, true ) ;
```

4.1.4.23 MLAxisIsCnctd

Pipe Network

Function - Verifies if a pipe is currently connected to the axis.

4.1.4.23.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.

4.1.4.23.2 Outputs

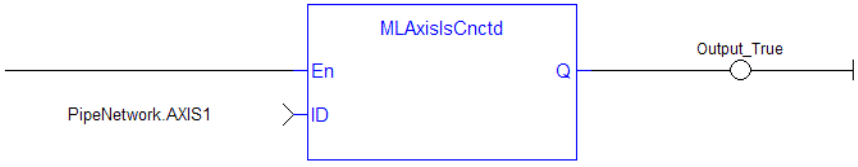
Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE when the function successfully executes.

4.1.4.23.3 Remarks

- Returns TRUE if a pipe is connected.

4.1.4.23.4 FBD Language Example

4.1.4.23.5 FFLD Language Example



4.1.4.23.6 IL Language Example

Not available.

4.1.4.23.7 ST Language Example

```
MLAxisIsCnctd(PipeNetwork.Axis1 )  ;
```

4.1.4.24 MLAxisIsTrigged

Pipe Network ✓

Function - Verifies if the axis received a trigger event.

4.1.4.24.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.
InputID	DINT	No range	N/A	No default	ID of the triggered Fast input of an axis (i.e., IN1 and IN2 on S300). **InputID** INT Range = 0, 1 • 0 = Touch Probe 1 / Capture Engine 0 • 1 = Touch Probe 2 / Capture Engine 1
edge	DINT	No range	N/A	No default	Configures the Inputs as: • 0 = Disabled. • 1 = Rising edge. • 2 = Falling edge.

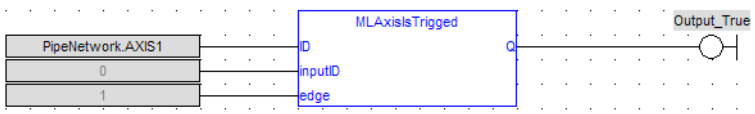
4.1.4.24.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

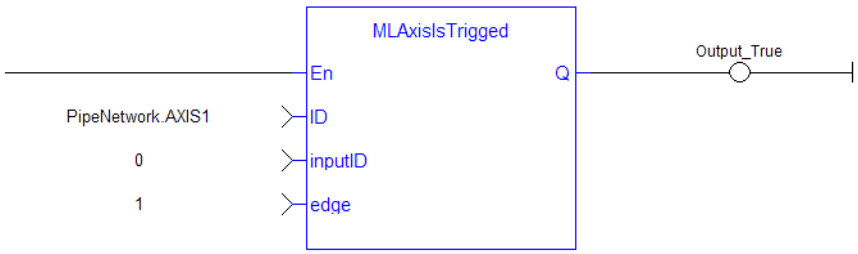
4.1.4.24.3 Remarks

- Returns TRUE if the Fast Input event has been **triggered** and not yet been reset.

4.1.4.24.4 FBD Language Example



4.1.4.24.5 FFLD Language Example



4.1.4.24.6 IL Language Example

Not available.

4.1.4.24.7 ST Language Example

```
MLAxisIsTrigged (PipeNetwork.Axis1, 0,1 ) ;
```

See Also

"MLAxisRstFastIn" (→ p. 266)

4.1.4.25 MLAxisMoveVel

Pipe Network ✓

Function - Jog at the specified speed.

4.1.4.25.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.
Speed	LREAL	No range	User unit/sec	No default	Sets the axis speed. See Set the Axis Block Position Units for more information.

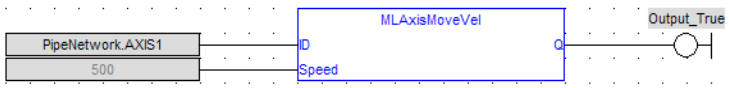
4.1.4.25.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE when the function successfully executes. (after the motion has reached jog speed)

4.1.4.25.3 Remarks

- Previous function name: MLAxisRun
- Returns TRUE if the function succeeded.
- See "Examples of Axis Functions" (→ p. 278) for more information.

4.1.4.25.4 FBD Language Example



4.1.4.25.5 FFLD Language Example

4.1.4.25.6 IL Language Example

Not available.

4.1.4.25.7 ST Language Example

```
MLAxisMoveVel(PipeNetwork.Axis1, 500 )  ;
```

See Also

- "MLAxisGenWriteAcc" (→ p. 243)
- "MLAxisGenWriteDec" (→ p. 244)
- "MLAxisGenWriteSpd" (→ p. 245)

4.1.4.26 MLAxisPipePos

Pipe Network ✓

Function - Returns the pipe position of the axis.

4.1.4.26.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	-2147483648 to 2147483647	N/A	No default	ID Name of the Axis block.
Speed	LREAL	No range	User unit/sec	No default	Sets the axis speed. See Set the Axis Block Position Units for more information.

4.1.4.26.2 Outputs

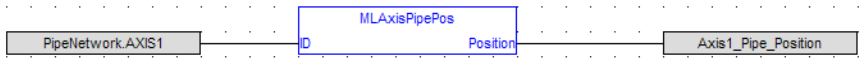
Output	Data Type	Range	Unit	Description
OK	BOOL			Returns TRUE when the function successfully executes.
Position	LREAL			See Set the Axis Block Position Units for more information.

4.1.4.26.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

4.1.4.26.4 FBD Language Example



4.1.4.26.5 FFLD Language Example

4.1.4.26.6 IL Language Example

Not available.

4.1.4.26.7 ST Language Example

```
Axis1_Pipe_Position := MLAxisPipePos (PipeNetwork.Axis1 ) ;
```

See Also

- "MLAxisCmdPos" (→ p. 234)
- "MLAxisFBackPos" (→ p. 236)
- "MLAxisGenPos" (→ p. 239)
- "MLAxisReadActPos" (→ p. 255)
- "MLAxisWritePipPos" (→ p. 273)

4.1.4.27 MLAxisPower

Pipe Network ✓

Function - Powers up or down the axis.
Enables or disables a Servo or Stepper drive mapped to the axis.

4.1.4.27.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.
On	BOOL	0, 1	N/A	No default	Flag to: • Power up (TRUE) the axis. • Power down (FALSE) the axis.

4.1.4.27.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

4.1.4.27.3 Remarks

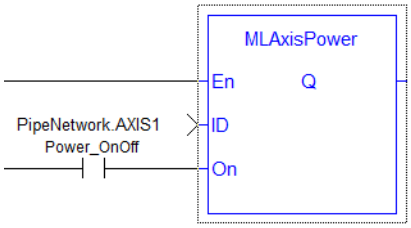
IMPORTANT

Powering on an axis affects the position and motion state of an axis.
Do now call **MLAxisPower** with the **On** input flag set to **True** while the axis is in motion.

- Previous Function Names:
  - MLAxisPowerOn
  - MLAxisPowerOff
- When the axis is powered up:
  - The **ReferencePosition** is modified to equal the **ActualPosition**.
  - KAS updates the **GeneratorPosition**.

4.1.4.27.4 FBD Language Example

4.1.4.27.5 FFLD Language Example



4.1.4.27.6 IL Language Example

Not available.

4.1.4.27.7 ST Language Example

```
(* If Power_OnOff is TRUE then power in ON, otherwise OFF*)
MLAxisPower( PipeNetwork.Axis1, Power_OnOff) ;
```

See Also

"MLAxisPowerDOff" (→ p. 253)

4.1.4.28 MLAxisPowerDOff

Pipe Network ✓

Function - This function has been deprecated.

4.1.4.29 MLAxisRatedTq

Pipe Network ✓

Function - Allows conversion of drive torque values from rated torque units (1000 = rated motor continuous torque) to N.m (Newton meter).

4.1.4.29.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	Pipe network identifier of the axis block.

Input	Data Type	Range	Unit	Default	Description
Torque	LREAL	No range	N.m (Newton meter)	No default	Actual torque applied by the drive associated to the axis. Rated torque = Nominal Drive Current * Torque factor = DRV.ICONT * MOTOR.KT. **About SDO** DRV.ICONT is obtained by SDO parameter: index 5083h (sub-index 0). MOTOR.KT is obtained by SDO parameter: index 3593h (sub-index 0). For more information, see: • Communication SDOs • Manufacturer specific SDOs • Profile specific SDOs To read/write an SDO object with an index greater than 16#7FFF (32767), the value must be entered in this form: `any_to_int(index # in hex format)`. Example: `any_to_int(16#8321)`. • The actual units of **DRV.ICONT** and **MOTOR.KT** are 1/1000 of the actual values if obtained by SDO. • The formula, if using the SDO values, is: • Rated Torque = Torque = (SDO (DRV.ICONT)/1000) * (SDO (MOTOR.KT)/1000).

4.1.4.29.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

4.1.4.29.3 Remarks

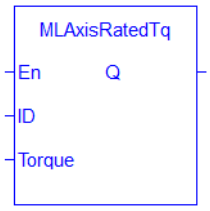
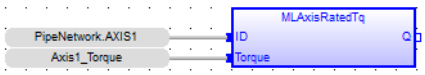
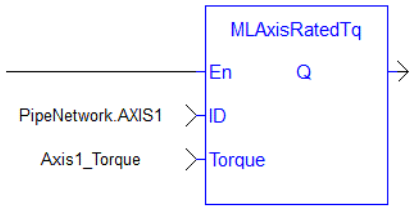


Figure 4-18: MLAxisRatedTq

4.1.4.29.4 FBD Language Example



4.1.4.29.5 FFLD Language Example



4.1.4.29.6 IL Language Example

Not available.

4.1.4.29.7 ST Language Example

```
MLAxisRatedTq(PipeNetwork.Axis1, Axis1_Torque )  ;
```

See Also

"MLAxisReadTq" (→ p. 259)

4.1.4.30 MLAxisReadActPos

Pipe Network

Function - Returns the actual position of the axis.

4.1.4.30.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.

4.1.4.30.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.
Position	LREAL		User units	Returns the absolute position of the axis. See Set the Axis Block Position Units for more information.

4.1.4.30.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- Previous function name: MLAxisActualPos

4.1.4.30.4 FBD Language Example

4.1.4.30.5 FFLD Language Example

4.1.4.30.6 IL Language Example

Not available.

4.1.4.30.7 ST Language Example

```
Axis1_Position := MLAxisReadActPos( PipeNetwork.Axis1 ) ;
```

See Also

- "MLAxisCmdPos" (→ p. 234)
- "MLAxisFBackPos" (→ p. 236)
- "MLAxisGenPos" (→ p. 239)
- "MLAxisPipePos" (→ p. 251)
- "MLAxisWritePipPos" (→ p. 273)

4.1.4.31 MLAxisReadFBUnit

Pipe Network

Function - Gets the feedback units per revolution value of the axis.

4.1.4.31.1 Inputs

Input	Data Type	Range	Unit	Default	Description
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block.

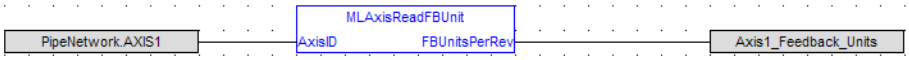
4.1.4.31.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL			Returns TRUE when the function successfully executes.
FBUnitsPerRev	LREAL			Returns the Axis Feedback Units per revolution.

4.1.4.31.3 Remarks

None

4.1.4.31.4 FBD Language Example



4.1.4.31.5 FFLD Language Example

4.1.4.31.6 IL Language Example

Not available.

4.1.4.31.7 ST Language Example

```
Axis1_Feedback_Units := MLAxisReadFBUnit(PipeNetwork.Axis1 )  ;
```

4.1.4.32 MLAxisReadFEUU

Pipe Network

Function - Returns the difference between the reference position and the actual position of the drive mapped to the specified axis.

4.1.4.32.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	Pipe network identifier of the axis block.

4.1.4.32.2 Outputs

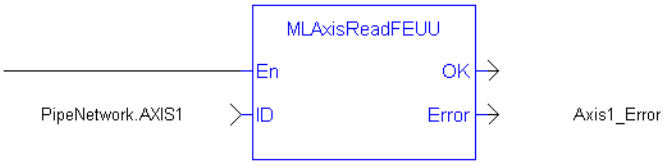
Output	Data Type	Range	Unit	Description
Error	LREAL		User units	Returns the difference between the reference position and the actual position of the drive mapped to the specified axis.

4.1.4.32.3 Remarks

None

4.1.4.32.4 FBD Language Example

4.1.4.32.5 FFLD Language Example



4.1.4.32.6 IL Language Example

Not available.

4.1.4.32.7 ST Language Example

```
Axis1_Error := MLAxisReadFEUU(PipeNetwork.Axis1 )  ;
```

See Also

"MLAxisReadActPos" (→ p. 255)

4.1.4.33 MLAxisReadGenStatus

Pipe Network

Function - Returns the status of the internal generator of the axis.

4.1.4.33.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.

4.1.4.33.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL			Returns TRUE when the function successfully executes.
Default (.Q)	DINT			Shows the status of the internal generator based on the "Internal Generator Status" (→ p. 258) table.

4.1.4.33.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- Previous function name: MLAxisGenStatus

4.1.4.33.3.1 Internal Generator Status

0	RUN mode (acceleration).
1	RUNNING or STOPPED.
2	MOVE: Changing move destination.
3	MOVE: Changing move destination.
4	MOVE: Acceleration.
5	MOVE: Constant speed (travel speed).
6	MOVE: Deceleration.
7	MOVE: Single step (micro movement).

4.1.4.33.4 FBD Language Example

4.1.4.33.5 FFLD Language Example

4.1.4.33.6 IL Language Example

Not available.

4.1.4.33.7 ST Language Example

```
MLAxisReadGenStatus(PipeNetwork.Axis1 )  ;
```

See Also

- "MLAxisGenIsRdy" (→ p. 238)
- "MLAxisStatus" (→ p. 267)

4.1.4.34 MLAxisReadModPos

Pipe Network

Function - Gets the value period of the axis.

4.1.4.34.1 Inputs

Input	Data Type	Range	Unit	Default	Description
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block.

4.1.4.34.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL			Returns TRUE when the function successfully executes.
ModuloPosition	LREAL		User units	Returns the Axis Value Period.

4.1.4.34.3 Remarks

None

4.1.4.34.4 FBD Language Example

4.1.4.34.5 FFLD Language Example

4.1.4.34.6 IL Language Example

Not available.

4.1.4.34.7 ST Language Example

```
Axis1_Value_Period := MLAxisReadModPos(PipeNetwork.Axis1 )  ;
```

4.1.4.35 MLAxisReadTq

Pipe Network

Function - Returns the actual torque applied by the drive mapped to the specified axis.

4.1.4.35.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	Pipe network identifier of the axis block.

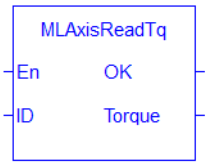
4.1.4.35.2 Outputs

Output	Data Type	Range	Unit	Description
Torque	LREAL	No range	N.m (Newton meter)	Actual torque applied by the drive associated to the axis. If the "MLAxisRatedTq" (→ p. 253) function is not invoked, the Output value is rated motor continuous torque (where 1000.0 = rated torque).

4.1.4.35.3 Remarks

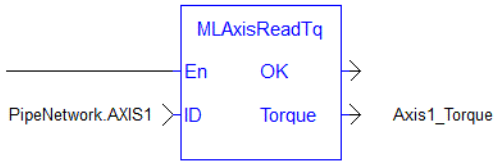
NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.



4.1.4.35.4 FBD Language Example

4.1.4.35.5 FFLD Language Example



4.1.4.35.6 IL Language Example

Not available.

4.1.4.35.7 ST Language Example

```
Axis1_Torque := MLAxisReadTq(PipeNetwork.Axis1 )  ;
```

See Also

- "MLAxisRatedTq" (→ p. 253)
- "MLAxisReadActPos" (→ p. 255)
- "MLAxisReadVel" (→ p. 261)

4.1.4.36 MLAxisReadUUnits

Pipe Network ✓

Function - Gets the User units per revolution value of the axis.

4.1.4.36.1 Inputs

Input	Data Type	Range	Unit	Default	Description
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block.

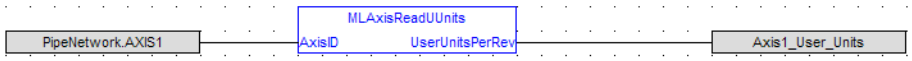
4.1.4.36.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL			Returns TRUE when the function successfully executes.
UserUnitsPerRev	LREAL			Returns the Axis User Units per revolution.

4.1.4.36.3 Remarks

None

4.1.4.36.4 FBD Language Example



4.1.4.36.5 FFLD Language Example

4.1.4.36.6 IL Language Example

Not available.

4.1.4.36.7 ST Language Example

```
Axis1_User_Units := MLAxisReadUUnits(PipeNetwork.Axis1 )  ;
```

4.1.4.37 MLAxisReadVel

Pipe Network

Function - Returns the actual velocity of the axis based on the data provided by the drive's feedback device.

4.1.4.37.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	Pipe network identifier of the axis block.

4.1.4.37.2 Outputs

Output	Data Type	Range	Unit	Description
Velocity	LREAL			The actual velocity of the axis.

4.1.4.37.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

In AKD, S300, and S700 drives:

- The actual velocity is calculated internally by the drive.
- The **Velocity Actual Value** object (CoE object 0x606C, subindex 0) must be included in the drive's Input (Tx) PDO data for the controller to read the axis actual velocity from the drives.
  - This is added using the PDO Editor Tab.
- By default, the **Velocity Actual Value** object is included in the AKD PDOs 0x1B20, 0x1B22, and 0x1B23.

4.1.4.37.4 FBD Language Example

4.1.4.37.5 FFLD Language Example

4.1.4.37.6 IL Language Example

Not available.

4.1.4.37.7 ST Language Example

```
Axis1_Velocity := MLAxisReadVel(PipeNetwork.Axis1 )  ;
```

See Also

- "MLAxisReadActPos" (→ p. 255)
- "MLAxisReadTq" (→ p. 259)

4.1.4.38 MLAxisReAlgnRdy

Function - Verifies if an axis is ready.

4.1.4.38.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.

4.1.4.38.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

4.1.4.38.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- Returns TRUE if the internal realignment axis is ready.

4.1.4.38.4 FBD Language Example

4.1.4.38.5 FFLD Language Example

4.1.4.38.6 IL Language Example

Not available.

4.1.4.38.7 ST Language Example

```
MLAxisReAlgnRdy(PipeNetwork.Axis1 )  ;
```

See Also

"MLAxisReAlign" (→ p. 263)

4.1.4.39 MLAxisReAlign

Pipe Network

Function - Realigns the actual position with the reference position by moving the axis by the specified delta position.

4.1.4.39.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Acceleration	LREAL	No range	User unit/sec^2	No default	Sets the Realign Acceleration. See Set the Axis Block Position Units for more information.
Deceleration	LREAL	No range	User unit/sec^2	No default	Sets the Realign Deceleration rate. See Set the Axis Block Position Units for more information.
DeltaPos	LREAL	No range	User units	No default	Sets the Axis Delta Position or the relative distance to be moved. See Set the Axis Block Position Units for more information.
ID	DINT	No range	N/A	No default	ID Name of the Axis block.
Speed	LREAL	No range	User unit/sec	No default	Sets the Axis Speed. See Set the Axis Block Position Units for more information.

4.1.4.39.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

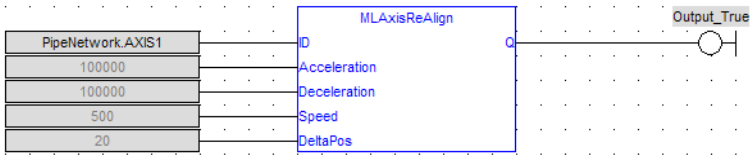
4.1.4.39.3 Remarks

NOTE

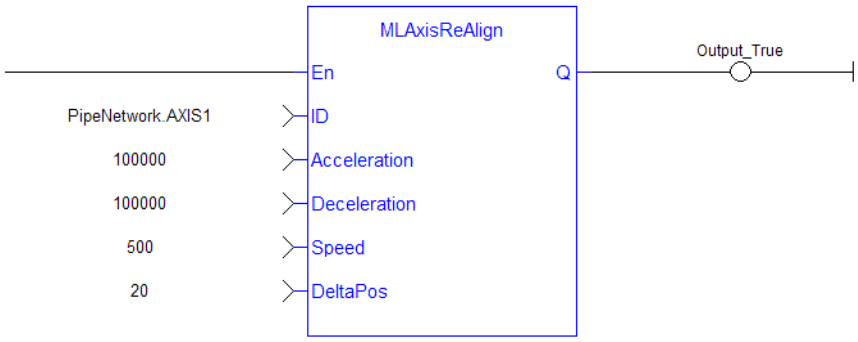
The realign function does not work properly if the "MLAxisStop" (→ p. 269) function is continuously executed using its Start input.

- When stopping the drive a motion profile is applied to decelerate.
- During the deceleration, the Reference position changes.
- Calling **MLAxisReAlign** realigns the actual position with the reference position by moving the axis by the specified delta position, typically calculated by the application code.
- After a "MLAxisStop" (→ p. 269) is executed, **MLAxisReAlign** is required for the Pipe Position to be used again.

4.1.4.39.4 FBD Language Example



4.1.4.39.5 FFLD Language Example



4.1.4.39.6 IL Language Example

Not available.

4.1.4.39.7 ST Language Example

```
MLAxisReAlign(PipeNetwork.Axis1, 100000, 100000, 500, 20 )  ;
```

See Also

"MLAxisReAlgnRdy" (→ p. 262)

4.1.4.40 MLAxisRel

Pipe Network ✓

Function - A selected Axis performs a move for a specified distance relative to the current position.

4.1.4.40.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.
DeltaPosition	LREAL	No range	User units	No default	Sets the Axis Delta Position or the relative distance to be moved. See Set the Axis Block Position Units for more information.

4.1.4.40.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes. • This occurs immediately after the function is called. • The function does not wait for the motion profile to be completed. •

4.1.4.40.3 Remarks

NOTE

Use MLAxisGenIsRdy to know when a move has completed.
The output of **MLAxisRel** can occur before moves have finished.

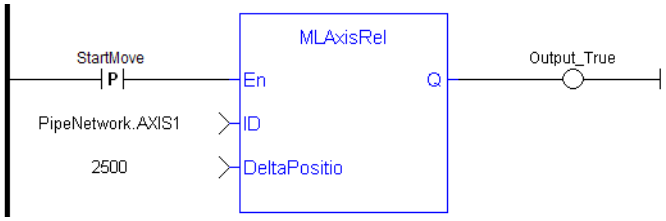
- See "Examples of Axis Functions" (→ p. 278) for more information.

4.1.4.40.3.1 DeltaPosition Axis

- The **DeltaPosition** input is signed so:
  - The move can be in the positive or negative direction.
  - The Axis moves this distance in User units.
- The travel speed, acceleration, deceleration, and User units of the move are values inherited from the selected Axis.
- The default settings are entered when an Axis is created and initiated, and can be changed with other MLAxis commands.

4.1.4.40.4 FBD Language Example

4.1.4.40.5 FFLD Language Example



NOTE

A pulse contact is required to start the FB.

4.1.4.40.6 IL Language Example

Not available.

4.1.4.40.7 ST Language Example

```
MLAxisRel(PipeNetwork.Axis1, 2500 )  ;
```

See Also

- "MLAxisGenWriteAcc" (→ p. 243)
- "MLAxisGenWriteDec" (→ p. 244)

- "MLAxisGenWriteSpd" (→ p. 245)
- "MLAxisWriteUUnits" (→ p. 276)

4.1.4.41 MLAxisResetErrors

Pipe Network

Function - Clears errors of the specified axis.

4.1.4.41.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.

4.1.4.41.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

4.1.4.41.3 Remarks

- Previous function name: MLAxisClrErrors

4.1.4.41.4 FBD Language Example

4.1.4.41.5 FFLD Language Example

4.1.4.41.6 IL Language Example

Not available.

4.1.4.41.7 ST Language Example

```
MLAxisResetErrors( PipeNetwork.Axis1 ) ;
```

4.1.4.42 MLAxisRstFastIn

Pipe Network

Function - Writes in the Latch Control Word to reset the Fast Input.

4.1.4.42.1 Inputs

Input	Data Type	Range	Unit	Default	Description
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block.

Input	Data Type	Range	Unit	Default	Description
InputID	DINT	No range	N/A	No default	ID name of the Fast input of an axis (i.e., IN1 and IN2 on S300). **InputID** INT Range = 0, 1 • 0 = Touch Probe 1 / Capture Engine 0 • 1 = Touch Probe 2 / Capture Engine 1

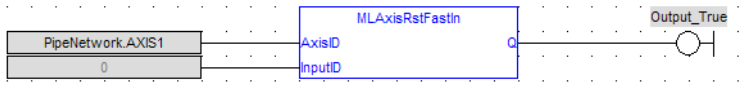
4.1.4.42.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

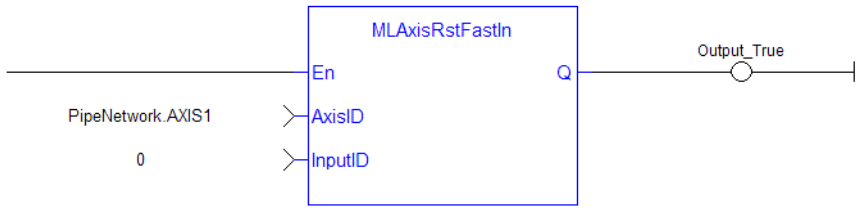
4.1.4.42.3 Remarks

None

4.1.4.42.4 FBD Language Example



4.1.4.42.5 FFLD Language Example



4.1.4.42.6 IL Language Example

Not available.

4.1.4.42.7 ST Language Example

```
MLAxisRstFastIn(PipeNetwork.Axis1, 0 )  ;
```

See Also

- "MLAxisCfgFastIn" (→ p. 233)
- "MLAxisIsTrigged" (→ p. 249)

4.1.4.43 MLAxisStatus

Function - Returns the status of the axis.

4.1.4.43.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.

4.1.4.43.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL			Returns TRUE when the function successfully executes.
Default (.Q)	DINT			Returns the status of the axis.

Bit	Description
0	Initialized (1 if initialized).
1	• Power (1 if power is on). • Is linked to Bit 1 (Switched on) of the Status Word.
2	• Enabled (1 if enabled). • Is linked to Bit 0 (Ready to switch on) of the Status Word.
3	• Found (1 if found on the network). • EtherCAT state is Pre-Operational.
4	• Configured (1 if configured) EtherCAT state is **Safe-Operational**.
5	• Running (1 if running) EtherCAT state is **Operational**.
6	Error (1 if in error).
7	Simulated (1 if working with a simulated axis).
8	Connected (1 if a pipe is connected).
9	Warning (1 if the drive signals a warning).
10	Stopping (1 if the drive is performing a Stop).
11	Stopped (1 if the drive has finished the Stop).
12 to 31	Reserved

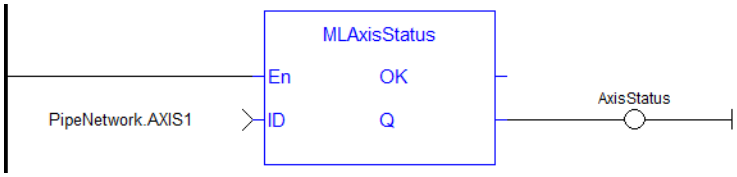
4.1.4.43.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

4.1.4.43.4 FBD Language Example

4.1.4.43.5 FFLD Language Example



4.1.4.43.6 IL Language Example

Not available.

4.1.4.43.7 ST Language Example

```
AxisStatus := MLAxisStatus(PipeNetwork.AXI_A1_Axis) ;
IF AxisStatus.11 THEN
MLAxisStop(PipeNetwork.AXI_A1_Axis,FALSE,DEF_A1_StopDec) ;
END_IF;
```

```
AxisStatus := MLAxisStatus(PipeNetwork.AXIS1);
If AxisStatus.0 Then
   (*Axis is initialized*)
ElsIf AxisStatus.1 Then
   (*Axis' power is ON*)
ElsIf AxisStatus.2 Then
   (*Axis is READY to be enabled*)
End_If;
```

4.1.4.44 MLAxisStop

Pipe Network ✓

Function Block - Stop with the specified deceleration.
This does NOT remove the input source, but to stop the drive from continuing to move.

4.1.4.44.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.
Start	BOOL	0, 1	N/A	No default	
Deceleration	LREAL	No range	User unit/sec^2	No default	See Set the Axis Block Position Units for more information.

4.1.4.44.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Becomes TRUE when the Axis is completely stopped. •
GenPos	LREAL		User units	Corresponds to the **Generator Position** input to the axis at the time the stop is triggered. See Axis Pipe Block for more information about Generator Position.
PipePos	LREAL		User units	Corresponds to the **Pipe Position** input to the axis at the time the stop is triggered. See Axis Pipe Block for more information about Pipe Position.

Output	Data Type	Range	Unit	Description

RealignPos	LREAL		User units	Realign Position is the **Reference Position** at which the stop is triggered.
				• The Realign Position is:
				  • Obtained by converting the last value sent to the drive from drive interface units into user units.
				  • Useful to return to the point at which the trajectory was abandoned or to realign the master to the slave.
				• See Axis Pipe Block for more information about Reference Position.

StopPos	LREAL		User units	Corresponds to the last **Reference Position** sent to the drive at the time when the Axis is completely stopped. • It is functionally different than the **Actual Position** because that position is the drive position converted to user units. • The correct delta for the realign move to get in sync with the trajectory in order to realign the slave to the master is the current **Reference Position** minus the **Stop Position** for the realign move. • After stopping, if the axis is disabled and the motor position is manually altered, this distance must be considered when performing the realign.

4.1.4.44.3 Remarks

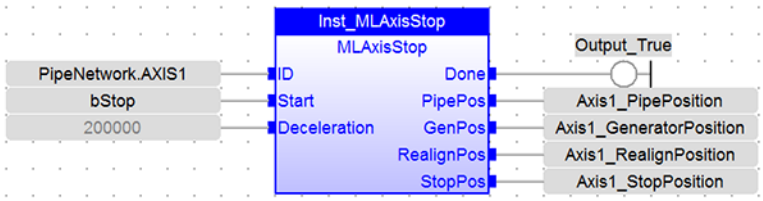
NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

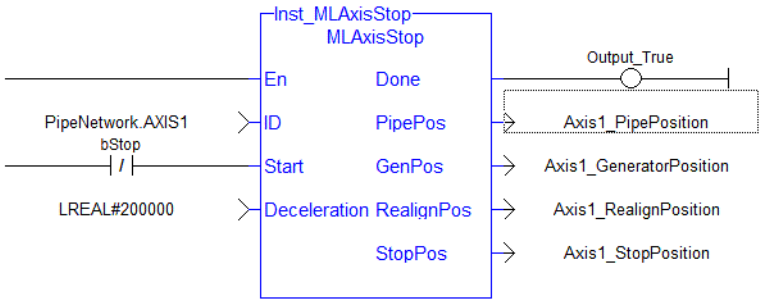
- After stopping the drive, the motion must be restarted by realigning the **Actual Position** with the **Reference Position**.
- When the stop occurs, the master keeps moving and the axis starts ignoring the **Pipe Position** value and begins a controlled stop based on the input parameters.
  See Axis Pipe Block for more information about Pipe Position.
  - At this point, any Axis Block level profile (issued from an FB like MLAxisAbs, MLAxisRel...) are aborted.
  - When the stop is complete, it is up to the application to decide how to move the axis, master, or both to a position where they can be realigned, and the master restarted.

- The "MLAxisReAlign" (→ p. 263) function is used to move the axis to a restart position in order to enable synchronized machine motion to start again.
  - Once the realign function is successfully completed, the **Pipe Position** is again summed with the **Generator Position** to create the **Reference Position**.

4.1.4.44.4 FBD Language Example



4.1.4.44.5 FFLD Language Example



4.1.4.44.6 IL Language Example

Not available.

4.1.4.44.7 ST Language Example

```
Inst_MLAxisStop(PipeNetwork.AXIS1, bStop, 200000);
If Inst_MLAxisStop.Done Then
   Axis1_PipePosition       := Inst_MLAxisStop.PipePos;
   Axis1_GeneratorPosition  := Inst_MLAxisStop.GenPos;
   Axis1_RealignPosition    := Inst_MLAxisStop.RealignPos;
   Axis1_StopPosition       := Inst_MLAxisStop.StopPos;
End_if;
```

See Also

"MLAxisReAlign" (→ p. 263)

4.1.4.45 MLAxisTimeStamp

Pipe Network

Function - Returns the timestamp of the triggered axis.

4.1.4.45.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	No range	N/A	No default	Enables execution.
ID	DINT	No range	N/A	No default	ID Name of the Axis block.
InputID	DINT	0, 1	N/A	No default	ID of the triggered Fast input of an axis. • 0=first • 1=second (i.e., IN1 and IN2 on S300)

Input	Data Type	Range	Unit	Default	Description
edge	DINT	0, 2	N/A	No default	Configures the Inputs as: • 0= Disabled • 1=Rising Edge • 2=Falling edge

4.1.4.45.2 Outputs

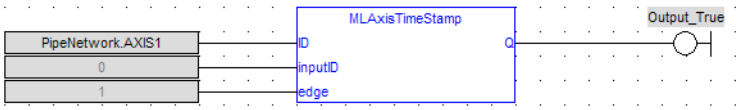
Output	Data Type	Range	Unit	Description
OK	BOOL			Returns TRUE when the function successfully executes.
Q	DINT		Microseconds	Returns the time stamp value. See Interpret a Timestamp for more information.

4.1.4.45.3 Remarks

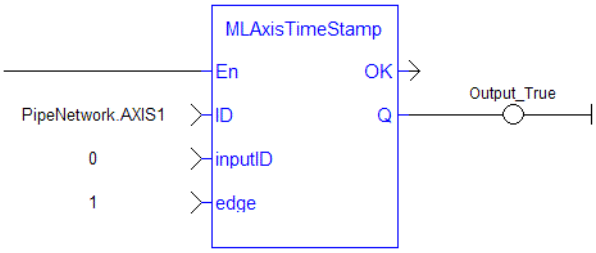
NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

4.1.4.45.4 FBD Language Example



4.1.4.45.5 FFLD Language Example



4.1.4.45.6 IL Language Example

Not available.

4.1.4.45.7 ST Language Example

```
MLAxisTimeStamp(PipeNetwork.Axis1, 0, 1 ) ;
```

See Also

- "MLAxisCfgFastIn" (→ p. 233)
- "MLAxisIsTrigged" (→ p. 249)
- "MLAxisRstFastIn" (→ p. 266)

4.1.4.46 MLAxisWriteModPos

Pipe Network

Function - Sets the value period of the axis.

4.1.4.46.1 Inputs

Input	Data Type	Range	Unit	Default	Description
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block.
ModuloPosition	LREAL	No range	User units	No default	Sets the Axis Period Value when Mode is set to Modulo.

4.1.4.46.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE when the function successfully executes.

4.1.4.46.3 Remarks

None

4.1.4.46.4 FBD Language Example

4.1.4.46.5 FFLD Language Example

4.1.4.46.6 IL Language Example

Not available.

4.1.4.46.7 ST Language Example

```
MLAxisWriteModPos(PipeNetwork.Axis1, 360) )  ;
```

4.1.4.47 MLAxisWritePipPos

Pipe Network

Function - Forces the pipe position internal value.

IMPORTANT

This function works only when no pipe is connected.

Input	Data Type	Range	Unit	Default	Description
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block.
PipePosition	LREAL	No range	User units	No default	Sets the Axis Pipe Position. See Set the Axis Block Position Units for more information.

4.1.4.47.1 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE when the function successfully executes.

4.1.4.47.2 Remarks

None

4.1.4.47.3 FBD Language Example

4.1.4.47.4 FFLD Language Example

4.1.4.47.5 IL Language Example

Not available.

4.1.4.47.6 ST Language Example

```
MLAxisWritePipPos(PipeNetwork.Axis1, 3000 ) ;
```

See Also

- "MLAxisCmdPos" (→ p. 234)
- "MLAxisFBackPos" (→ p. 236)
- "MLAxisGenPos" (→ p. 239)
- "MLAxisPipePos" (→ p. 251)
- "MLAxisReadActPos" (→ p. 255)

4.1.4.48 MLAxisWritePos

Pipe Network

Function - Sets a new value to an axis' current location.

4.1.4.48.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	No range	N/A	No default	ID Name of the Axis block.
Position	LREAL	No range	User units	No default	The new value for the axis' current location.

4.1.4.48.2 Outputs

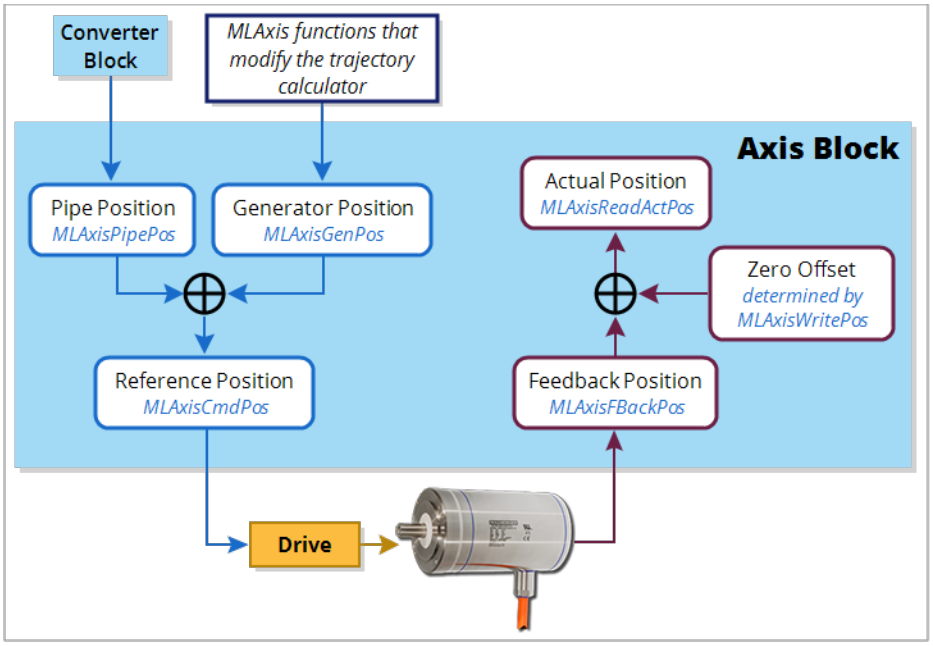
Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE when the function successfully executes.

4.1.4.48.3 Remarks

- Previous function name: MLAxisSetZero
- After this function is called, the axis' current location has a value equal to the **Position** argument.

Associated Data on Positions

This data are illustrated here:



NOTE

All positions are in user units with modulo applied if active, unless specified.

Position / Offset	Description
Actual Position	This is the actual position of the underlying axis as reported by the drive. • It is the sum of the feedback value (Position actual value) returned from the communication link to the drive and any zero-offset due to an MLWritePos function block ("MLAxisWritePipPos" (→ p. 273), "MLAxisWritePos" (→ p. 274)). `ActualPos := FeedbackPos + ZeroOffset`
Feedback Position	This is the current position the drive reports for an axis, scaled to user units. It does not take into account the value of the Zero Offset or axis modulo.
Generator Position	This is the summation of all previous commands (i.e., calls to functions which perform motion such as "MLAxisAbs" (→ p. 226), "MLAxisMoveVel" (→ p. 250), and "MLAxisRel" (→ p. 264)) to the Axis internal motion generator. • It is modified by "MLAxisWritePos" (→ p. 274) to insure no jumps in the **Reference Position** command. • It accumulates changes in pipe position due to activate and deactivation of the pipe the Axis block is associated with.

Position / Offset	Description
Pipe Position	This is the output of the convertor block is written into the **Pipe Position** value whenever the Convertor block is connected to the axis and the pipe is active.
Reference Position	This is the commanded axis position sent to the drive. It is the summation of **Pipe Position** and **Generator Position**. ReferencePosition = Pipe Position + Generator Position
Zero Offset	This adjusts the coordinate system so the Actual Position reports correct values after homing or using "MLAxisWritePos" (→ p. 274).

4.1.4.48.4 FBD Language Example

4.1.4.48.5 FFLD Language Example

4.1.4.48.6 IL Language Example

Not available.

4.1.4.48.7 ST Language Example

```
MLAxisWritePos(PipeNetwork.Axis1, 0) ;
```

4.1.4.49 MLAxisWriteUUnits

Pipe Network

Function - Set the user units per revolution value of the axis.

4.1.4.49.1 Inputs

Input	Data Type	Range	Unit	Default	Description
AxisID	DINT	No range	N/A	No default	ID Name of the Axis block.

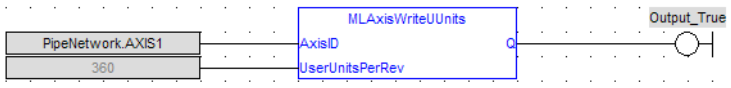
4.1.4.49.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE when the function successfully executes.
UserUnitsPerRev	LREAL	No range	N/A	Sets the Axis User Units per revolution.

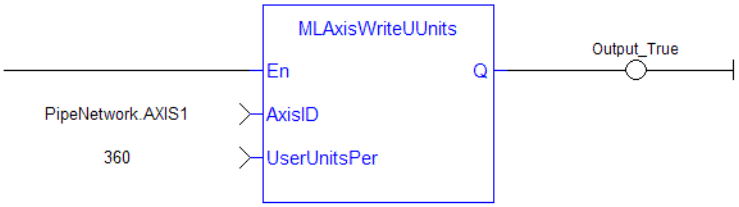
4.1.4.49.3 Remarks

- User units are user-defined position units used in the KAS application.
- Selected units must be as natural as possible and must make sense for the machine.
  - It must be related to the final moving object (e.g., the driven belt rather than the axis shaft).
  - The same unit must be used for all related axes for simplicity reasons.
- Speeds are defined in [User unit/sec] and accelerations in [User unit/sec2].

4.1.4.49.4 FBD Language Example



4.1.4.49.5 FFLD Language Example



4.1.4.49.6 IL Language Example

Not available.

4.1.4.49.7 ST Language Example

```
MLAxisWriteUUnits(PipeNetwork.Axis1, 360 ) ;
```

4.1.4.50 MLPNAxisCreate

Pipe Network

Function - Creates a new axis object.

4.1.4.50.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Name	STRING	No range	N/A	No default	Name of the created Axis.
DriverName	STRING	No range	N/A	No default	Is the Motion bus driver name or Simulated.
DriveAxisNumber	UINT	1, 256	N/A	No default	• This one-based number specifies the axis on the drive. • For a single-axis drive, this number should be 1.
Address	DINT	No range	N/A	No default	Axis motion bus address.

4.1.4.50.2 Outputs

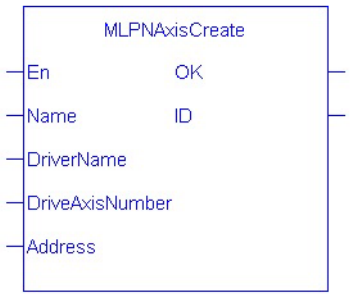
Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Returns TRUE when the function successfully executes.

4.1.4.50.3 Remarks

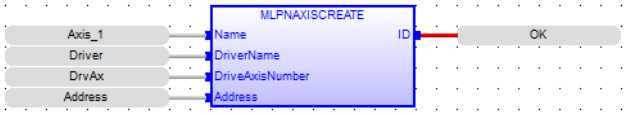
Returns the ID of the newly created axis object or 0 if the function failed.

TIP

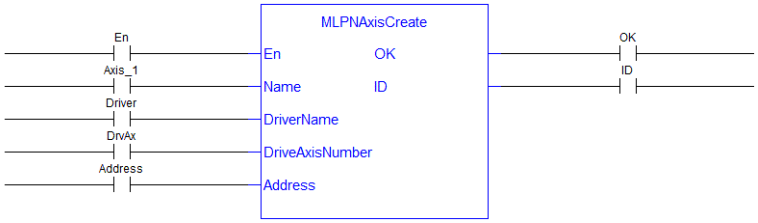
This function should be called after "MLMotionInit" (→ p. 549) is called and before "MLMotionStart" (→ p. 552) is called.



4.1.4.50.4 FBD Language Example



4.1.4.50.5 FFLD Language Example



4.1.4.50.6 IL Language Example

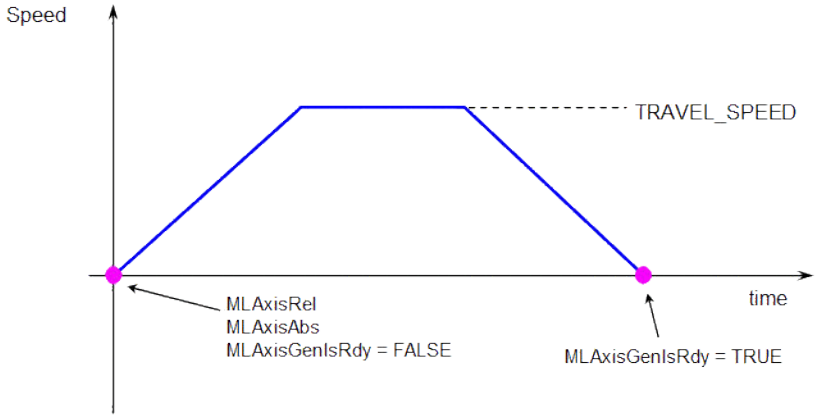
Not available.

4.1.4.50.7 ST Language Example

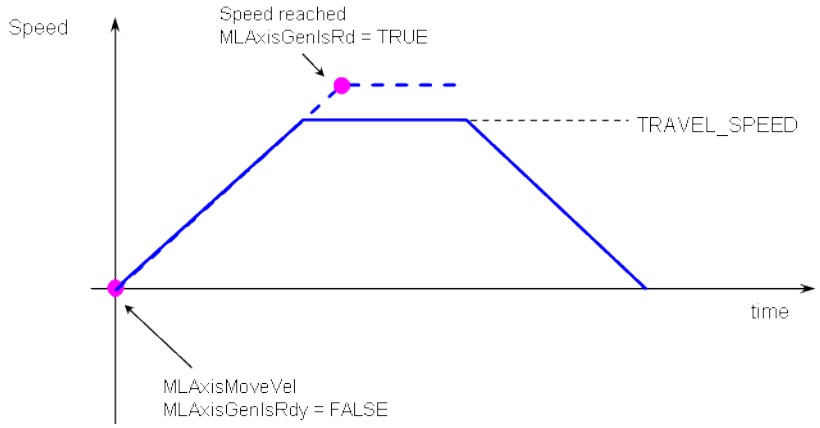
```
PipeNetwork.AXIS1 := MLPNAxisCreate('AXIS1','SercosDriver',0,1001);
```

4.1.4.51 Examples of Axis Functions

4.1.4.51.1 MLAxisMoveVel(Speed)



- **MLAxisMoveVel(Speed)** starts to run the axis.
- **MLAxisGenIsRdy** returns TRUE when the Speed is reached.



4.1.4.51.2 MLAxisMoveVel(0.0)

- **MLAxisMoveVel(0.0)** reduces the speed down to 0 (zero).
- **MLAxisGenIsRdy** returns TRUE once the axis is ready.

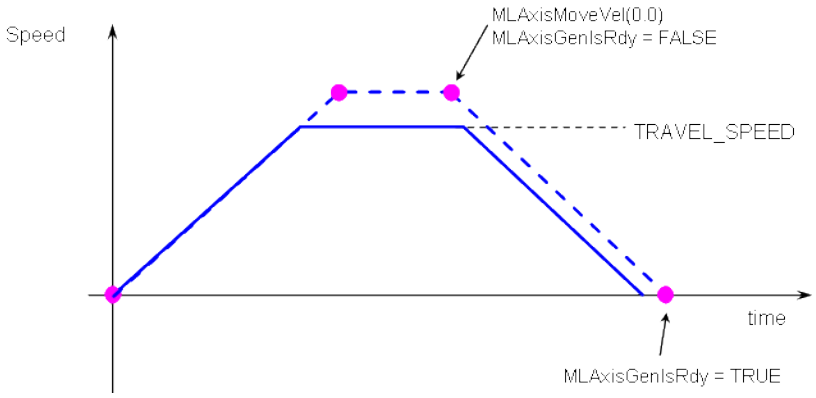


Figure 4-19: Axis Functions Usage

4.1.5 CAM Profile

Name	Description
MLCamInit	Initializes a CAM Pipe Block for use in a PLC Program.
MLCamSwitch	Switches the CAM Profile in a selected CAM object.
MLPrfReadIOffset	Returns the Input Offset value of a selected CAM Profile.
MLPrfReadIScale	Returns the Input Ratio value of a selected CAM Profile.
MLPrfReadOOffset	Returns the Output Offset value of a selected CAM Profile.
MLPrfReadOScale	Returns the Output Ratio value of a selected CAM Profile.
MLPrfWriteIOffset	Sets the Input Offset value of a selected CAM Profile.
MLPrfWriteIScale	Sets the Input Ratio value of a selected CAM Profile.
MLPrfWriteOOffset	Sets the Output Offset value of a selected CAM Profile.
MLPrfWriteOScale	Sets the Output Ratio value of a selected CAM Profile.

Name	Description
MLProfileBuild	Creates a CAM Profile from application data that can be executed by a CAM block in Pipe Network or PLCopen.
MLProfileCreate	Creates a new CAM Profile Object for use in a PLC Program or Pipe Network CAM block.
MLProfileInit	Initializes a previously created CAM Profile object for use in a PLC Program or Pipe Network CAM block.
MLProfileRelease	Removes a Profile so the ProfileID can be used by a different or new Profile.

4.1.5.1 MLCamInit

Function - Initializes a CAM Pipe Block for use in a PLC Program.

4.1.5.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	CAM	ID number of a created Pipe Block.
ProfileName	STRING	No range	N/A	No default	Name of the current profile assigned to the CAM. It must be a declared profile object.
ModuloPosition	LREAL	No range	User units	360.0	Value of the period of the cam output values expressed in user units, for a cyclic system.

4.1.5.1.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the CAM Pipe Block is initialized.

4.1.5.1.3 Remarks

This function block is automatically called if a CAM Block is added to the Pipe Network, with user-defined settings then entered in the Pipe Blocks Properties screen.

- The CAM Pipe Block is used to generate motion profiles of any shape.
  - These profiles are created and initiated separately and the shape is modified with the CAM Editor.
  - With the Editor profiles can be changed graphically or by manually changing values in a numeric table relating input and output values with specific slopes.
  - The CAM Editor software tool provides the capability to visualize, analyze, edit, and smooth profiles.

With the Pipe Network (PN) CAM block:

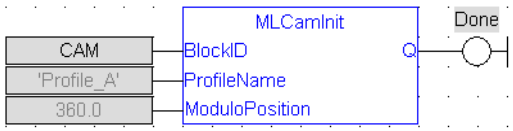
- The CAM block's profile is in reference to the input positions coming into the PN CAM block (Master Absolute)
- The PN CAM block output positions are in reference to PN CAM block's output position at the end of the last CAM cycle (Slave Relative).

- Profile switching can be done on the fly, without losing synchronization and without dead time.
  - In addition, the offsets and ratios of CAM Profiles can be changed on the fly.
  - See CAM Profile Switching for more information.

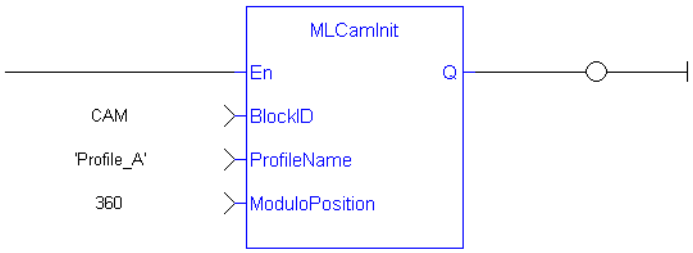
NOTE

CAM objects are normally created in the Pipe Network using the graphical engine.
You do not have to add **MLCamInit** function blocks to their programs.
Parameters are entered directly in pop-up windows and the code is automatically added to the current project.

4.1.5.1.4 FBD Language Example



4.1.5.1.5 FFLD Language Example



4.1.5.1.6 IL Language Example

Not available.

4.1.5.1.7 ST Language Example

```
//Initialize  a Pipe Network block named "CAM" with a profile named "Profile_
A", set the cam modulo position to 360
CAM := MLBlkCreate( 'CAM', 'CAM' );
MLCamInit( CAM, 'Profile_A', 360.0 );
```

See Also

- "MLProfileCreate" (→ p. 542)
- "MLProfileInit" (→ p. 544)

4.1.5.2 MLCamSwitch

Pipe Network ✓

Function - Switches the CAM Profile in a selected CAM object.

4.1.5.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized CAM Pipe Block.
ProfileID	DINT	-2147483648 to 2147483647	N/A	No default	Name of the new CAM Profile which is assigned to the CAM Pipe Block. It must be a declared profile object.

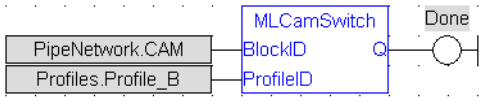
4.1.5.2.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the CAM Profile is changed.

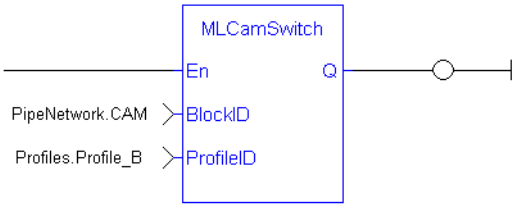
4.1.5.2.3 Remarks

- Can be used in combination with a comparator to verify profiles are switched at a time where the input and output values of both the old and new profiles are equal.
  - This is so an Axis receives continuous position values and does not jump.
- These profiles are created and initiated separately and the shape is created with the CAM Editor.
  - With the Editor, profiles can be changed graphically or by manually changing values in a numeric table relating input and output values with specific slopes.
  - The CAM Editor is used to visualize, analyze, edit, and smooth profiles.
- See CAM Profile Switching for more information.

4.1.5.2.4 FBD Language Example



4.1.5.2.5 FFLD Language Example



4.1.5.2.6 IL Language Example

Not available.

4.1.5.2.7 ST Language Example

```
//Switch CAM Profile
MLCamSwitch(PipeNetwork.CAM, Profiles.Profile_B);
```

See Also

- "MLPrfWriteIOffset" (→ p. 287)
- "MLPrfWriteOScale" (→ p. 291)
- "MLProfileCreate" (→ p. 542)
- "MLProfileInit" (→ p. 544)

4.1.5.3 MLPrfReadIOffset

Function - Returns the Input Offset value of a selected CAM Profile.

4.1.5.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ProfileID	DINT	-2147483648 to 2147483647	N/A	No default	Name of an initialized CAM Profile.

4.1.5.3.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Returns TRUE when the function successfully executes.
Offset	LREAL	No range	N/A	Returns the Input Offset of the selected CAM Profile.

4.1.5.3.3 Remarks

- Offsets can be changed on the fly to modify the CAM Profile while maintaining its shape.
- A change in input offset is equivalent to shifting the CAM Profile on the X or Input Axis.

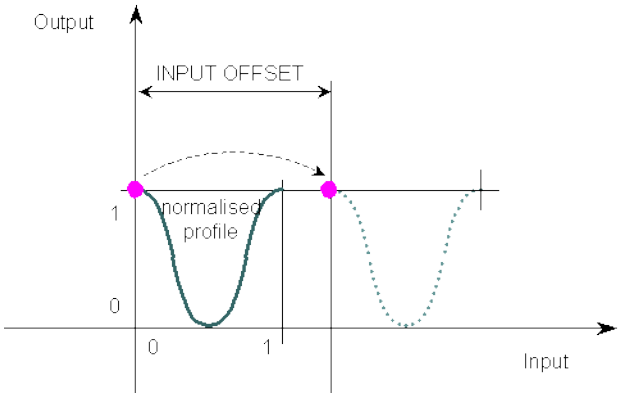
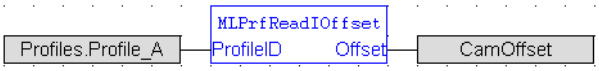


Figure 4-20: MLPrfReadIOffset

4.1.5.3.4 FBD Language Example



4.1.5.3.5 FFLD Language Example

4.1.5.3.6 IL Language Example

Not available.

4.1.5.3.7 ST Language Example

```
//Save value of input offset
CamOffset := MLPrfReadIOffset( Profiles.Profile_A );
```

See Also

- "MLPrfWriteIOffset" (→ p. 287)
- "MLProfileCreate" (→ p. 542)
- "MLProfileInit" (→ p. 544)

4.1.5.4 MLPrfReadIScale

Pipe Network

Function - Returns the Input Ratio value of a selected CAM Profile.

4.1.5.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ProfileID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized CAM Profile.

4.1.5.4.2 Outputs

Output	Data Type	Range	Unit	Description
Ratio	LREAL			Returns the Input Ratio value of a selected CAM Profile.

4.1.5.4.3 Remarks

- Previous function name: MLPrfGetIRatio
- Ratios can be changed on the fly to modify the CAM Profile while maintaining its basic shape.
- A change in input ratio is equivalent to stretching the CAM Profile on the X (or Input) Axis.
- A negative value is not allowed.

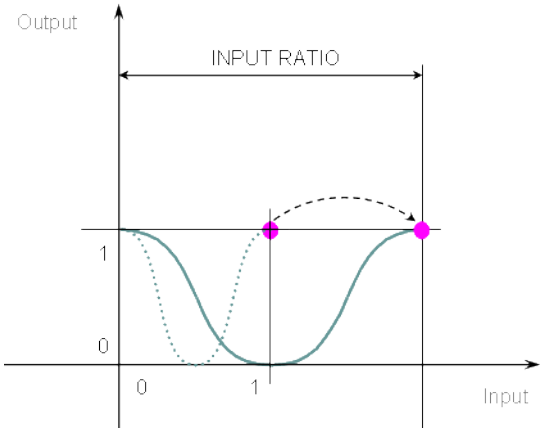


Figure 4-21: MLPrfReadIScale

4.1.5.4.4 FBD Language Example

4.1.5.4.5 FFLD Language Example

4.1.5.4.6 IL Language Example

Not available.

4.1.5.4.7 ST Language Example

```
//Save value of input ratio
CamRatio := MLPrfReadIScale( Profiles.Profile_A );
```

See Also

- "MLPrfWriteIScale" (➔ p. 289)
- "MLProfileCreate" (➔ p. 542)
- "MLProfileInit" (➔ p. 544)

4.1.5.5 MLPrfReadOOffset

Pipe Network

Function - Returns the Output Offset value of a selected CAM Profile.

4.1.5.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ProfileID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized CAM Profile.

4.1.5.5.2 Outputs

Output	Data Type	Range	Unit	Description
Offset	LREAL			Returns the Output Offset of the selected CAM Profile.

4.1.5.5.3 Remarks

- Offsets can be changed on the fly to modify the CAM Profile while maintaining its shape.
- A change in output offset is equivalent to shifting the CAM Profile on the Y (or Output) Axis.

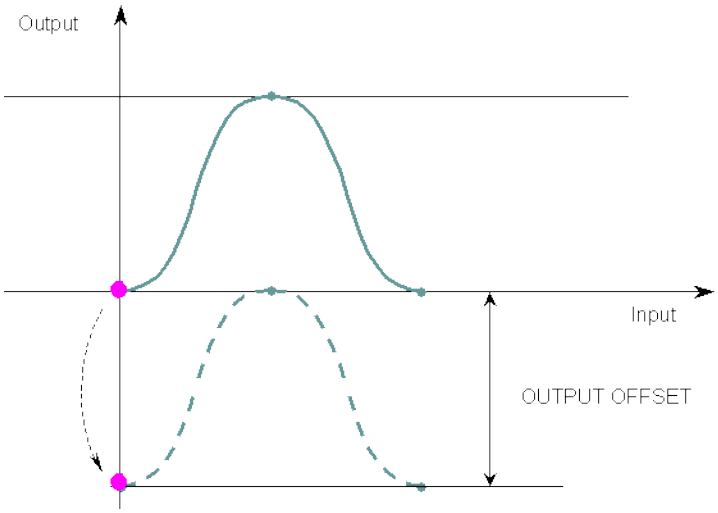
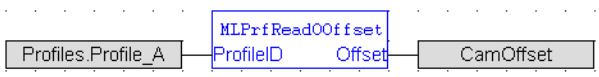


Figure 4-22: MLPrfReadOOffset

4.1.5.5.4 FBD Language Example



4.1.5.5.5 FFLD Language Example

4.1.5.5.6 IL Language Example

Not available.

4.1.5.5.7 ST Language Example

```
//Save value of output offset
CamOffset := MLPrfReadOOffset( Profiles.Profile_A );
```

See Also

- "MLPrfWriteOOffset" (→ p. 290)
- "MLProfileCreate" (→ p. 542)
- "MLProfileInit" (→ p. 544)

4.1.5.6 MLPrfReadOScale

Pipe Network

Function - Returns the Output Ratio value of a selected CAM Profile.

4.1.5.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ProfileID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized CAM Profile.

4.1.5.6.2 Outputs

Output	Data Type	Range	Unit	Description
Ratio	LREAL			Returns the Output Ratio value of a selected CAM Profile.

4.1.5.6.3 Remarks

- Previous function name: MLPrfGetORatio
- Ratios can be changed on the fly to modify the CAM Profile while maintaining its basic shape.
- A change in output ratio is equivalent to stretching, and flipping if negative, the CAM Profile on the Y (or Output) Axis.

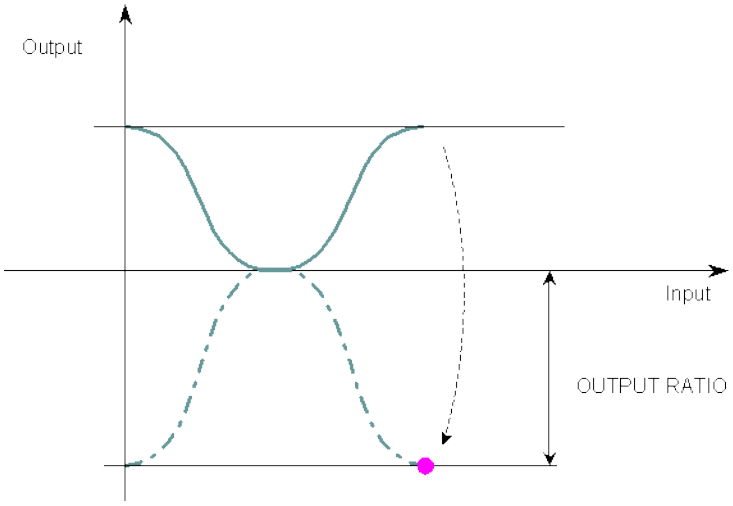


Figure 4-23: MLPrfReadOScale

4.1.5.6.4 FBD Language Example

4.1.5.6.5 FFLD Language Example

4.1.5.6.6 IL Language Example

Not available.

4.1.5.6.7 ST Language Example

```
//Save value of output ratio
CamRatio := MLPrfReadOScale( Profiles.Profile_A );
```

See Also

- "MLPrfWriteOScale" (→ p. 291)
- "MLProfileCreate" (→ p. 542)
- "MLProfileInit" (→ p. 544)

4.1.5.7 MLPrfWriteIOffset

Pipe Network

Function - Sets the Input Offset value of a selected CAM Profile.

4.1.5.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ProfileID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized CAM Profile.

Input	Data Type	Range	Unit	Default	Description
Offset	LREAL	No range	N/A	No default	Desired new value of Input Offset.

4.1.5.7.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the Input Offset is changed to the new value.

4.1.5.7.3 Remarks

- Offsets can be changed on the fly to modify the CAM Profile while maintaining its shape.
- A change in input offset is equivalent to shifting the CAM Profile on the X (or Input) Axis.

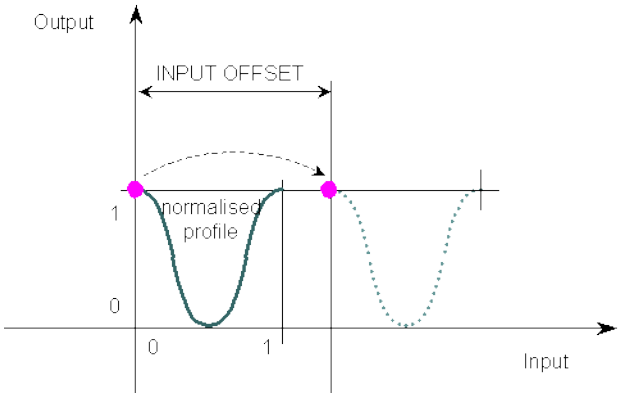
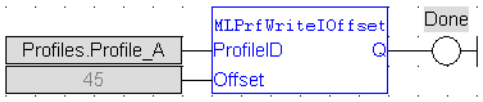
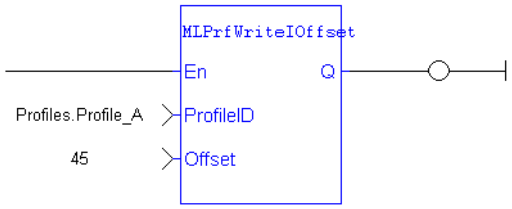


Figure 4-24: MLPrfWriteIOffset

4.1.5.7.4 FBD Language Example



4.1.5.7.5 FFLD Language Example



4.1.5.7.6 IL Language Example

Not available.

4.1.5.7.7 ST Language Example

```
//Change the value of input offset
MLPrfWriteIOffset( Profiles.Profile_A , 45 );
```

See Also

- "MLPrfReadIOffset" (→ p. 282)
- "MLProfileCreate" (→ p. 542)
- "MLProfileInit" (→ p. 544)

4.1.5.8 MLPrfWriteIScale

Pipe Network

Function - Sets the Input Ratio value of a selected CAM Profile.

4.1.5.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ProfileID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized CAM Profile.
Ratio	LREAL	Positive	N/A	No default	Desired new value of Input Ratio.

4.1.5.8.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the Input Ratio is changed to the new value.

4.1.5.8.3 Remarks

- Previous function name: MLPrfSetIRatio
- Ratios can be changed on the fly to modify the CAM Profile while maintaining its basic shape.
- A change in input ratio is equivalent to stretching the CAM Profile on the X (or Input) Axis.

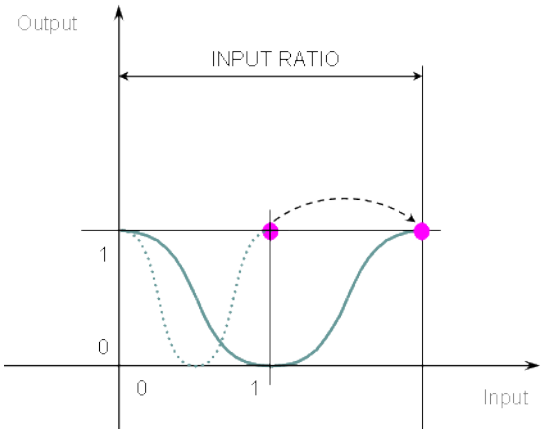
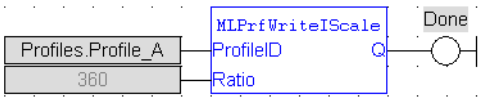
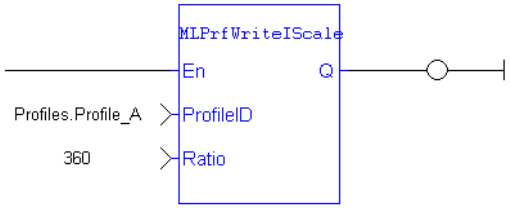


Figure 4-25: MLPrfWriteIScale

4.1.5.8.4 FBD Language Example



4.1.5.8.5 FFLD Language Example



4.1.5.8.6 IL Language Example

Not available.

4.1.5.8.7 ST Language Example

```
//Change value of input ratio
MLPrfWriteIScale( Profiles.Profile_A, 360 );
```

See Also

- "MLPrfReadIScale" (→ p. 284)
- "MLProfileCreate" (→ p. 542)
- "MLProfileInit" (→ p. 544)

4.1.5.9 MLPrfWriteOOffset

Pipe Network

Function - Sets the Output Offset value of a selected CAM Profile.

4.1.5.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ProfileID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized CAM Profile.
Offset	LREAL	No range	N/A	No default	Desired new value of Output Offset.

4.1.5.9.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE if the Output Offset is changed to the new value.

4.1.5.9.3 Remarks

- Offsets can be changed on the fly to modify the CAM Profile while maintaining its shape.
- A change in output offset is equivalent to shifting the CAM Profile on the Y (or Output) Axis.

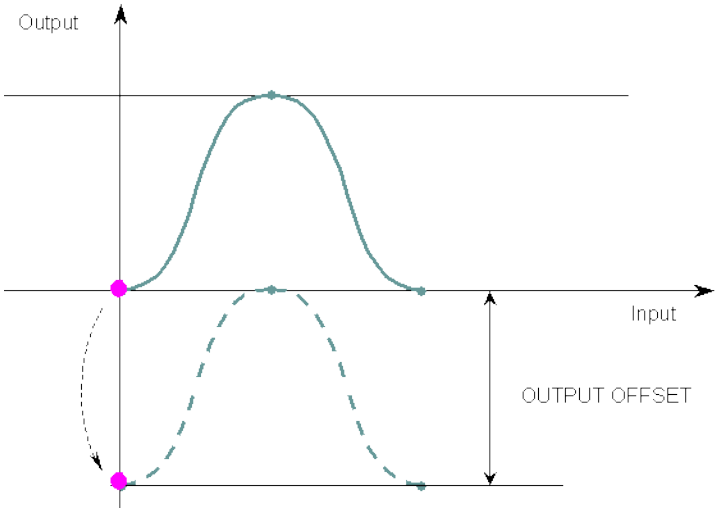
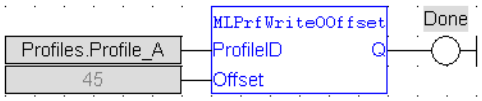


Figure 4-26: MLPrfWriteOOffset

4.1.5.9.4 FBD Language Example



4.1.5.9.5 FFLD Language Example

4.1.5.9.6 IL Language Example

Not available.

4.1.5.9.7 ST Language Example

```
//Change value of output offset
MLPrfWriteOOffset( Profiles.Profile_A , 45 );
```

See Also

- "MLPrfReadOOffset" (→ p. 285)
- "MLProfileCreate" (→ p. 542)
- "MLProfileInit" (→ p. 544)

4.1.5.10 MLPrfWriteOScale

Pipe Network

Function - Sets the Output Ratio value of a selected CAM Profile.

4.1.5.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ProfileID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized CAM Profile.
Ratio	LREAL	No range	N/A	No default	Desired new value of Output Ratio.

4.1.5.10.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the Output Ratio is changed to the new value.

4.1.5.10.3 Remarks

- Previous function name: MLPrfSetORatio
- Ratios can be changed on the fly to modify the CAM Profile while maintaining its basic shape.
- A change in output ratio is equivalent to stretching (flipping if negative - see "MLPrfWriteOScale" (→ p. 292)) the CAM Profile on the Y (or Output) Axis.

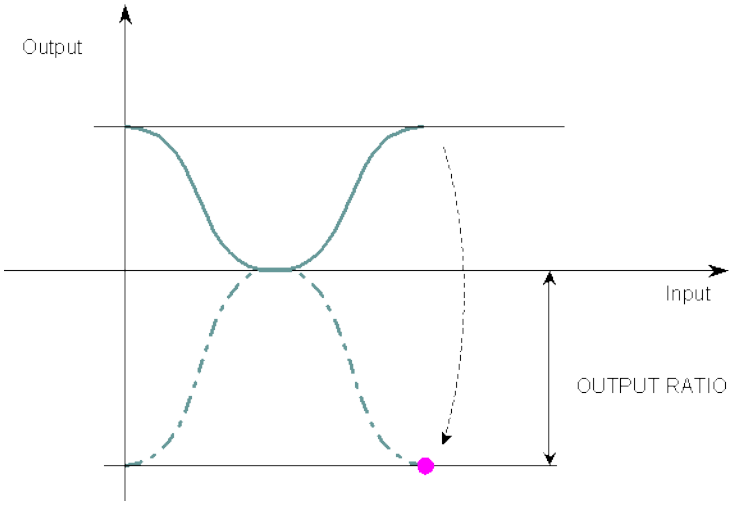
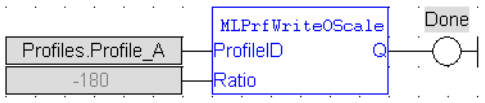
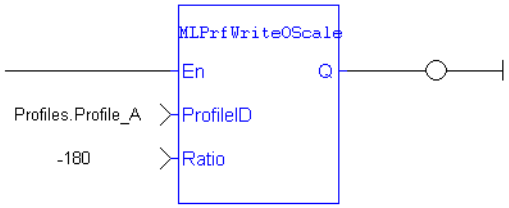


Figure 4-27: MLPrfWriteOScale

4.1.5.10.4 FBD Language Example



4.1.5.10.5 FFLD Language Example



4.1.5.10.6 IL Language Example

Not available.

4.1.5.10.7 ST Language Example

```
//Change value of output ratio
MLPrfWriteOScale( Profiles.Profile_A , -180 );
```

See Also

- "MLPrfReadOScale" (→ p. 286)
- "MLProfileCreate" (→ p. 542)
- "MLProfileInit" (→ p. 544)

4.1.6 Comparator

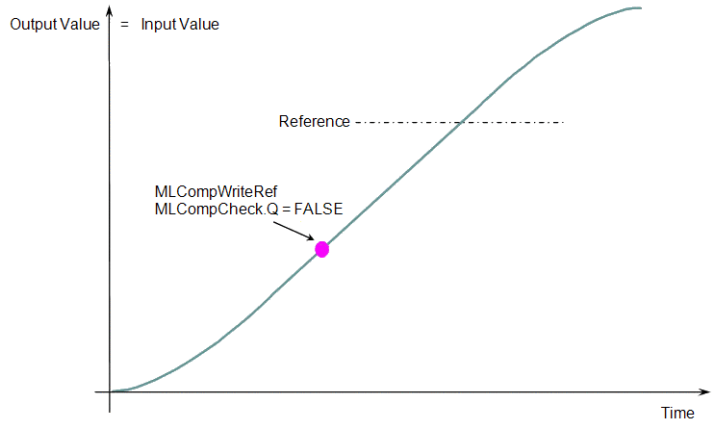
TIP

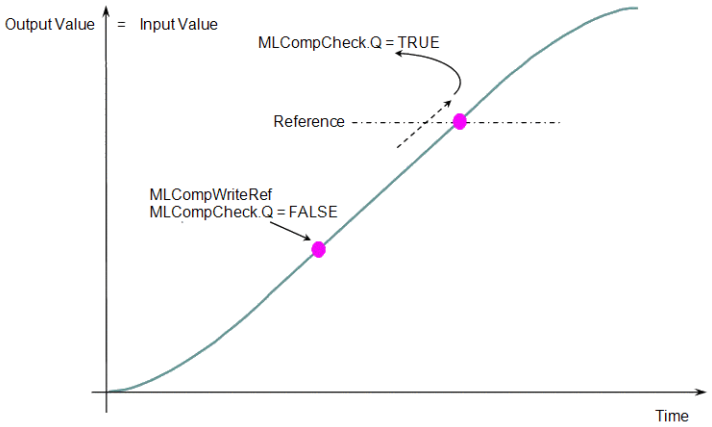
- See "Examples: Comparator Functions" (→ p. 293) for a Comparator function example.

Name	Description
MLCompCheck	Verifies if the reference of a comparator Pipe Block has been crossed.
MLCompInit	Initializes a comparator Pipe Block with user-defined settings for use in a PLC Program.
MLCompReadRef	Returns the reference position of a comparator block.
MLCompReset	Clears the Transition Flag of a comparator Pipe Block.
MLCompWriteRef	Sets the reference position of a comparator block.

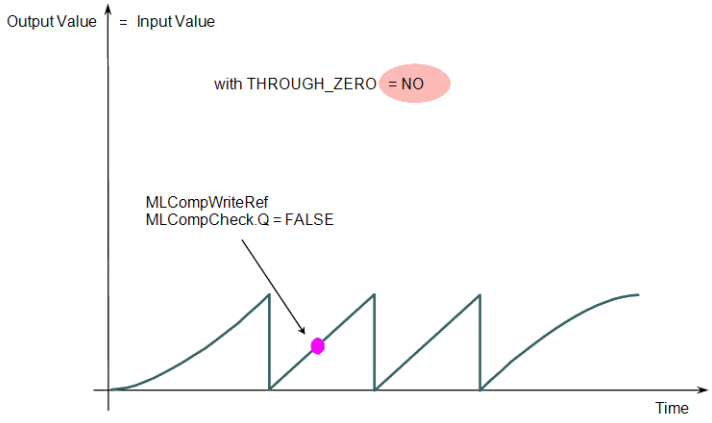
4.1.6.1 Examples: Comparator Functions

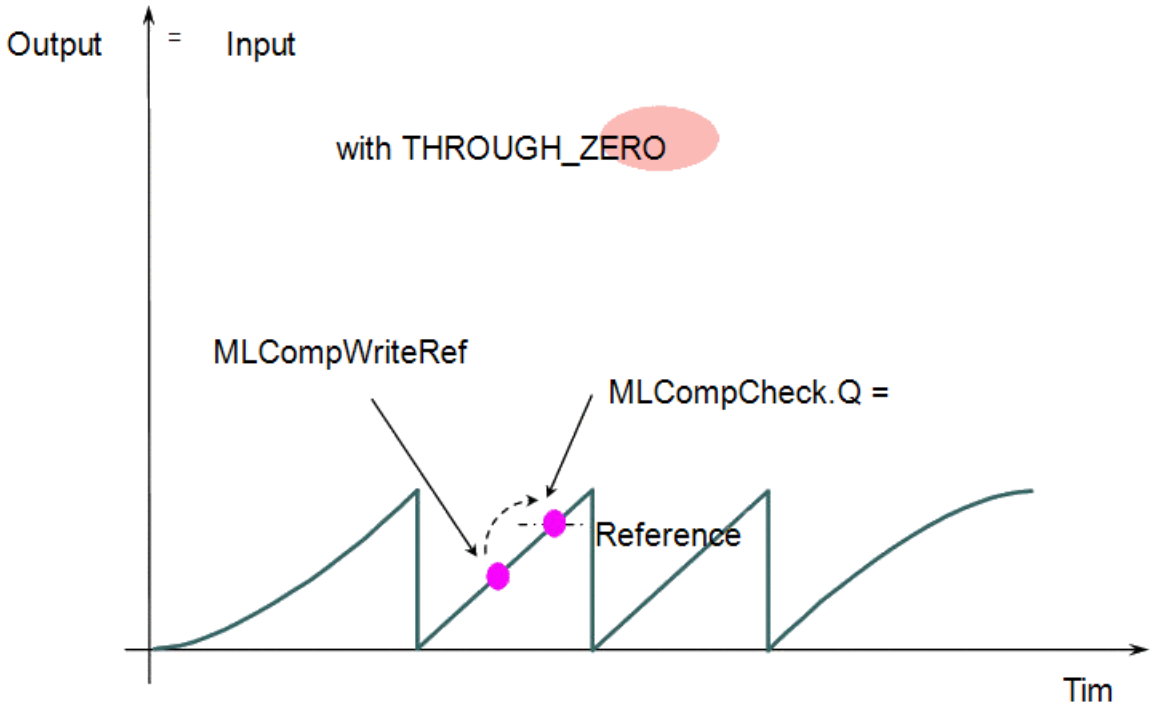
When you call the "MLCompWriteRef" (→ p. 300) function, the output for "MLCompCheck" (→ p. 295) becomes TRUE as soon as the input value reaches the reference.



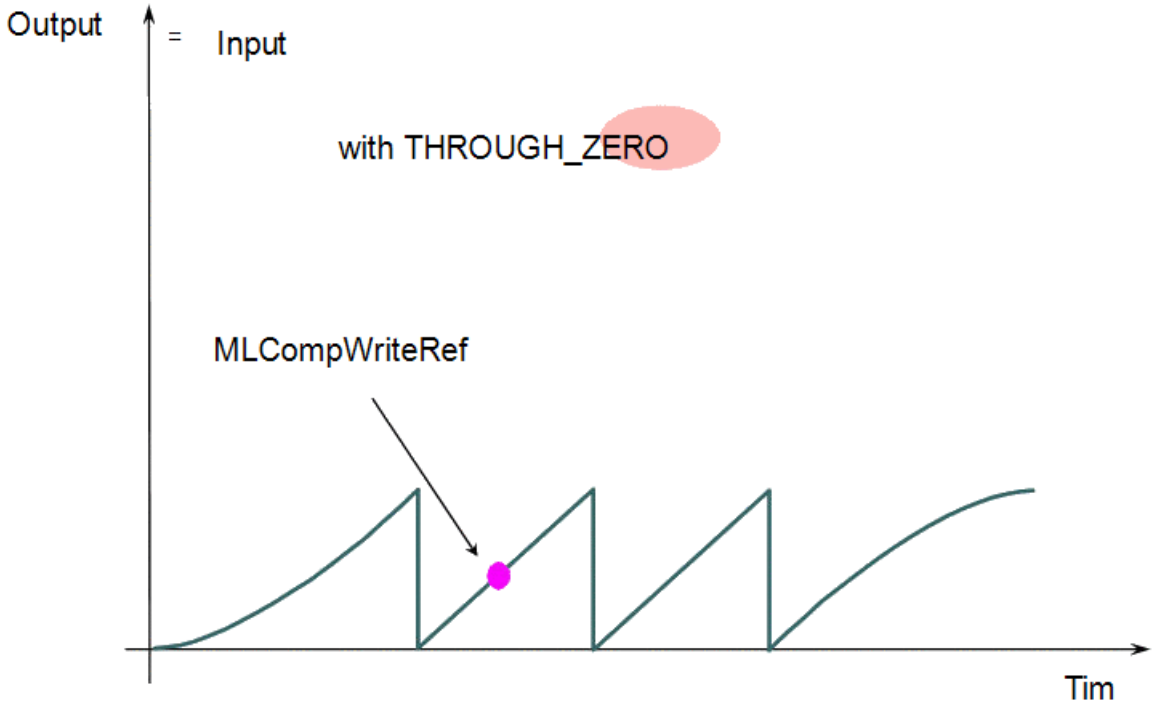


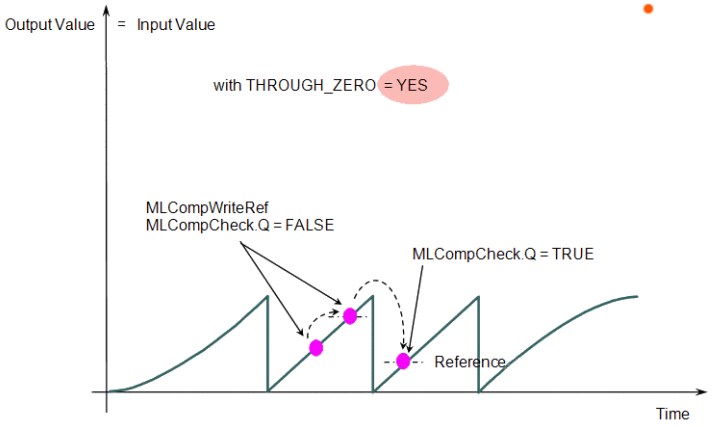
The same function can also be called for a cyclic input value.





When the THROUGH_ZERO parameter is set to YES, the output for "MLCompCheck" (→ p. 295) becomes TRUE as soon as the input value reaches the reference, but not before it has passed through zero.





4.1.6.2 MLCompCheck

Pipe Network

Function - Verifies if the reference of a comparator Pipe Block has been crossed.

4.1.6.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Comparator object.

4.1.6.2.2 Outputs

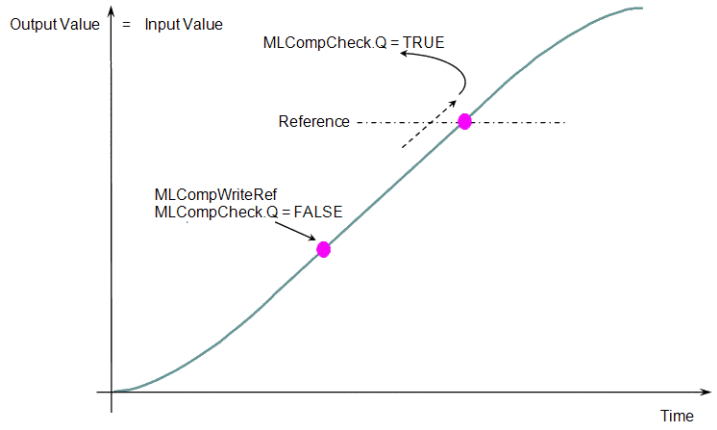
Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if reference position of the Comparator object has been crossed.

4.1.6.2.3 Remarks

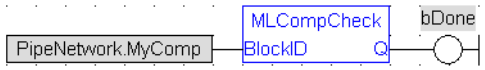
NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- Returns the Transition Flag of a comparator object, which turns TRUE if the input position to the comparator is greater or equal to the reference.
  - The Comparator Transition Flag stays TRUE until it is reset.



4.1.6.2.4 FBD Language Example



4.1.6.2.5 FFLD Language Example

4.1.6.2.6 IL Language Example

Not available.

4.1.6.2.7 ST Language Example

```
//Check if Comparator Reference has been reached
bCrossed := MLCompCheck( PipeNetwork.MyComp );
```

See Also

- "MLCompReadRef" (→ p. 298)
- "MLCompReset" (→ p. 299)
- "MLCompWriteRef" (→ p. 300)

4.1.6.3 MLCompInit

Pipe Network

Function - Initializes a comparator Pipe Block with user-defined settings for use in a PLC Program.

4.1.6.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of a created Comparator Pipe Block.
ModuloPosition	LREAL	No range	User units	No default	Value of the period of a cyclic system.
ThroughZero	BOOL	0, 1	N/A	No default	• When TRUE, the system must cross zero and then the reference position before the Transition Flag is set. • When FALSE, the Transition Flag is set immediately if the input pipe position is greater then or equal to the Reference value.
Reference	LREAL	No range	User units	No default	Set the reference position in the new Comparator object.

4.1.6.3.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when function starts to execute.

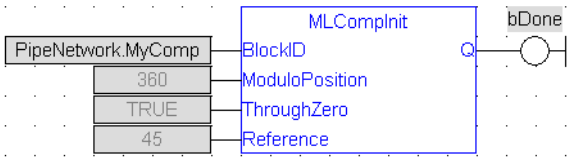
4.1.6.3.3 Remarks

- Function block is automatically called if a Comparator Block is added to the Pipe Network.
  - User-defined settings are entered in the Pipe Blocks Properties screen.
- The Transition Flag of a comparator object turns TRUE if the input position to the comparator is greater or equal to the reference.
- The Comparator Transition Flag stays TRUE until it is reset.
- If the input **ThroughZero** is set to TRUE, the system must cross zero and then the reference position before the Transition Flag is set.
- If **ThroughZero** is FALSE, the Transition Flag is set immediately if the input pipe position is greater or equal to the Reference value.

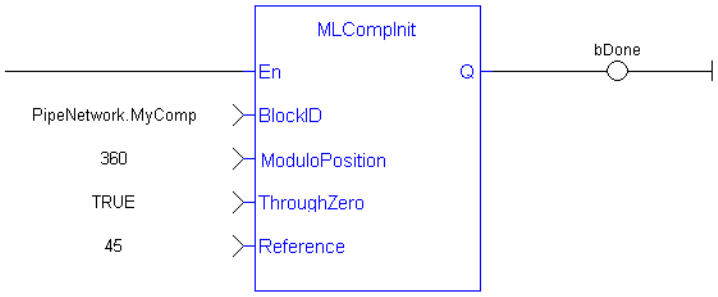
NOTE

Comparator objects are normally created in the Pipe Network using the graphical engine.
You do not have to add **MLCompInit** function blocks to their programs.
Parameters are entered directly in windows, and the code is then automatically added to the current project.

4.1.6.3.4 FBD Language Example



4.1.6.3.5 FFLD Language Example



4.1.6.3.6 IL Language Example

Not available.

4.1.6.3.7 ST Language Example

```
      //Initiate a created Comparator Block named "MyComp" to:
//   Modulo of 360
//   Require the input position to first cross 0 before the
//      MLCompCheck output is triggered
//   Input compared position to 45

MyComp := MLBlkCreate( 'MyComp', 'COMPARATOR' );
MLCompInit( MyComp, 360.0, TRUE, 45.0 );
```

See Also

- "MLBlkCreate" (→ p. 212)
- "MLCompCheck" (→ p. 295)
- "MLCompReset" (→ p. 299)
- "MLCompWriteRef" (→ p. 300)

4.1.6.4 MLCompReadRef

Pipe Network ✓

Function - Returns the reference position of a comparator block.

4.1.6.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Comparator object.

4.1.6.4.2 Outputs

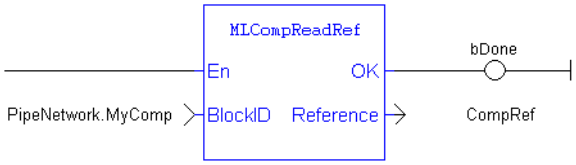
Output	Data Type	Range	Unit	Description
Reference	LREAL	No range	User units	Returns the current reference position of the Comparator object.

4.1.6.4.3 Remarks

- The Transition Flag of a comparator object turns TRUE if the input position to the comparator is greater or equal to the reference.
- The Comparator Transition Flag stays TRUE until it is reset.

4.1.6.4.4 FBD Language Example

4.1.6.4.5 FFLD Language Example



4.1.6.4.6 IL Language Example

Not available.

4.1.6.4.7 ST Language Example

```
//Return the Comparator Reference value
CompRef := MLCompReadRef( PipeNetwork.MyComp );
```

See Also

- "MLCompCheck" (→ p. 295)
- "MLCompReset" (→ p. 299)
- "MLCompWriteRef" (→ p. 300)

4.1.6.5 MLCompReset

Pipe Network

Function - Clears the Transition Flag of a comparator Pipe Block.

4.1.6.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Comparator object.

4.1.6.5.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when function starts to execute.

4.1.6.5.3 Remarks

- The Transition Flag of a comparator object turns TRUE if the input position to the comparator is greater or equal to the reference.
- The Comparator Transition Flag stays TRUE until it is reset.

4.1.6.5.4 FBD Language Example

4.1.6.5.5 FFLD Language Example

4.1.6.5.6 IL Language Example

Not available.

4.1.6.5.7 ST Language Example

```
//Clear the Transition Flag of a Comparator object
MLCompReset( PipeNetwork.MyComp );
```

See Also

- "MLCompCheck" (→ p. 295)
- "MLCompReadRef" (→ p. 298)
- "MLCompWriteRef" (→ p. 300)

4.1.6.6 MLCompWriteRef

Pipe Network

Function - Sets the reference position of a comparator block.

4.1.6.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Comparator object.
ThroughZero	BOOL	0, 1	N/A	No default	• When TRUE, the system must cross zero and then the reference position before the Transition Flag is set. • When FALSE, the Transition Flag is set immediately if the input pipe position is greater then or equal to the Reference value.
Reference	LREAL	No range	User units	No default	New reference position to set in the selected Comparator object.

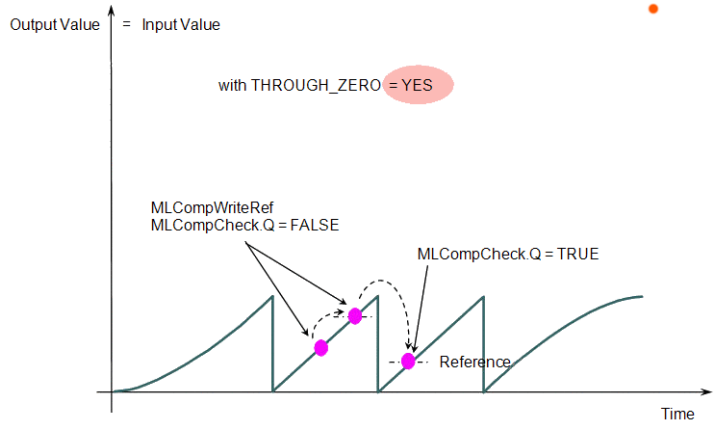
4.1.6.6.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when function starts to execute.

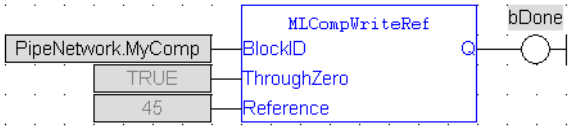
4.1.6.6.3 Remarks

- The Transition Flag of a comparator object turns TRUE if the input position to the comparator is greater or equal to the reference.
- The Comparator Transition Flag stays TRUE until it is reset.

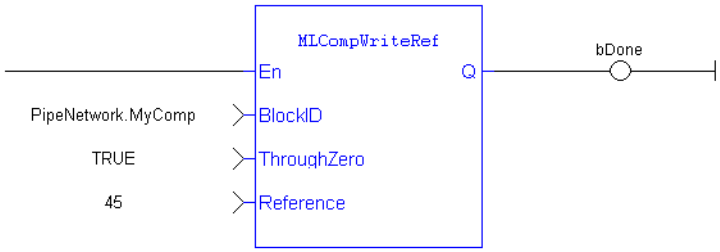
- If the input ThroughZero is set to TRUE, system must cross zero and then the reference position before the Transition Flag is set.
- If ThroughZero is FALSE, Transition Flag is set immediately if the input pipe position is greater then or equal to the Reference value.



4.1.6.6.4 FBD Language Example



4.1.6.6.5 FFLD Language Example



4.1.6.6.6 IL Language Example

Not available.

4.1.6.6.7 ST Language Example

```
//Set the Comparator Reference value
MLCompWriteRef( PipeNetwork.MyComp , TRUE , 45 );
```

See Also

- "MLCompCheck" (→ p. 295)
- "MLCompReadRef" (→ p. 298)
- "MLCompReset" (→ p. 299)

4.1.7 Convertor

Name	Description
MLCNVConECAT	Connects the output of a pipe convertor block to an EtherCAT Output (Rx) PDO object.

Name	Description
MLCNVConnect	Connects a converter Pipe Block to the specified axis.
MLCNVConnectEx	Connects the output of a pipe to an axis data other than the control position. Connects an extra converter Pipe Block to the specified axis.
MLCNVDisconnect	Disconnect a converter Pipe Block from its associated axis.
MLCNVInit	Initializes a converter Pipe Block.

4.1.7.1 MLCNVConECAT

Function - Connects the output of a pipe convertor block to an EtherCAT Output (Rx) PDO object.

4.1.7.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	No range	N/A	No default	The convertor block whose output value is written to the PDO object. Example: PipeNetwork:CNV1.
DeviceAddr	INT	No range	N/A	No default	The device address of the PDO object to be written. • EtherCAT devices are numbered in order. • The first device is 1001, the second 1002, etc.
Index	UINT	No range	N/A	No default	The index of the PDO object to be written. • The index can be determined from the table located in the PDO Selection/Mapping tab of the EtherCAT device page. • In Project Explorer, under EtherCAT, select the device, then select the PDO Selection/Mapping tab.
SubIndex	USINT	No range	N/A	No default	The sub-index of the PDO object to be written. • The sub-index can be determined from the table located in the PDO Selection/Mapping tab of the EtherCAT device page. • In Project Explorer, under EtherCAT, select the device, then select the PDO Selection/Mapping tab.

4.1.7.1.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if this function has successfully connected the output of the pipe convertor block to the EtherCAT Output (Rx) PDO Object.

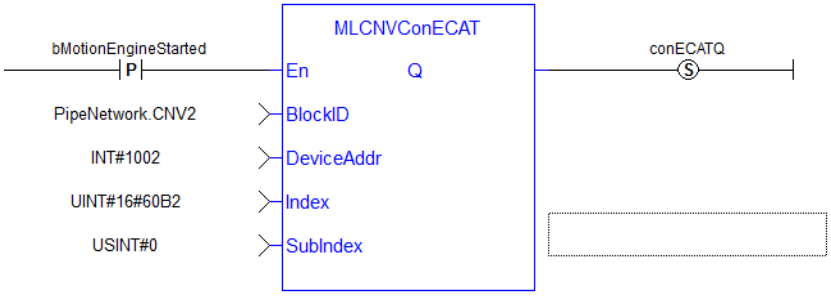
4.1.7.1.3 Remarks

- The output value of the convertor block is written to the PDO object every update of the convertor block.

- The pipe block is specified by the BlockID input.
- The PDO object is specified by the DeviceAddr, Index, and SubIndex inputs.

4.1.7.1.4 FBD Language Example

4.1.7.1.5 FFLD Language Example



4.1.7.1.6 IL Language Example

Not available.

4.1.7.1.7 ST Language Example

```
//Connect a converter Pipe Block named “CNV2”  to PDO 16#60B2 (Accel FF) on
ECAT address 1002.
MLCNVConECAT( PipeNetwork.CNV2, 1002, 16#60B2, 0 );
```

See Also

- "MLCNVDisconnect" (➔ p. 306)
- "MLCNVInit" (➔ p. 307)

4.1.7.2 MLCNVConnect

Function - Connects a converter Pipe Block to the specified axis.

4.1.7.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Converter object.
AxisID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Axis object.

4.1.7.2.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the converter is connected to the Axis object.

4.1.7.2.3 Remarks

- When using the Pipe Network for coordinated motion, Pipe Blocks have to be Activated, Connected, and then Powered On before move commands work.
- The Converter block changes the incoming flow of values to continuous position output with no periodicity.
  - If a converter block is not connected to an Axis, it does not send position output values to its assigned Axis.
  - Every pipe branch must end in a converter, whether or not it is connected to a destination Axis object.

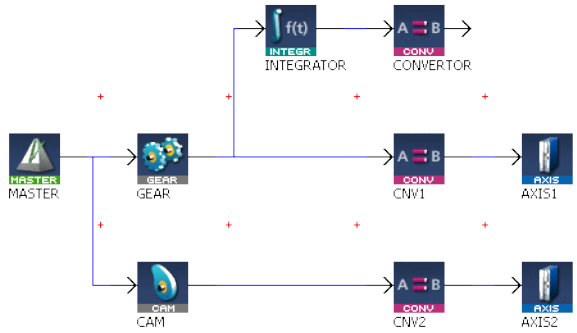


Figure 4-28: MLCNVConnect

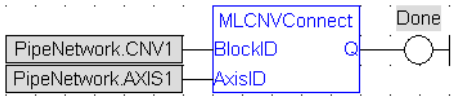
NOTE

All converters in the Pipe Network can be connected at once with the command Pipe Network(MLPN_Connect).
This calls automatically generated code with **MLCNVConnect** commands for each Converter block.
In a multi-axis program only one command can be used to connect Pipe Blocks instead of writing code for each Axis separately.

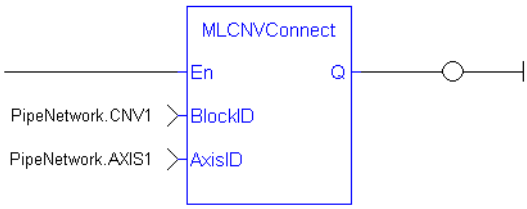
TIP

The converter block has the ability to control the analog output on the AKD.

4.1.7.2.4 FBD Language Example



4.1.7.2.5 FFLD Language Example



4.1.7.2.6 IL Language Example

Not available.

4.1.7.2.7 ST Language Example

```
//Connect a converter Pipe Block named "CNV1"  to Pipe Block AXIS1
MLCNVConnect( PipeNetwork.CNV1, AXIS1 );
```

See Also

- "MLCNVConnectEx" (→ p. 305)
- "MLCNVDisconnect" (→ p. 306)
- "MLCNVInit" (→ p. 307)

4.1.7.3 MLCNVConnectEx

Function - Connects the output of a pipe to an axis data other than the control position.

4.1.7.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Converter object.
AxisID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Axis object.

ValueID	DINT	-2147483648 to 2147483647	N/A	No default	Specify this constant:
					• EC_ADDITIVE_TORQUE_VALUE
					  • For torque feed-forward.
					• EC_ANALOG_OUTPUT
					  • For control of Analog Output: AKD parameter: AOUT.VALUEU.
					  • See Example: Pre-defined Constants for more information.
					• If the Analog Output is mapped to a PLC variable, the connection to the analog output by EC_ANALOG_OUTPUT does not work because the output value is overwritten by the PLC mapped variable data.
					• To function properly, the AOUT.MODE must be set to User Mode (mode = 0).

ValueInfo	DINT	-2147483648 to 2147483647	N/A	No default	This value is ignored. It must be set to 0 (zero).

4.1.7.3.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the converter is connected to the Axis object.

4.1.7.3.3 Remarks

- This function:
  - Connects an extra converter Pipe Block to the specified axis.
  - Connects several converter Pipe Blocks to the same axis and acts on different data.
- Normally a Converter block sends position values to an Axis.
  - However, some cases exist that require additional information (e.g., torque feed-forward (IDN 3056)) that needs to be provided by a second converter.

NOTE

This FB does NOT work if you have Simulated the device.
In this instance, the FB continuously generates error messages in the Controller log window.

IMPORTANT

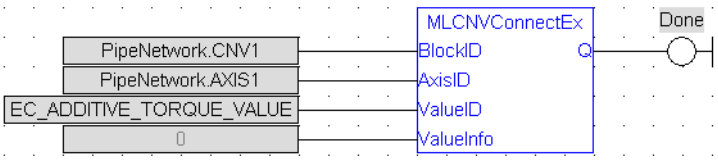
Add 16#8000 to the designated IDN number for ValueID input. 8000 in hexadecimal signals a vendor-specific IDN value.

- The PDO values are overwritten by Mapped PLC variables including either:
  - A possible link to the mapping of variables.
  - The section on **MLParamWrite()** warning indicating that the function block write of Analog output is overwritten by the **MLCnvConnectEx** function.

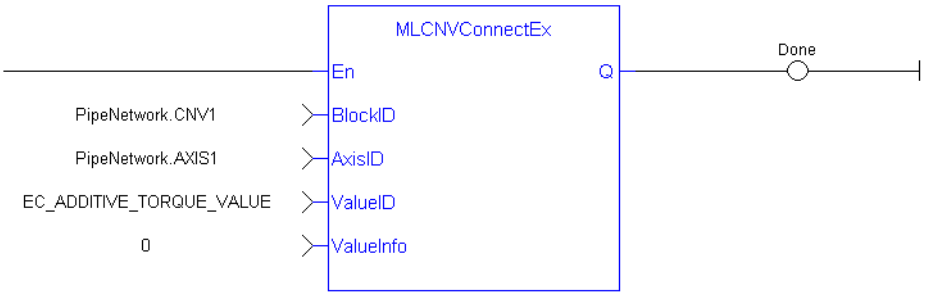
4.1.7.3.3.1 Precedence Rules

1. A PLC variable mapped to Analog Output takes precedence.
2. If **MLCNVConnect** assigns a Pipe output to an Analog Output, it takes precedence over a **DriveParamWrite** function call.
3. **DriveParamWrite** modifies the Analog Output but is overwritten by the higher precedent options if they are present.

4.1.7.3.4 FBD Language Example



4.1.7.3.5 FFLD Language Example



4.1.7.3.6 IL Language Example

Not available.

4.1.7.3.7 ST Language Example

```
//Connect a converter Pipe Block named "CNV1"  to the pipe block named AXIS1,
And send feed-forward (EC_ADDITIVE_TORQUE_VALUE) to the drive
MLCNVConnectEx( PipeNetwork.CNV1, PipeNetwork.AXIS1, EC_ADDITIVE_TORQUE_
VALUE, 0 );
```

See Also

- "MLCNVConnect" (→ p. 303)
- "MLCNVDisconnect" (→ p. 306)
- "MLCNVInit" (→ p. 307)

4.1.7.4 MLCNVDisconnect

Function - Disconnect a converter Pipe Block from its associated axis.

4.1.7.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Converter object.

4.1.7.4.2 Outputs

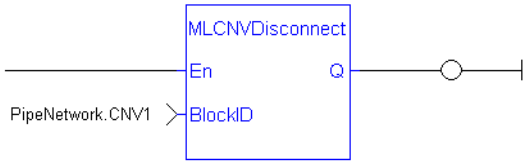
Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the converter is disconnected from the Axis object.

4.1.7.4.3 Remarks

- If a converter block is not connected to an Axis, it does not send position output values to its assigned Axis.
  - Can disconnect one or multiple Axis from the Pipe Network and still send single-axis motion commands.
- Axis can be disconnected while the Pipe Positions are reset to different values or if coordinated motion is only not needed with every axis in the project in a certain state.

4.1.7.4.4 FBD Language Example

4.1.7.4.5 FFLD Language Example



4.1.7.4.6 IL Language Example

Not available.

4.1.7.4.7 ST Language Example

```
//Disconnect a converter Pipe Block  name " CNV1" from its present connection
MLCNVDisconnect( PipeNetwork.CNV1);
```

See Also

- "MLCNVConnect" (→ p. 303)
- "MLCNVInit" (→ p. 307)

4.1.7.5 MLCNVInit

Function - Initializes a converter Pipe Block.

4.1.7.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of a created Pipe Block.
Mode	DINT	1, 2	N/A	No default	Determines the type of input to the Converter Object. • 1 for Position mode. • 2 for Speed mode.

4.1.7.5.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the Convertor Pipe Block is initialized.

4.1.7.5.3 Remarks

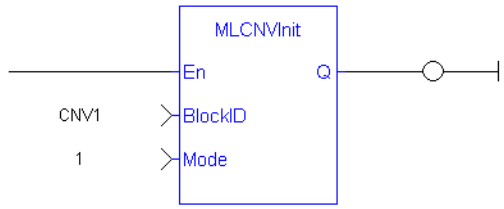
- **MLCNVInit** is automatically called if a Convertor Block is added to the Pipe Network, with the input mode (position or speed) entered in the Pipe Blocks Properties screen.
- The Converter block changes the incoming flow of speed or position values to continuous position output with no periodicity.

NOTE

Converter objects are normally created in the Pipe Network using the graphical engine.
You do not have to add MLCNVInit function blocks to their programs.
Parameters are entered directly in pop-up windows and the code is then automatically added to the current project.

4.1.7.5.4 FBD Language Example

4.1.7.5.5 FFLD Language Example



4.1.7.5.6 IL Language Example

Not available.

4.1.7.5.7 ST Language Example

```
// Initiate a created convertor block named "CNV1"
CNV1 := MLBlkCreate( 'CNV1', 'CONVERTOR' );
MLCNVInit( CNV1, 1 );
```

See Also

- "MLBlkCreate" (→ p. 212)
- "MLCNVConnect" (→ p. 303)

4.1.8 Delay

Name	Description
MLDelayInit	Initializes a delay object.

4.1.8.1 MLDelayInit

Pipe Network

Function - Initializes a delay object.

4.1.8.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of a created Pipe Block.
CycleDelay	DINT	0, 9	Cycle	0 (zero))	Number of delay cycles.

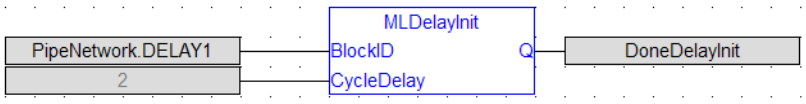
4.1.8.1.2 Outputs

Output	Data Type	Range	Unit	Description

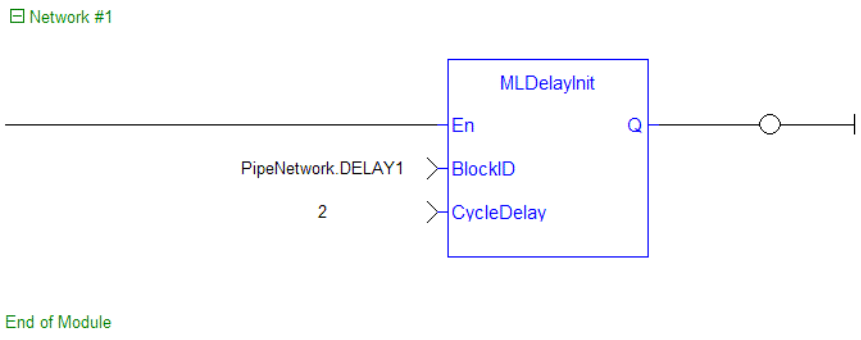
4.1.8.1.3 Remarks

- This is automatically created in the compiled code of a Pipe Network.
  - It is included in the MLPN_CREATE_OBJECT (created in ST) which is typically executed in a project as part of the startup sequence of the Pipe Network.
- Returns TRUE if the function succeeded.

4.1.8.1.4 FBD Language Example



4.1.8.1.5 FFLD Language Example



4.1.8.1.6 IL Language Example

Not available.

4.1.8.1.7 ST Language Example

```
MLDelayInit(PipeNetwork.DELAY1, 2 );
```

4.1.9 Derivator

Name	Description
MLDerInit	Initializes an derivator object.
MLDerReadInModPos	Returns the Input ModuloPosition of the derivator block.
MLDerWriteInModPos	Sets the Input ModuloPosition of the Derivator block.

4.1.9.1 MLDerInit

Function - Initializes an derivator object.

4.1.9.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of a created Pipe Block.
ModuloPosition	LREAL	No range	User units	360.0	Input ModuloPosition of Derivator object.

4.1.9.1.2 Outputs

Input	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the Derivator object is initialized.

4.1.9.1.3 Remarks

- Function block is automatically called if a Derivator Block is added to the Pipe Network, with user-defined settings entered in the Pipe Blocks Properties screen.
- Input ModuloPosition is defined to manage the periodicity (modulo) of the input values.

NOTE

Derivator objects are normally created in the Pipe Network using the graphical engine.
You do not have to add MLDerInit function blocks to their programs.
Parameters are entered directly in pop-up windows and the code is automatically added to the current project.

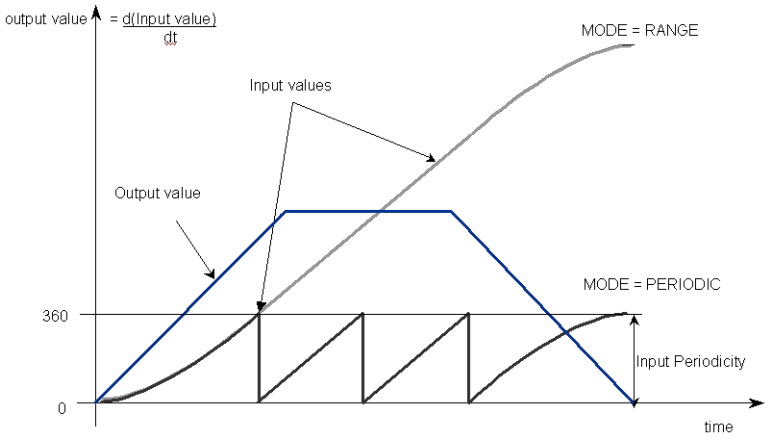
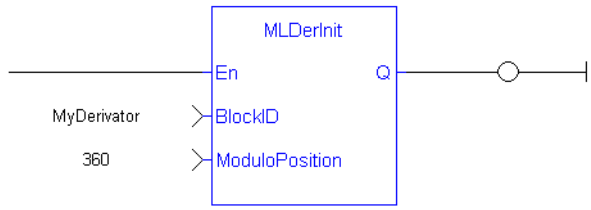


Figure 4-29: MLDerInit

4.1.9.1.4 FBD Language Example

4.1.9.1.5 FFLD Language Example



4.1.9.1.6 IL Language Example

Not available.

4.1.9.1.7 ST Language Example

```
//Create and Initiate a Derivator object
MyDerivator := MLBlkCreate( 'MyDerivator', 'DERIVATOR' );
MLDerInit( MyDerivator, 360.0 );
```

See Also

- "MLBlkCreate" (→ p. 212)
- "MLDerReadInModPos" (→ p. 311)
- "MLDerReadInModPos" (→ p. 311)

4.1.9.2 MLDerReadInModPos

Pipe Network

Function - Returns the Input ModuloPosition of the derivator block.

4.1.9.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Derivator object.

4.1.9.2.2 Outputs

Output	Data Type	Range	Unit	Description
ModuloPosition	LREAL			Current Input ModuloPosition of the selected Derivator object.

4.1.9.2.3 Remarks

Input ModuloPosition is defined to manage the periodicity (modulo) of the input values.

Example

If the input value increases each millisecond by one degree then the output value is 1000 degrees per second.

Suddenly, the input value skips from 359 to 0 (zero).

- Input ModuloPosition = 360, the output continues to indicate 1000 degrees per second.
  - This indicates that rollover into the next period has been properly handled
- Input ModuloPosition = 1000, the output then indicates 359,000 degrees per second.
  - This indicates that the input has incorrectly interpreted roll-over as a 359 degree move in one millisecond.

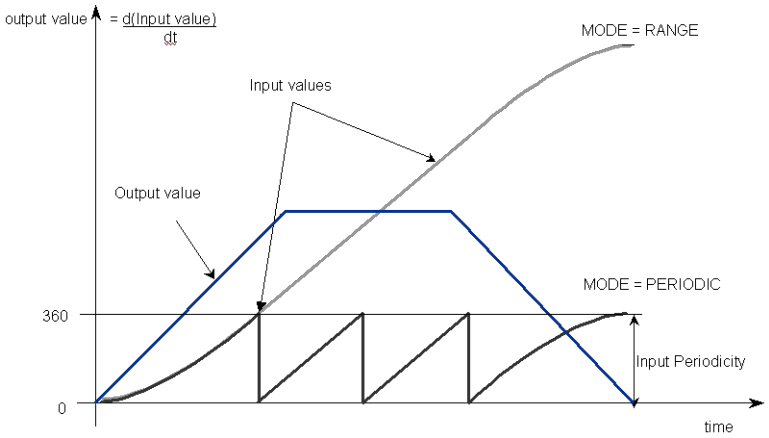


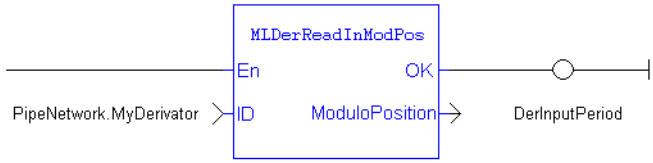
Figure 4-30: MLDerReadInModPos

NOTE

The first calculation of a Derivator Pipe Block just after the pipe installation indicates 0 (zero) regardless of the initial input value.

4.1.9.2.4 FBD Language Example

4.1.9.2.5 FFLD Language Example



4.1.9.2.6 IL Language Example

Not available.

4.1.9.2.7 ST Language Example

```
//save the current input MODULO_POSITION of a Derivator object
DerInputPeriod := MLDerReadInModPos ( PipeNetwork.MyDerivator );
```

See Also

- "MLDerInit" (→ p. 310)
- "MLDerWriteInModPos" (→ p. 313)

4.1.9.3 MLDerWriteInModPos

Function - Sets the Input ModuloPosition of the Derivator block.

4.1.9.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Derivator object.
ModuloPosition	LREAL	No range	User units	No default	Designated new value of Input ModuloPosition of the selected Derivator object.

4.1.9.3.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the Input ModuloPosition value is changed.

4.1.9.3.3 Remarks

Input ModuloPosition is defined to manage the periodicity (modulo) of the input values.

Example

If the input value increases each millisecond by one degree then the output value is 1000 degrees per second.

Suddenly, the input value skips from 359 to 0 (zero).

- Input ModuloPosition = 360, the output continues to indicate 1000 degrees per second.
  - This indicates that rollover into the next period has been properly handled
- Input ModuloPosition = 1000, the output then indicates 359,000 degrees per second.
  - This indicates that the input has incorrectly interpreted roll-over as a 359 degree move in one millisecond.

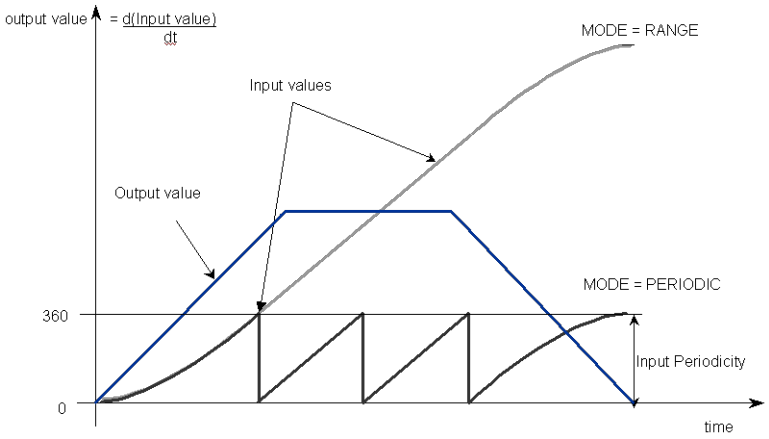
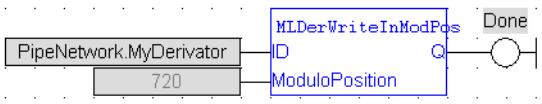


Figure 4-31: MLDerWriteInModPos

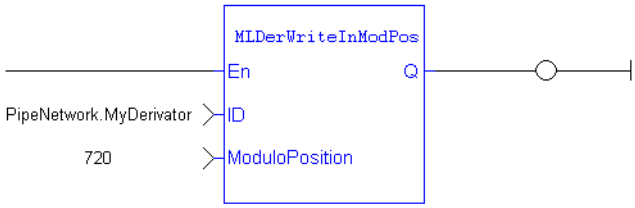
NOTE

The first calculation of a Derivator Pipe Block just after the pipe installation indicates 0 (zero) regardless of the initial input value.

4.1.9.3.4 FBD Language Example



4.1.9.3.5 FFLD Language Example



4.1.9.3.6 IL Language Example

Not available.

4.1.9.3.7 ST Language Example

```
//change the input MODULO_POSITION of a Derivator object to 720
MLDerWriteInModPos ( PipeNetwork.MyDerivator, 720 );
```

See Also

- "MLDerInit" (→ p. 310)
- "MLDerReadInModPos" (→ p. 311)

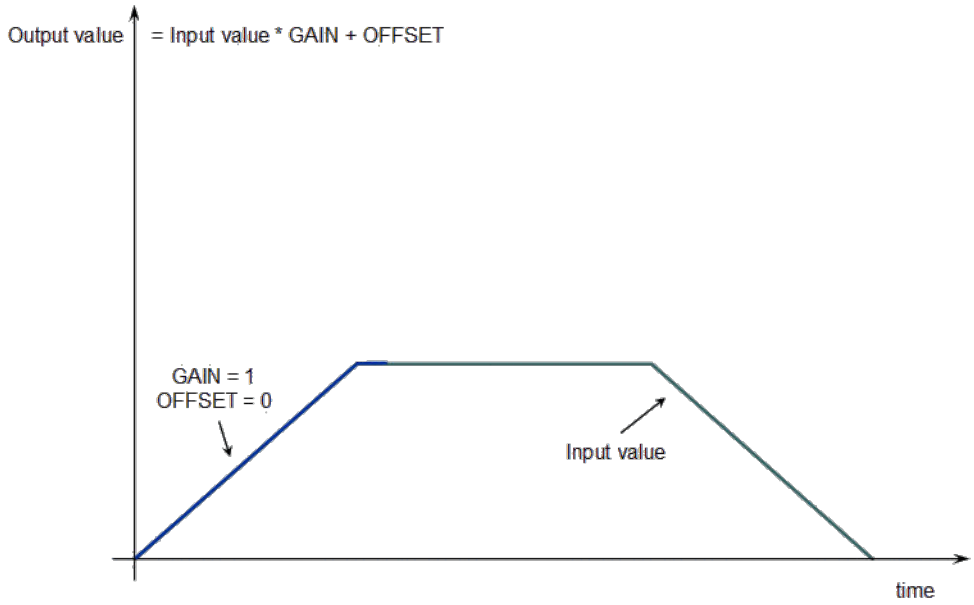
4.1.10 Gear

Name	Description
MLGearInit	Initializes a Gear Pipe Block with user-defined settings and for use in a PLC program.
MLGearReadOffset	Returns the Offset value of a selected Gear Block from the Pipe Network.

Name	Description
MLGearReadOffSlp	Returns the Offset Slope value of a selected Gear Block from the Pipe Network.
MLGearReadRatio	Returns the Ratio value of a selected Gear Block from the Pipe Network.
MLGearReadRatSlp	Returns the Ratio Slope value of a selected Gear Block from the Pipe Network.
MLGearWriteOff	Sets the Offset value of a selected Gear Pipe Block.
MLGearWriteOSlp	Sets the Offset Slope value of a selected Gear Pipe Block.
MLGearWriteRatio	Sets the Ratio value of a selected Gear Pipe Block.
MLGearWriteRatSlp	Sets the Ratio Slope value of a selected Gear Pipe Block.

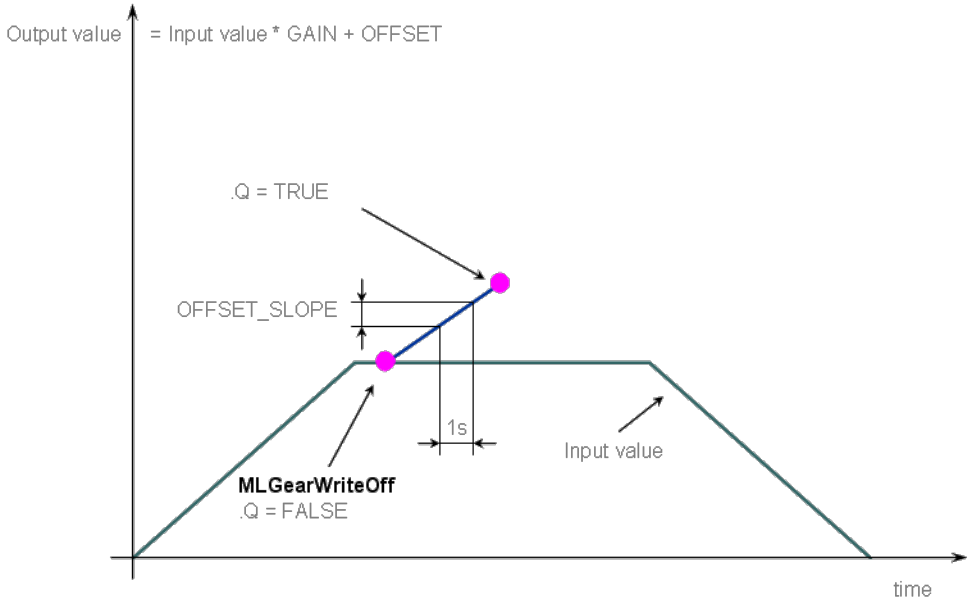
4.1.10.1 Examples of Gear Functions

The output value starts with offset = 0 and gain = 1 (blue line)



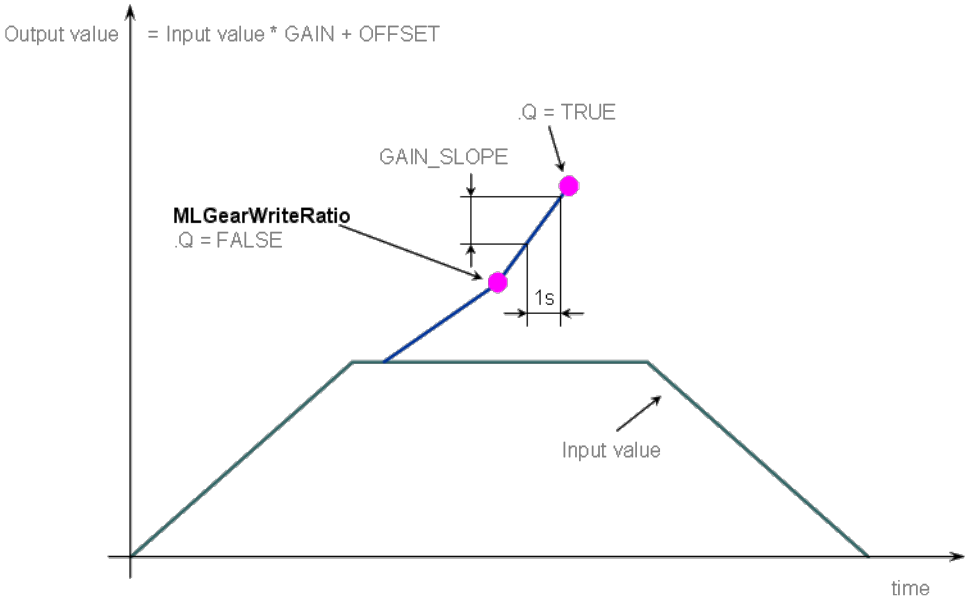
Call the **MLGearWriteOff** function to modify the Offset.

This is where Offset_Slope is set with the **MLGearWriteOSlp** function.

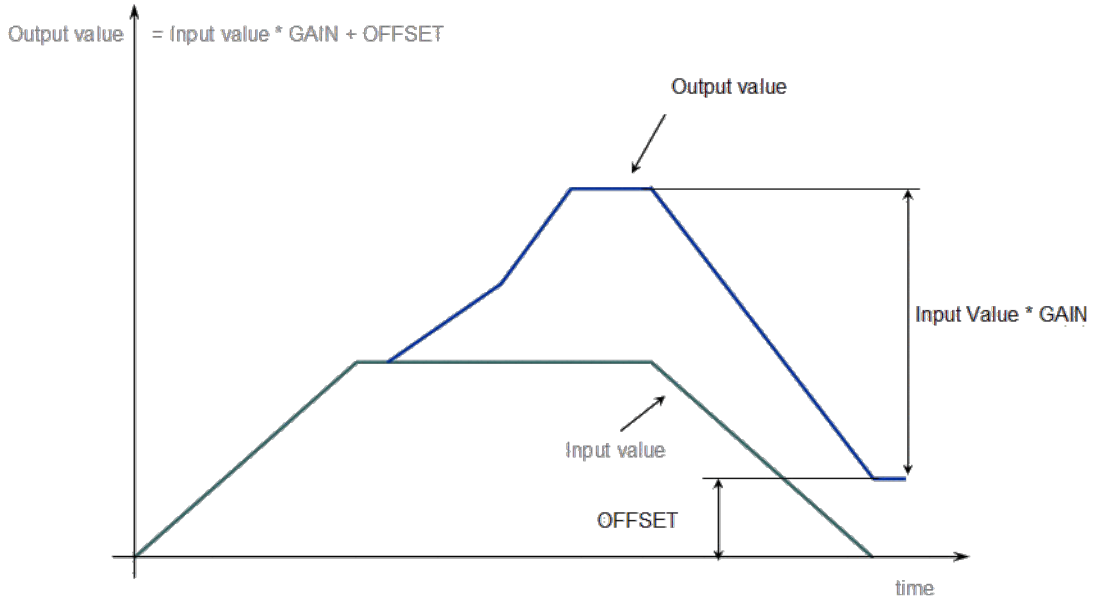


After setting the Offset (Q=TRUE in the previous figure), call the **MLGearWriteRatio** function to modify the gear Ratio.

This is where Gain_Slope is set with the **MLGearWriteRatSlp** function.



The output value is adapted with the gear offset and ratio (blue line).



Gear Functions Usage

4.1.10.2 MLGearInit

Pipe Network

Function - Initializes a Gear Pipe Block with user-defined settings and for use in a PLC program.

4.1.10.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	GEAR	ID number of a created Pipe Block.
Ratio	LREAL	No range	N/A	1.0	Ratio of new Gear Pipe Block. • Values lower then 1.0 can be entered, but require a leading 0 (zero). • Example: 0.8 instead of .8.
Offset	LREAL	No range	N/A	0.0	Offset of new Gear Pipe Block. • Values lower then 1.0 can be entered, but require a leading 0 (zero). • Example: 0.8 instead of .8.
UseUserRatioSlope	BOOL	0, 1	N/A	FALSE	FALSE to use the maximum Slope. • FALSE causes an instantaneous gear change within one cycle. • Use TRUE to use user-defined RatioSlope.

Input	Data Type	Range	Unit	Default	Description
RatioSlope	LREAL	No range	1/sec	0.0	User-defined limit at which step changes in Ratio are implemented. • Values lower then 1.0 can be entered, but require a leading 0 (zero). • Example: 0.8 instead of .8.
UseUserOffsetSlope	BOOL	0, 1	N/A	FALSE	FALSE to use the maximum Slope. • FALSE causes an instantaneous gear change within one cycle. • Use TRUE to use user-defined OffsetSlope.
OffsetSlope	LREAL	No range	User unit/sec	0.0	User-defined limit at which step changes in Offset are implemented. • Values lower then 1.0 can be entered, but require a leading 0 (zero). • Example: 0.8 instead of .8.
Modulo	BOOL	0, 1	N/A	FALSE	TRUE when the mode is modulo. Modulo mode adapts the output values according to the ModuloPosition (modulo).

4.1.10.2.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the Gear Pipe Block is initialized. • The output of a Gear Block = Input value * Ratio + Offset. •

4.1.10.2.3 Remarks

- This function block is automatically called if a Gear Block is added to the Pipe Network, with user-defined settings entered in the **Pipe Blocks Properties** screen.
- The Pipe Block is assigned a **Name**, **Ratio**, **Offset**, and **Slopes** for changes in Ratio and Offset values.
  - Choose between Modulo or Not modulo mode.
  - Slopes set the limit at which step changes in Ratio and Offset are implemented.

IMPORTANT

Set the `RatioSlope < (Ratio * EtherCAT Update Rate).`
The Gear block makes a jump (without a ramp) from one gear to the next when the RatioSlope is greater than the Ratio change factor multiplied by the update rate scale factor.

4.1.10.2.3.1 Modulo Value

If the Gear block's input is a modulo value and the position delta is greater than ½, the modulo value within one sample period in the opposite direction, the Gear block cannot detect the change in the direction of motion.

Example

- The sample period is: 1 msec.
- The Master is configured for a 360 degree modulo.
- The Master position is changed by >180 degrees within 1 msec.
  - In this case, the Gear block cannot determine whether the direction is in the same or opposite direction.

4.1.10.3 Avoid Modulo Calculation Problems

- Either deactivate and reactivate the PipeNetwork when forcing the Master position with "MLMstForcePos" (→ p. 337).
- Use a "MLMstAbs" (→ p. 332) or "MLMstRel" (→ p. 345) move to force the Master's position value.

Example

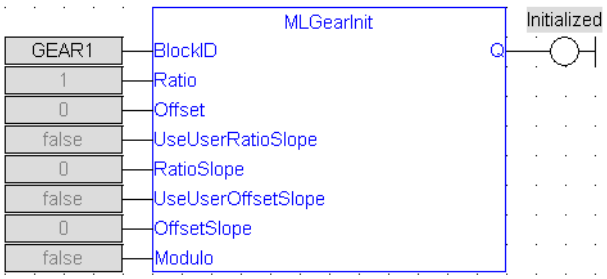
To force the Master position to 0 (zero), use this code:

```
PipeNetwork(MLPN_DEACTIVATE);
MLMstForcePos(PipeNetwork.MASTER, 0);
PipeNetwork(MLPN_ACTIVATE);
```

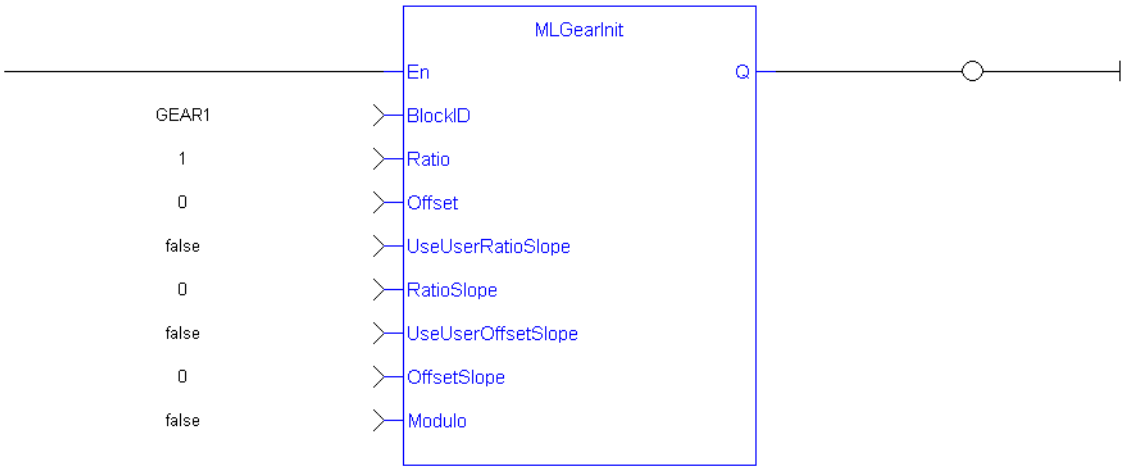
TIP

Gear objects are normally created in the Pipe Network using the graphical engine.
You do not have to add **MLGearInit** function blocks to their programs.
Parameters are entered directly in pop-up windows and the code is automatically added to the current project.

4.1.10.3.1 FBD Language Example



4.1.10.3.2 FFLD Language Example



4.1.10.3.3 IL Language Example

Not available.

4.1.10.3.4 ST Language Example

```
//Initialize a Gear Pipe Block named GEAR1 with:
//  Ratio = 1,Offset = 0, User Ratio Slope OFF, User Ratio
//  Slope = 0, Offset Slope = 0, and no Modulo
GEAR1 := MLBlkCreate( 'GEAR1', 'GEAR' );
MLGearInit( GEAR1, 1.0, 0.0, false, 0.0, false, 0.0, false);
```

See Also

- "MLBlkCreate" (→ p. 212)
- "MLGearWriteRatio" (→ p. 325)

4.1.10.4 MLGearReadOffset

Pipe Network ✓

Function - Returns the Offset value of a selected Gear Block from the Pipe Network.

4.1.10.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized Gear object.

4.1.10.4.2 Outputs

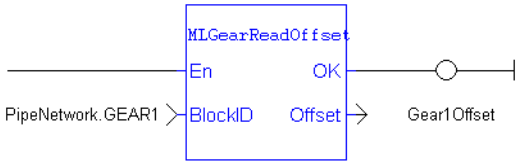
Output	Data Type	Range	Unit	Description
Offset	LREAL		User units	The offset value currently assigned to the selected Gear Pipe Block.

4.1.10.4.3 Remarks

The output of a Gear Block = Input value * Ratio + Offset.

4.1.10.4.4 FBD Language Example

4.1.10.4.5 FFLD Language Example



4.1.10.4.6 IL Language Example

Not available.

4.1.10.4.7 ST Language Example

```
//Find the Offset value of Gear1 Pipe Block
Gear1Offset := MLGearReadOffset( PipeNetwork.GEAR1 );
```

See Also

- "MLGearInit" (→ p. 317)
- "MLGearWriteOff" (→ p. 323)

4.1.10.5 MLGearReadOffSlp

Pipe Network

Function - Returns the Offset Slope value of a selected Gear Block from the Pipe Network.

4.1.10.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized Gear object.

4.1.10.5.2 Outputs

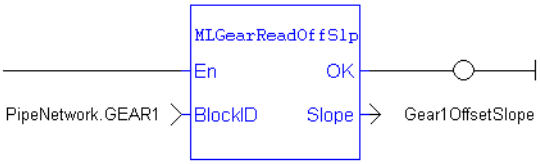
Output	Data Type	Range	Unit	Description
Slope	LREAL		User unit/sec	The Offset Slope value currently assigned to the selected Gear Pipe Block. This can be a different sign than what is programmed with **MLGearWriteOSlp**.

4.1.10.5.3 Remarks

- Offset Slope sets the limit in User Units per Second at which step changes in offset are implemented.
- The default value when creating a Gear Block is OFFSET_SLOPE_MAX or infinite.

4.1.10.5.4 FBD Language Example

4.1.10.5.5 FFLD Language Example



4.1.10.5.6 IL Language Example

Not available.

4.1.10.5.7 ST Language Example

```
//Find the Offset Slope value of Gear1 Pipe Block
Gear1OffsetSlope := MLGearReadOffSlp(PipeNetwork.GEAR1);
```

See Also

- "MLGearInit" (→ p. 317)
- "MLGearWriteOSlp" (→ p. 324)

4.1.10.6 MLGearReadRatio

Pipe Network

Function - Returns the Ratio value of a selected Gear Block from the Pipe Network.

4.1.10.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized Gear object.

4.1.10.6.2 Outputs

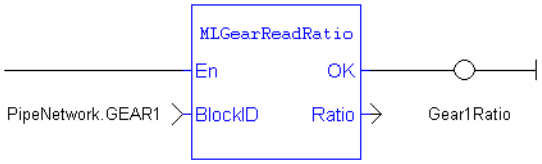
Output	Data Type	Range	Unit	Description
Ratio	LREAL	No range	N/A	The Ratio value currently assigned to the selected Gear Pipe Block.

4.1.10.6.3 Remarks

The output of a Gear Block = Input value * Ratio + Offset.

4.1.10.6.4 FBD Language Example

4.1.10.6.5 FFLD Language Example



4.1.10.6.6 IL Language Example

Not available.

4.1.10.6.7 ST Language Example

```
//Find the Ratio value of Gear1 Pipe Block
Gear1Ratio := MLGearReadRatio(PipeNetwork.GEAR1);
```

See Also

- "MLGearInit" (→ p. 317)
- "MLGearWriteRatio" (→ p. 325)

4.1.10.7 MLGearReadRatSlp

Pipe Network

Function - Returns the Ratio Slope value of a selected Gear Block from the Pipe Network.

4.1.10.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized Gear object.

4.1.10.7.2 Outputs

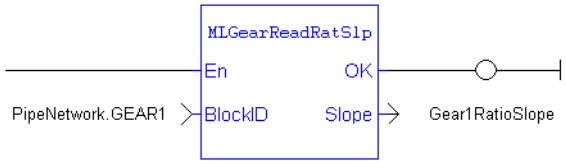
Output	Data Type	Range	Unit	Description
Slope	LREAL		1/sec (or s^{-1})	The Ratio Slope value currently assigned to the selected Gear Pipe Block. This can be a different sign than what is programmed with **MLGearWriteRatSlp**.

4.1.10.7.3 Remarks

The Ratio Slope sets the limit in 1/Seconds (or s^{-1}) at which step changes in Ratio are implemented.

4.1.10.7.4 FBD Language Example

4.1.10.7.5 FFLD Language Example



4.1.10.7.6 IL Language Example

Not available.

4.1.10.7.7 ST Language Example

```
//Find the Ratio Slope value of Gear1 Pipe Block
Gear1RatioSlope := MLGearReadRatSlp(PipeNetwork.GEAR1);
```

See Also

- "MLGearInit" (→ p. 317)
- "MLGearWriteRatSlp" (→ p. 326)

4.1.10.8 MLGearWriteOff

Pipe Network

Function - Sets the Offset value of a selected Gear Pipe Block.

4.1.10.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized Gear Pipe Block.

Input	Data Type	Range	Unit	Default	Description
Offset	LREAL	No range	User units	No default	New Offset value to be assigned to the selected Gear Pipe Block. • Values lower then 1.0 can be entered, but require a leading 0 (zero). • Example: 0.8 instead of .8.

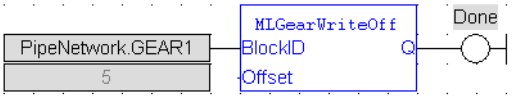
4.1.10.8.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE if the Offset value is changed in the selected Gear Pipe Block. • The output of a Gear Block = Input value * Ratio + Offset. •

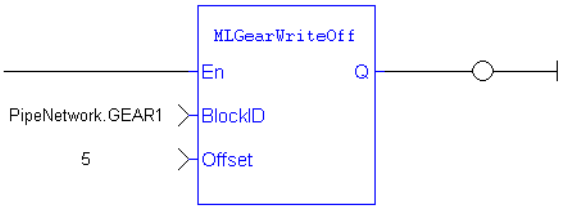
4.1.10.8.3 Remarks

None

4.1.10.8.4 FBD Language Example



4.1.10.8.5 FFLD Language Example



4.1.10.8.6 IL Language Example

Not available.

4.1.10.8.7 ST Language Example

```
//Set the Offset value of Gear1 Pipe Block to 5 User Units
MLGearWriteOff(PipeNetwork.GEAR1, 5.0);
```

See Also

- "MLGearInit" (→ p. 317)
- "MLGearReadOffset" (→ p. 320)

4.1.10.9 MLGearWriteOSlp

Pipe Network

Function - Sets the Offset Slope value of a selected Gear Pipe Block.

4.1.10.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized Gear Pipe Block.
Slope	LREAL	No range	User unit/sec	No default	New Offset Slope value to be assigned to the selected Gear Pipe Block. • Values lower then 1.0 can be entered, but require a leading 0 (zero). • Example: 0.8 instead of .8.

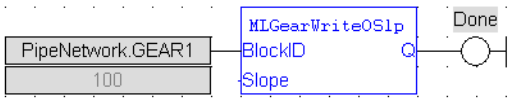
4.1.10.9.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE if the Offset Slope value is changed in the selected Gear Pipe Block.

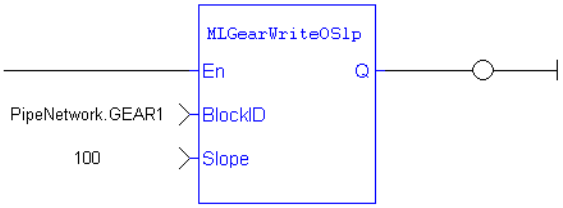
4.1.10.9.3 Remarks

- Offset Slope sets the limit in User Units per Second at which step changes in offset are implemented.
- The default value when creating a Gear Block is OFFSET_SLOPE_MAX or infinite.

4.1.10.9.4 FBD Language Example



4.1.10.9.5 FFLD Language Example



4.1.10.9.6 IL Language Example

Not available.

4.1.10.9.7 ST Language Example

```
//Set the Offset Slope value of Gear1 Pipe Block to 100
MLGearWriteOSlp(PipeNetwork.GEAR1, 100.0);
```

See Also

- "MLGearInit" (→ p. 317)
- "MLGearReadOffSlp" (→ p. 321)

4.1.10.10 MLGearWriteRatio

Pipe Network

Function - Sets the Ratio value of a selected Gear Pipe Block.

4.1.10.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized Gear Pipe Block.
Ratio	LREAL	No range	User unit/sec	No default	New Ratio value to be assigned to the selected Gear Pipe Block. • Values lower then 1.0 can be entered, but require a leading 0 (zero). • Example: 0.8 instead of .8.

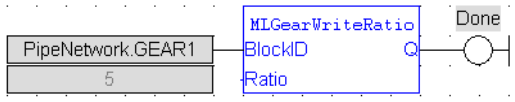
4.1.10.10.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE if the Ratio value is changed in the selected Gear Pipe Block.

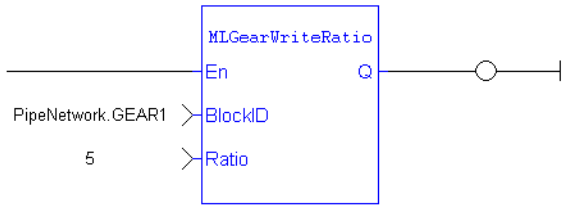
4.1.10.10.3 Remarks

The output of a Gear Block = Input value * Ratio + Offset.

4.1.10.10.4 FBD Language Example



4.1.10.10.5 FFLD Language Example



4.1.10.10.6 IL Language Example

Not available.

4.1.10.10.7 ST Language Example

```
//Set the Ratio value of Gear1 Pipe Block to 5
MLGearWriteRatio(PipeNetwork.GEAR1, 5.0);
```

See Also

- "MLGearInit" (→ p. 317)
- "MLGearReadRatio" (→ p. 321)

4.1.10.11 MLGearWriteRatSlp

Pipe Network

Function - Sets the Ratio Slope value of a selected Gear Pipe Block.

4.1.10.11.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized Gear Pipe Block.
Slope	LREAL	No range	1/sec	No default	New Ratio value to be assigned to the selected Gear Pipe Block. • Values lower then 1.0 can be entered, but require a leading 0 (zero). • Example: 0.8 instead of .8.

4.1.10.11.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the Ratio Slope value is changed in the selected Gear Pipe Block.

4.1.10.11.3 Remarks

- Ratio Slope sets the limit at which step changes in ratio are implemented.
- The GEAR block output adds a position offset to the GEAR block input when using a RatioSlope.
- See "RatioSlope Offset" (→ p. 327) for examples.

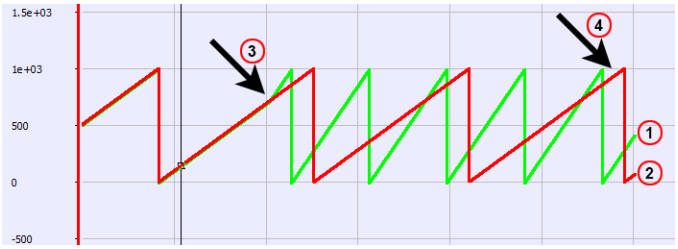
IMPORTANT

Set the `RatioSlope < (Ratio * EtherCAT Update Rate).`
The Gear block makes a jump (without a ramp) from one gear to the next when the RatioSlope is greater than the Ratio change factor multiplied by the update rate scale factor.

4.1.10.11.3.1 RatioSlope Offset

4.1.10.12 With Offset

- If **MLGearWriteRatSlp** is set as `MLGearWriteRatSlp( PipeNetwork.GEAR1 12 , Gear1RatioSlope 500.0 );` to generate a ramp (instead of a step) when going from a gear ratio of 1 to 2, there is a position offset when the gear ratio settles as 2.
- In this image:
  - The ratio goes from 1.0 to 2.0.
  - Green is PN Gear Block Output.
  - Red is Gearbox Input.



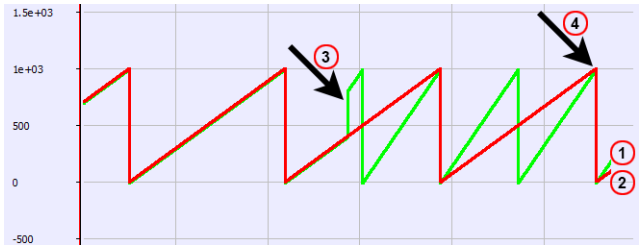
1. Green line: PN Gear Block Output.
2. Red line: PN Gearbox Input.
3. When the ratio changes.
4. Phase difference.

4.1.10.13 Without Offset

If **MLGearWriteRatSlp** is set without a ramp,

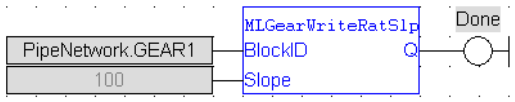
```
MLGearWriteRatSlp( PipeNetwork.GEAR1    12 , Gear1RatioSlope 1e+301 );
```

, there is no offset.

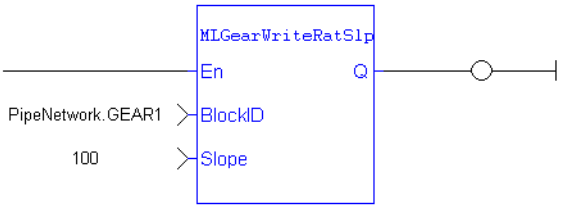


1. Green line: PN Gear Block Output.
2. Red line: PN Gearbox Input.
3. When the ratio changes.
4. Synched.

4.1.10.13.1 FBD Language Example



4.1.10.13.2 FFLD Language Example



4.1.10.13.3 IL Language Example

Not available.

4.1.10.13.4 ST Language Example

```
//Set the Ratio Slope value of Gear1 Pipe Block to 100
MLGearWriteRatSlp(PipeNetwork.GEAR1, 100.0);
```

See Also

- "MLGearInit" (→ p. 317)
- "MLGearReadOffSlp" (→ p. 321)

4.1.11 Integrator

Name	Description
MLIntInit	Initializes an integrator object.
MLIntWriteOutVal	Sets the output value of an integrator object.

4.1.11.1 MLIntWriteOutVal

Pipe Network ✓

Function - Sets the output value of an integrator object.

4.1.11.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Integrator object.
Value	LREAL	No range	User units	No default	Designated new output value of the selected Integrator object.

4.1.11.1.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the output value if the Integrator object is changed.

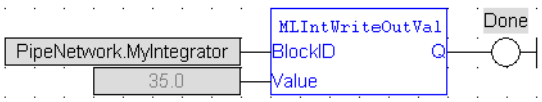
4.1.11.1.3 Remarks

- This function can force the output to an entered value not dependent on the input value from the Pipe Network.
- Output value can jump to another value instantly after the function is executed if the Pipe Network is running

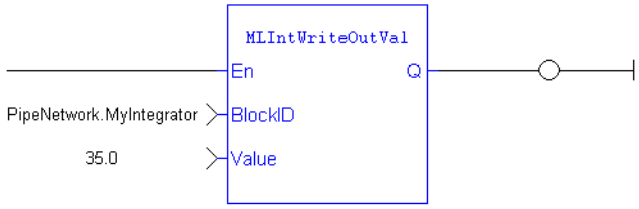
NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

4.1.11.1.4 FBD Language Example



4.1.11.1.5 FFLD Language Example



4.1.11.1.6 IL Language Example

Not available.

4.1.11.1.7 ST Language Example

```
//change the output value of an integrator object to 35
MLIntWriteOutVal ( PipeNetwork.MyIntegrator, 35.0 );
```

See Also

"MLIntInit" (→ p. 329)

4.1.11.2 MLIntInit

Pipe Network

Function - Initializes an integrator object.

4.1.11.2.1 Inputs

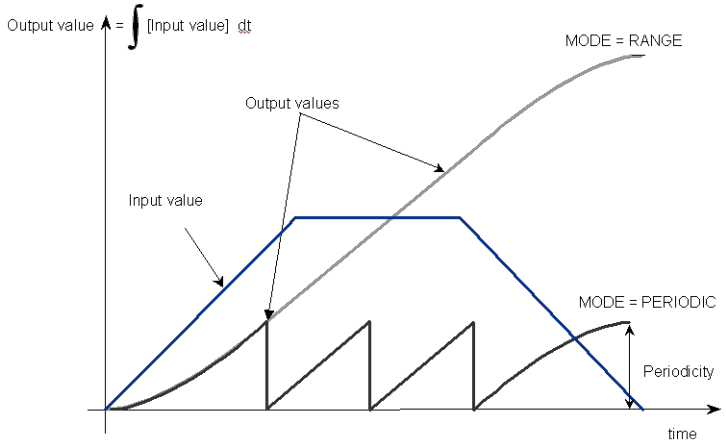
Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of a created Pipe Block.
ModuloPosition	LREAL	No range	User units	360.0	Output ModuloPosition of Integrator object.
Modulo	BOOL	0, 1	N/A	TRUE	TRUE when mode is modulo. Modulo mode adapts the output values according to the ModuloPosition (modulo).

4.1.11.2.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if the Integrator object is initialized.

4.1.11.2.3 Remarks

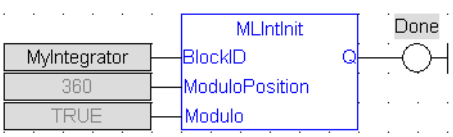
- The function block is automatically called if an Integrator Block is added to the Pipe Network.
  - User-defined settings are entered in the **Pipe Blocks Properties** screen.
- The Integrator object can operate in modulo or not modulo mode.
  - While in modulo mode, the output values are adapted according to the entered **ModuloPosition** value.



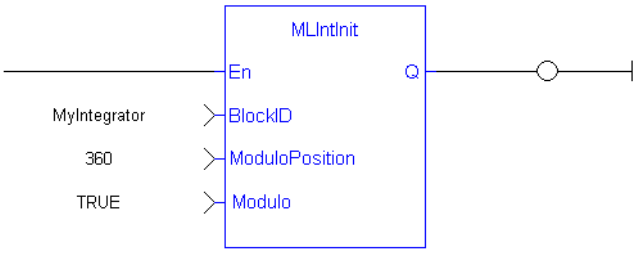
NOTE

Integrator objects are normally created in the Pipe Network using the graphical engine.
You do not have to add **MLIntInit** function blocks to their programs.
Parameters are entered directly in pop-up windows.
The code is automatically added to the current project.

4.1.11.2.4 FBD Language Example



4.1.11.2.5 FFLD Language Example



4.1.11.2.6 IL Language Example

Not available.

4.1.11.2.7 ST Language Example

```
//Initiate an Integrator Pipe Block named "MyIntegrator" with a Modulo of 360
MLIntInit(MyIntegrator, 360.0, true );
```

See Also

"MLBlkCreate" (→ p. 212)

"MLIntWriteOutVal" (→ p. 328)

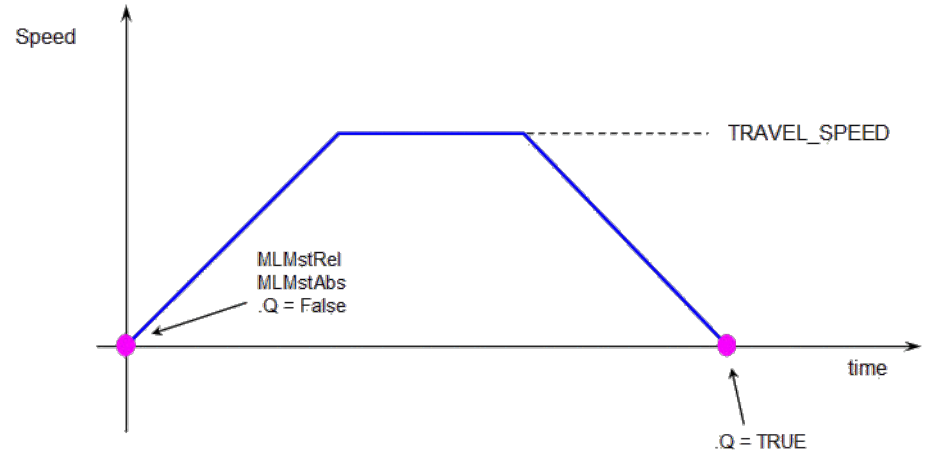
4.1.12 Master

TIP

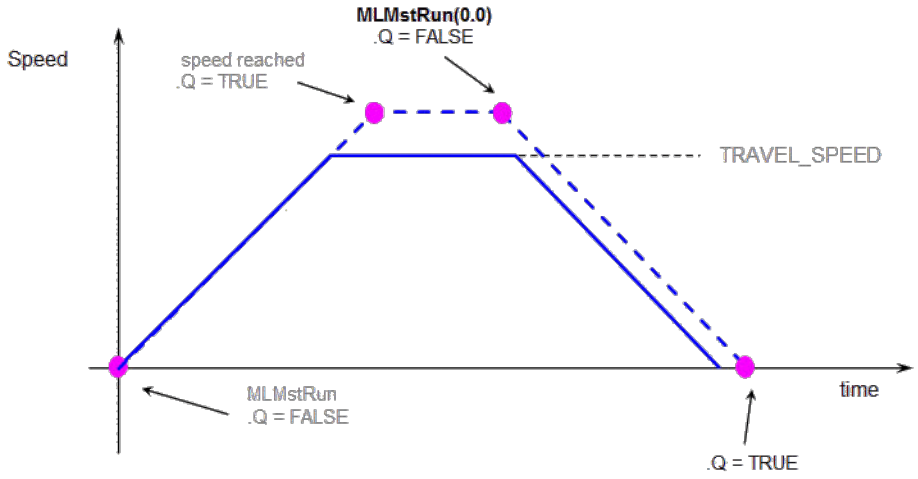
- See "Examples of Master Functions" (→ p. 331) for more information.

Name	Description
MLMstAbs	Performs a move to an absolute position.
MLMstAdd	Performs a move for a specified distance relative to the endpoint of the previous move.
MLMstForcePos	Forces the position of a Master Block to a specified position.
MLMstInit	Initializes a Master TMP generator block.
MLMstReadAccel	Gets the present acceleration value of a master block.
MLMstReadDecel	Gets the present deceleration value of a master block.
MLMstReadInitPos	Gets the initial position of a master block.
MLMstReadSpeed	Gets the speed of a master block.
MLMstRel	Performs a relative move for a specified distance from the current position.
MLMstRun	Jogs at the specified speed.
MLMstStatus	Returns the status of the generator.
MLMstWriteAccel	Sets the acceleration of a master block.
MLMstWriteDecel	Sets the deceleration of a master block.
MLMstWriteInitPos	Sets the initial position of a master block.
MLMstWriteSpeed	Sets the speed of a master block.

4.1.12.1 Examples of Master Functions



MLMstRun(0.0) reduce the speed down to 0.



Master Functions Usage

4.1.12.2 MLMstAbs

Pipe Network ✓

Function - Performs a move to an absolute position.

4.1.12.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Master Block.
Position	LREAL	No range	User units	No default	Sets the value of the absolute destination position. • See "Position with Modulo On" (→ p. 333) for more information. • See Set the Axis Block Position Units for more information.

4.1.12.2.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

4.1.12.2.3 Remarks

- See "MLMstAbs" (→ p. 332) for more information.

NOTE

Perform one of these actions to prevent undesired axis movement if modulo is turned on:

- Verify the **Position** value is within the modulo range.
- If the **Position** value is outside of the modulo range, call **MLMstAbs** function only once. See the **TIP** for an example of how to do this.

TIP

To reduce the load on the CPU, call **MLMstAbs** only once for each move.
This is achieved by adding a control variable, shown here:

```
          // Master: modulo is on and modulo range is 0 - 360 with rollover
at 360

If Not MoveStarted Then
     Position := 500;
     MLMstAbs( PipeNetwork.MASTER, Position);
     MovesStarted := TRUE;
End_if;
```

4.1.12.2.3.1 Position with Modulo On

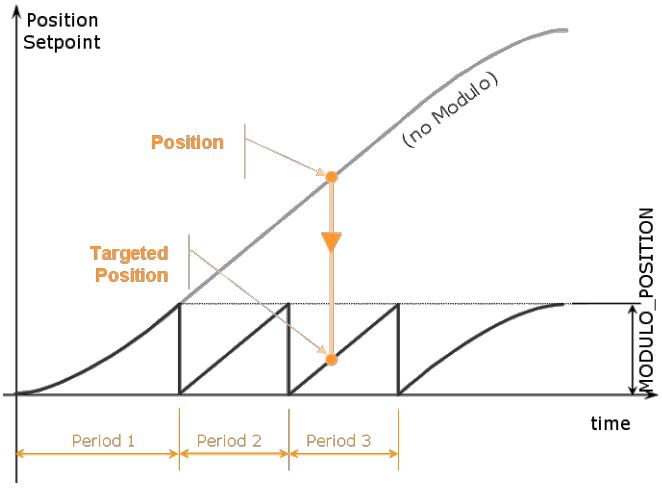
Position with Modulo On

NOTE

This information applies to both "MLMstAbs" (→ p. 332) and "MLAxisAbs" (→ p. 226).
For simplicity, the term Axis Block also refers to Master Block.

When the Modulo is turned on, the Axis Block moves to the targeted position during the corresponding period and is calculated as:

- If the Position input is between 0 (zero) and the Modulo position, the Axis Block moves within the **current** period (no position rollover).
- If the Position input is greater than the Modulo position, the Axis Block moves during one of the **next** period (positive position rollover).



The Axis Block works similarly for negative positions.

If the Position input is less than 0 (zero), the Axis Block moves during one of the **previous** period (negative position rollover).

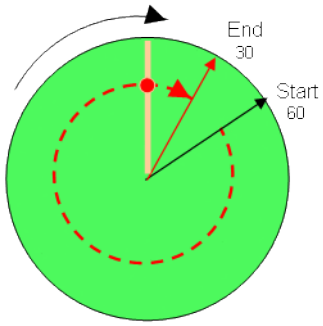
4.1.12.3 Forcing the Direction of Rotation

In some applications, the direction of rotation for the axis is forced in one direction only.

As a consequence, the motor movement goes to the next or previous modulo in these situations:

4.1.12.3.0.1 End Position is Less Than Start Position

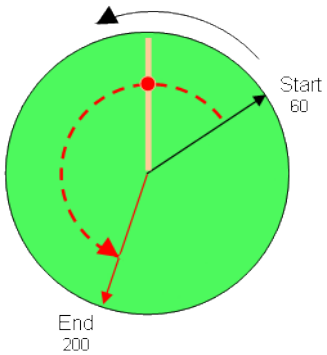
If the **End Position** is less than the **Start Position** and the direction of rotation for the axis is forced to be clockwise, the red point shows when the Modulo position is reached.



See Row 2 of the "MLMstAbs" (→ p. 332).

4.1.12.3.0.2 End Position is Greater Than Start Position

If the **End Position** is greater than the **Start Position** and the direction of rotation for the axis is forced to be counter clockwise.



See Row 4 of the "MLMstAbs" (→ p. 332).

Example Table

Start Position	End Position	Direction of Rotation	Cross Modulo	Position Input to MLAxisAbs (1)	RelativeDistance Moved (2)	
60	200	Clockwise	No	200	140	(i.e., 200 - 60 + 0)
60	30	Clockwise	Yes	390	330	(i.e., 30 - 60 + 360)
60	30	Counter clockwise	No	30	-30	(i.e., 30 - 60 - 0)
60	200	Counter clockwise	Yes	-160	-220	(i.e., 200 - 60 - 360)

With:

(1) **Position Input** = End Position (+ Modulo * Direction of rotation)

(2) **Relative Distance Moved** = End Position - Start Position (+ Modulo * Direction of rotation)

Where:

Direction of rotation = 1 when clockwise and -1 when anti-clockwise.

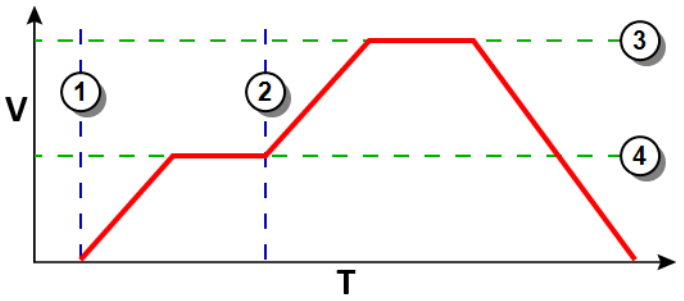
4.1.12.3.0.3 Travel Speed Update with MLAxisAbs

The travel speed of the generator can be updated using the function block "MLAxisGenWriteSpd" (→ p. 245).

Depending on the state of the generator, this speed is directly reflected on the current move or a future move.

- If **MLAxisAbs** is not currently being executed, the new travel speed is applied for the trajectory calculation for a future **MLAxisAbs** command.
- If **MLAxisAbs** is currently being executed, and a new MLAxisAbs with the **same** target position is called, the new travel speed is taken into account only if the current state of the TMP profile is the constant velocity or acceleration.
- If the axis was decelerating to stop at the goal position the new travel speed is not taken into account.
- If a **MLAxisAbs** is currently being executed, and a new MLAxisAbs with a **different** target position is called, the new travel speed is taken into account.

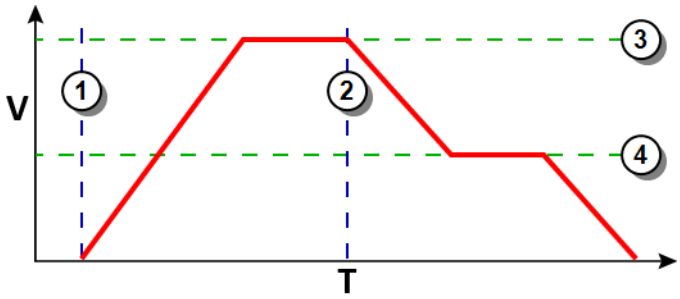
Example 1



1. First MLAxisAbs
2. Second MLAxisAbs
3. Initial Speed
4. New Speed

Figure 4-32: Initial speed is smaller than the new speed.

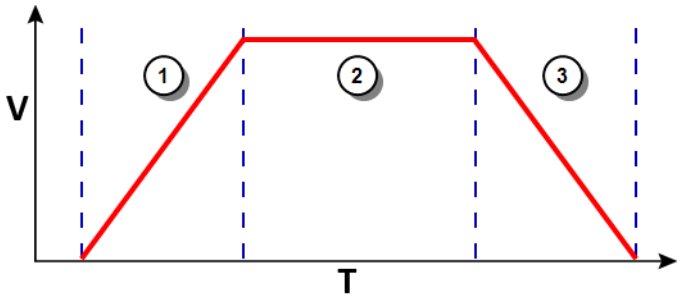
Example 2



1. First MLAxisAbs
2. Second MLAxisAbs
3. Initial Speed
4. New Speed

Figure 4-33: Initial speed is bigger than the new speed.

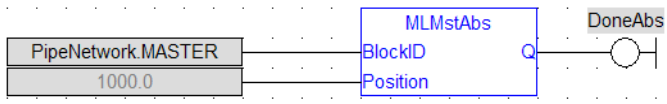
Example 3



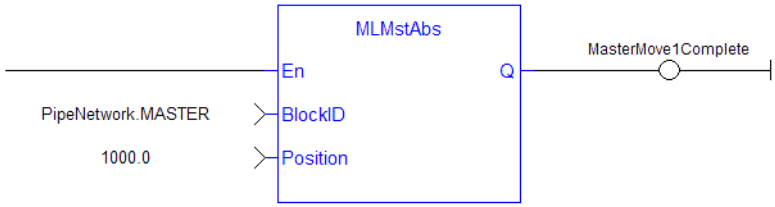
1. Acceleration
2. Constant Velocity
3. Deceleration

Figure 4-34: The speed update is taken into account only if the second MLAxisAbs is triggered during acceleration or constant velocity.

4.1.12.3.1 FBD Language Example



4.1.12.3.2 FFLD Language Example



4.1.12.3.3 IL Language Example

Not available.

4.1.12.3.4 ST Language Example

```
//Make an absolute positon move with a Master block called "MASTER" to
position 1000.0
MLMstAbs( PipeNetwork.MASTER, 1000.0 );
```

See Also

- "MLMstWriteSpeed" (→ p. 351)
- "MLMstReadDecel" (→ p. 342)

4.1.12.4 MLMstAdd

Pipe Network ✓

Function - Performs a move for a specified distance relative to the endpoint of the previous move.

4.1.12.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the FB to be executed.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Master Block.
DeltaPos	LREAL	No range	N/A	No default	Relative distance to move.

4.1.12.4.2 Outputs

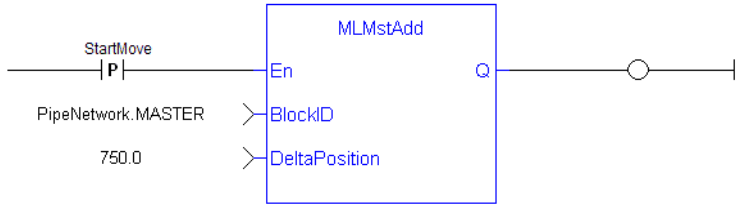
Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

4.1.12.4.3 Remarks

None

4.1.12.4.4 FBD Language Example

4.1.12.4.5 FFLD Language Example



NOTE

A pulse contact is required to start the FB.

4.1.12.4.6 IL Language Example

Not available.

4.1.12.4.7 ST Language Example

```
//At the endpoint of the previous move, with the Master  pipe block named
“MASTER”, make a move of 750.0
MLMstAdd( PipeNetwork.MASTER, 750.0 );
```

See Also

- "MLMstWriteDecel" (→ p. 349)
- "MLMstWriteSpeed" (→ p. 351)

4.1.12.5 MLMstForcePos

Pipe Network ✓

Function - Forces the position of a Master Block to a specified position.

4.1.12.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the FB to be executed.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Master Block.
Position	LREAL	No range	User units	No default	Defines the Master starting position when the motion starts.

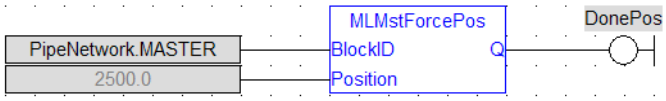
4.1.12.5.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE when the function successfully executes.

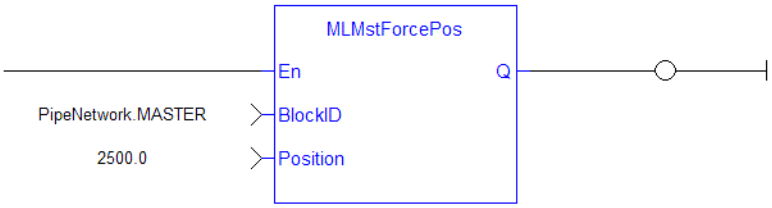
4.1.12.5.3 Remarks

- This block can only be executed when motion is **not** occurring.
- It can be used to force the master starting position to the desired values from which to start motion.

4.1.12.5.4 FBD Language Example



4.1.12.5.5 FFLD Language Example



NOTE

A pulse contact is required to start the FB.

4.1.12.5.6 IL Language Example

Not available.

4.1.12.5.7 ST Language Example

```
//Reset the output position of the Master Pipe Block named "Master" to 2500.0
MLMstForcePos(PipeNetwork.Master, 2500.0);
```

See Also

"MLMstReadInitPos" (→ p. 343)

4.1.12.6 MLMstInit

Pipe Network

Function - Initializes a Master TMP generator block.

4.1.12.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Master Block.
ModuloPosition	LREAL	No range	User units	No default	Modulo Position for cyclic motion systems expressed in user logical units (Position Rollover Value).
Period	LREAL	No range	Cycles	No default	Sampling period of the generator expressed according to the update cycle. Example: 2.0 means the sampling is done once every 2 cycles.
Speed	LREAL	No range	User units	No default	Travel speed value expressed in user logical units per second. • The travel speed value is used to set the constant speed part of the TMP. • See Set the Axis Block Position Units for more information.
Acceleration	LREAL	No range	User unit/sec^2	No default	Acceleration value expressed in user logical units per second squared. • The acceleration value is always used to generate the first part of the TMP. • See Set the Axis Block Position Units for more information.
Deceleration	LREAL	No range	User unit/sec^2	No default	Deceleration value expressed in user logical units per second squared. • The deceleration value is always used to generate the last part of the TMP. • See Set the Axis Block Position Units for more information.
InitialPosition	LREAL	No range	User units	No default	Initial position value expressed in user logical units. Used only at the pipe activation to initialize the position starting point.

Input	Data Type	Range	Unit	Default	Description
Modulo	BOOL	0, 1	N/A	No default	The available modes are: • TRUE - Modulo • FALSE - No modulo

4.1.12.6.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

4.1.12.6.3 Remarks

- This function is automatically created when the MLMaster Block is included in the Pipe Network Editor.
- Based on the parameters defined in the pipe block, the Inputs for this function are initialized by default.
  - See example in the "TMP Initialization" (→ p. 340) image.

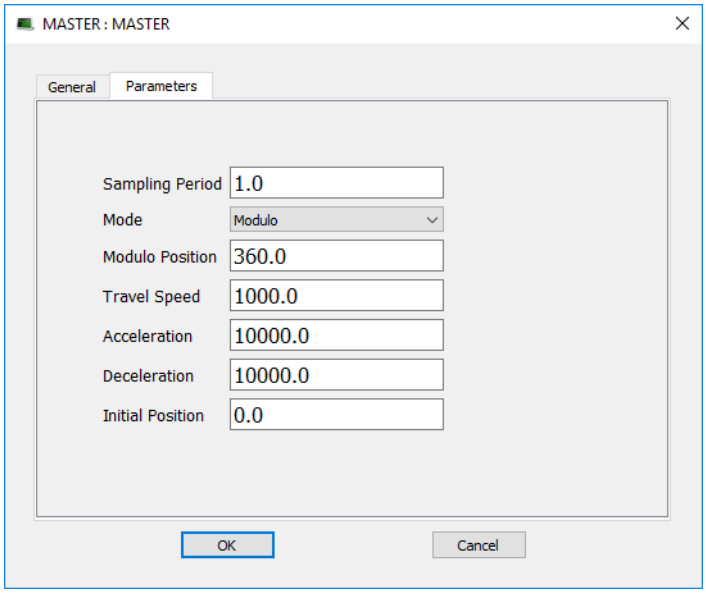
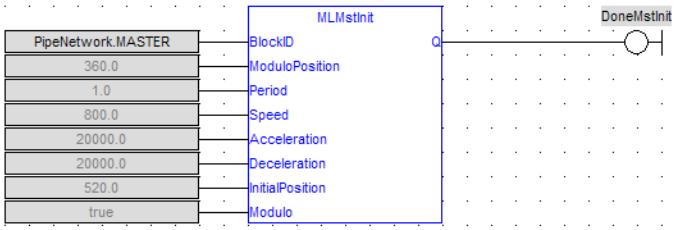
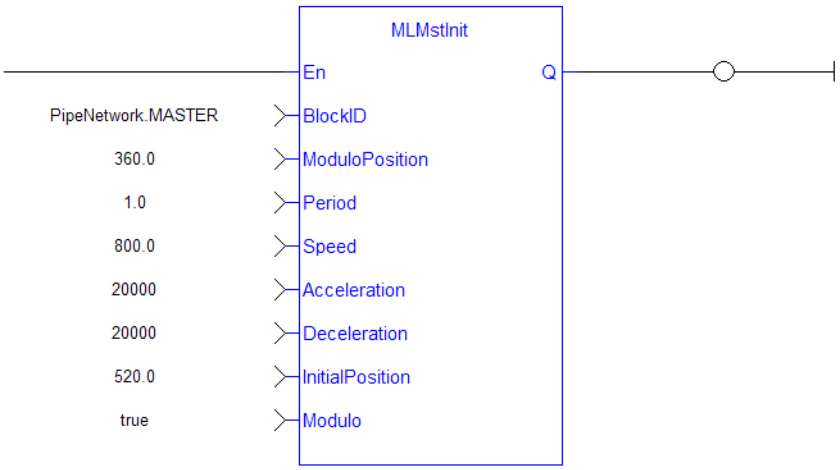


Figure 4-35: TMP Initialization

4.1.12.6.4 FBD Language Example



4.1.12.6.5 FFLD Language Example



4.1.12.6.6 IL Language Example

Not available.

4.1.12.6.7 ST Language Example

```
//Initialize  a Master Pipe Block named "MASTER" to a Modulo of 360, motion
generator sample period of 1,Speed of 1000.0, Accel and Decel of 10000.0,
Initial position of 0.0,
MLMstInit( PipeNetwork.MASTER, 360.0, 1.0, 1000.0, 10000.0, 10000.0, 0.0,
true );
```

4.1.12.7 MLMstReadAccel

Pipe Network

Function - Gets the present acceleration value of a master block.

4.1.12.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the FB to be executed.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Master Block.

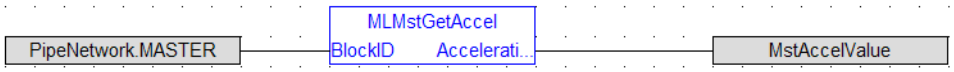
4.1.12.7.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL			Returns TRUE when the function successfully executes.
Acceleration	LREAL		User unit/sec^2	Returns the acceleration value. See Set the Axis Block Position Units for more information.

4.1.12.7.3 Remarks

None

4.1.12.7.4 FBD Language Example



4.1.12.7.5 FFLD Language Example

4.1.12.7.6 IL Language Example

Not available.

4.1.12.7.7 ST Language Example

```
// Read the present  acceleration of a  Pipe Block named "MASTER"
MLMstReadAccel( PipeNetwork.MASTER );
```

See Also

- "MLMstReadDecel" (→ p. 342)
- "MLMstReadSpeed" (→ p. 344)

4.1.12.8 MLMstReadDecel

Pipe Network ✓

Function - Gets the present deceleration value of a master block.

4.1.12.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the FB to be executed.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Master Block.

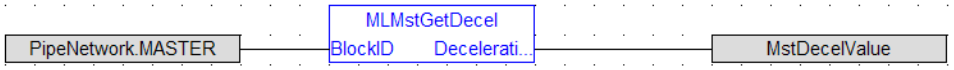
4.1.12.8.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL			Returns TRUE when the function successfully executes.
Deceleration	LREAL		User unit/sec^2	Returns the deceleration value. See Set the Axis Block Position Units for more information.

4.1.12.8.3 Remarks

None

4.1.12.8.4 FBD Language Example



4.1.12.8.5 FFLD Language Example

4.1.12.8.6 IL Language Example

Not available.

4.1.12.8.7 ST Language Example

```
// Read the present  deceleration  of a  Pipe Block named "MASTER"
MLMstReadDecel( PipeNetwork.MASTER );
```

4.1.12.9 MLMstReadInitPos

Pipe Network

Function - Gets the initial position of a master block.

4.1.12.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the FB to be executed.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	Pipe Network Block.

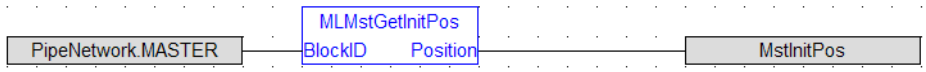
4.1.12.9.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL			Returns TRUE when the function successfully executes.
Position	LREAL		User units	Returns Initial Position. See Set the Axis Block Position Units for more information.

4.1.12.9.3 Remarks

None

4.1.12.9.4 FBD Language Example



4.1.12.9.5 FFLD Language Example

4.1.12.9.6 IL Language Example

Not available.

4.1.12.9.7 ST Language Example

```
MstInitPos := MLMstReadInitPos( PipeNetwork.MASTER );
```

4.1.12.10 MLMstReadSpeed

Pipe Network

Function - Gets the speed of a master block.

4.1.12.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the FB to be executed.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Master Block.

4.1.12.10.2 Outputs

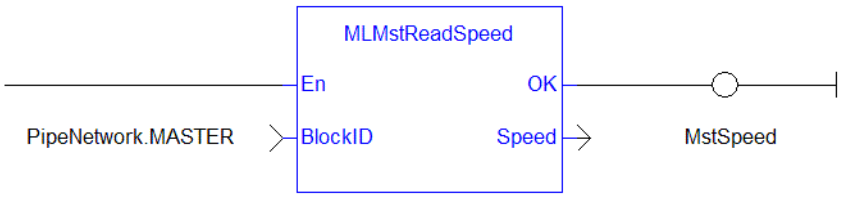
Output	Data Type	Range	Unit	Description
OK	BOOL			Returns TRUE when the function successfully executes.
Speed	LREAL		User unit/sec	Returns the current speed. See Set the Axis Block Position Units for more information.

4.1.12.10.3 Remarks

None

4.1.12.10.4 FBD Language Example

4.1.12.10.5 FFLD Language Example



4.1.12.10.6 IL Language Example

Not available.

4.1.12.10.7 ST Language Example

```
MstSpeed := MLMstReadSpeed( PipeNetwork.MASTER );
```

See Also

- "MLMstReadAccel" (→ p. 341)
- "MLMstReadDecel" (→ p. 342)

4.1.12.11 MLMstRel

Pipe Network ✓

Function - Performs a relative move for a specified distance from the current position.

4.1.12.11.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the FB to be executed.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Master Block.
DeltaPos	LREAL	No range	User units	No default	Relative distance to the move.

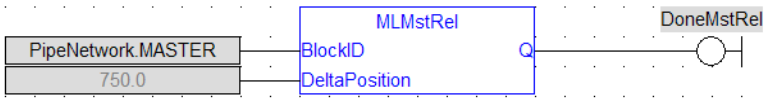
4.1.12.11.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

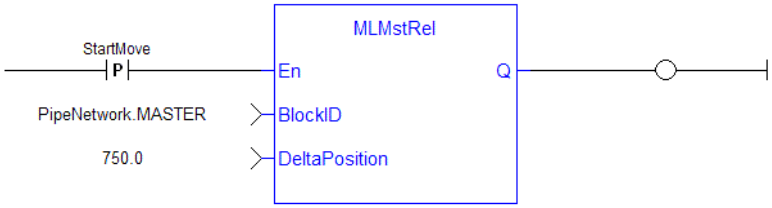
4.1.12.11.3 Remarks

None

4.1.12.11.4 FBD Language Example



4.1.12.11.5 FFLD Language Example



NOTE

A pulse contact is required to start the FB.

4.1.12.11.6 IL Language Example

Not available.

4.1.12.11.7 ST Language Example

```
MLMstRel( PipeNetwork.MASTER, 750.0 );
```

See Also

- "MLMstWriteDecel" (→ p. 349)
- "MLMstWriteSpeed" (→ p. 351)

4.1.12.12 MLMstRun

Pipe Network ✓

Function - Jogs at the specified speed.

4.1.12.12.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the FB to be executed.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Master Block.
Speed	LREAL	No range	User unit/sec^2	No default	Defines the Master starting position when the motion starts. See Set the Axis Block Position Units for more information.

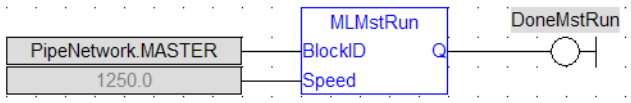
4.1.12.12.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

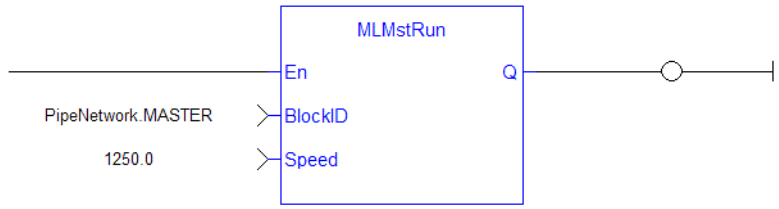
4.1.12.12.3 Remarks

None

4.1.12.12.4 FBD Language Example



4.1.12.12.5 FFLD Language Example



4.1.12.12.6 IL Language Example

Not available.

4.1.12.12.7 ST Language Example

```
MLMstRun( PipeNetwork.MASTER, 1250.0 );
```

See Also

- "MLMstWriteDecel" (→ p. 349)
- "MLMstWriteSpeed" (→ p. 351)

4.1.12.13 MLMstStatus

Function - Returns the status of the generator.

4.1.12.13.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the FB to be executed.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Master Block.

4.1.12.13.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL			Returns TRUE when the function successfully executes.
Default (.Q)	DINT			Returns the status of the generator.

4.1.12.13.3 Remarks

- The value returned is the state being executed by the TMP generator as it processes the various motion commands.
- Some states are transitory, others are stable until the next event takes place.

These terms are relevant to the returned values:

Term	Definition
Running	Speed is non-zero.
Stopped	Speed is 0 (zero).
Positioning	A target position has been programmed with a relative, additive, or absolute command.

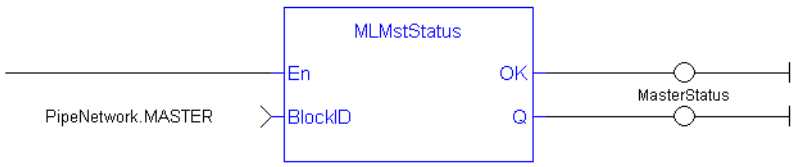
4.1.12.13.3.1 Status Definitions

Status	Definition
0	**New speed programmed** is entered when both: • A jog move (**MLMstRun**) is commanded. • The current speed is **not** at the commanded speed.
1	**Stable state Running or Stopped** is entered when both: • A jog move (**MLMstRun**) is commanded. • The current speed is at the commanded speed. **Stable state Running or Stopped** is entered when both: • A position move is programmed. • Motion is completed.
2	**Speed change** is entered when the current speed is greater than the commanded speed.
3	**Speed reversal while positioning** is entered when both: • A position move is programmed. • The distance to go requires a speed reversal.

Status	Definition
4	**Acceleration while positioning** current speed is below the travel speed.
5	**Constant Speed while positioning** is entered when both: • A positioning move is commanded. • The current speed is at the commanded speed.
6	**Deceleration while positioning** is entered when both: • A positioning move is commanded. • The current speed is changing to achieve the target position at 0 (zero) speed.
7	**Micro step** is entered when both: • A small change in position is required. • The current speed is 0 (zero).

4.1.12.13.4 FBD Language Example

4.1.12.13.5 FFLD Language Example



4.1.12.13.6 IL Language Example

Not available.

4.1.12.13.7 ST Language Example

```
MasterStatus := MLMstStatus( PipeNetwork.MASTER );
```

4.1.12.14 MLMstWriteAccel

Pipe Network

Function - Sets the acceleration of a master block.

4.1.12.14.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the FB to be executed.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Master Block.
Acceleration	LREAL	No range	User unit/sec^2	No default	Acceleration value expressed in user logical units per second squared. See Set the Axis Block Position Units for more information.

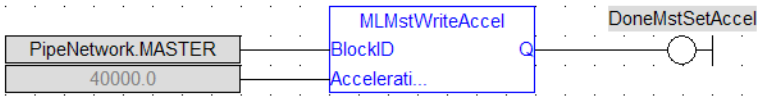
4.1.12.14.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

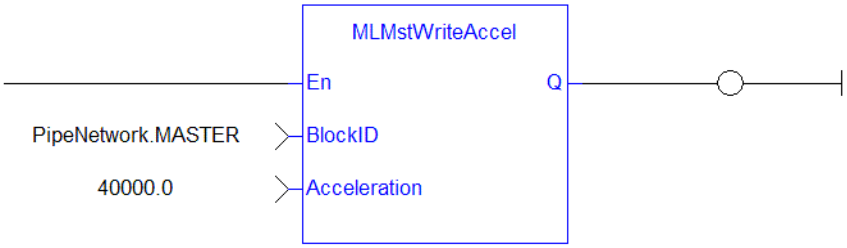
4.1.12.14.3 Remarks

None

4.1.12.14.4 FBD Language Example



4.1.12.14.5 FFLD Language Example



4.1.12.14.6 IL Language Example

Not available.

4.1.12.14.7 ST Language Example

```
MLMstWriteAccel( PipeNetwork.MASTER, 40000.0 );
```

See Also

- "MLMstAbs" (→ p. 332)
- "MLMstRel" (→ p. 345)
- "MLMstWriteDecel" (→ p. 349)
- "MLMstWriteSpeed" (→ p. 351)

4.1.12.15 MLMstWriteDecel

Pipe Network

Function - Sets the deceleration of a master block.

4.1.12.15.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the FB to be executed.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Master Block.

Input	Data Type	Range	Unit	Default	Description
Deceleration	LREAL	No range	User unit/sec^2	No default	Deceleration value expressed in user logical units per second squared. See Set the Axis Block Position Units for more information.

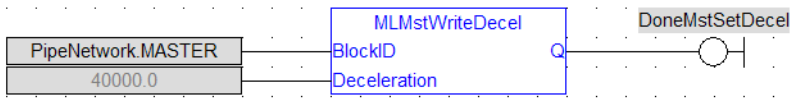
4.1.12.15.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

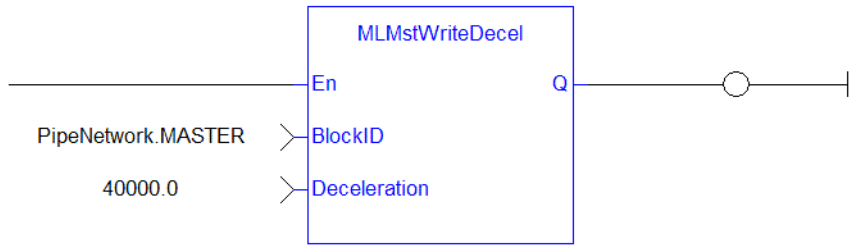
4.1.12.15.3 Remarks

None

4.1.12.15.4 FBD Language Example



4.1.12.15.5 FFLD Language Example



4.1.12.15.6 IL Language Example

Not available.

4.1.12.15.7 ST Language Example

```
MLMstWriteDecel( PipeNetwork.MASTER, 40000.0 );
```

See Also

"MLMstWriteSpeed" (→ p. 351)

4.1.12.16 MLMstWriteInitPos

Pipe Network

Function - Sets the initial position of a master block.

4.1.12.16.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the FB to be executed.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Master Block.
Position	LREAL	No range	User units	No default	Initial position.

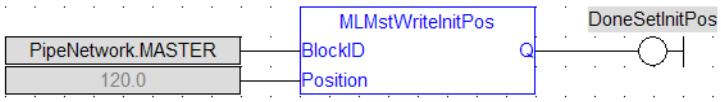
4.1.12.16.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

4.1.12.16.3 Remarks

None

4.1.12.16.4 FBD Language Example



4.1.12.16.5 FFLD Language Example

4.1.12.16.6 IL Language Example

Not available.

4.1.12.16.7 ST Language Example

```
MLMstWriteInitPos( PipeNetwork.MASTER, 120.0 );
```

4.1.12.17 MLMstWriteSpeed

Pipe Network

Function - Sets the speed of a master block.

4.1.12.17.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables the FB to be executed.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Master Block.

Input	Data Type	Range	Unit	Default	Description
Speed	LREAL	No range	User unit/sec	No default	Speed of the motion. See Set the Axis Block Position Units for more information.

4.1.12.17.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE when the function successfully executes.

4.1.12.17.3 Remarks

- Set the speed of motion for the "MLMstAbs" (→ p. 332) and "MLMstRel" (→ p. 345) blocks.
- This function or function block does not generate any motion.

4.1.12.17.4 FBD Language Example

4.1.12.17.5 FFLD Language Example

4.1.12.17.6 IL Language Example

Not available.

4.1.12.17.7 ST Language Example

```
MLMstWriteSpeed( PipeNetwork.MASTER, 400.0 );
```

See Also

"MLMstWriteDecel" (→ p. 349)

4.1.13 Phaser

TIP

- See "Example: Phaser Functions" (→ p. 353) for an example of Phaser functions.

Names	Description
MLPhaInit	Initializes a phaser Pipe Block.
MLPhaReadActPhase	Gets the actual phase value of a phaser block.
MLPhaReadPhase	Gets the phase value of a phaser block.
MLPhaReadSlope	Gets the phase slope value of a phaser block.

Names	Description
MLPhaWritePhase	Sets the phase value of a phaser block.
MLPhaWriteSlope	Sets the phase slope value of a phaser block.

TIP

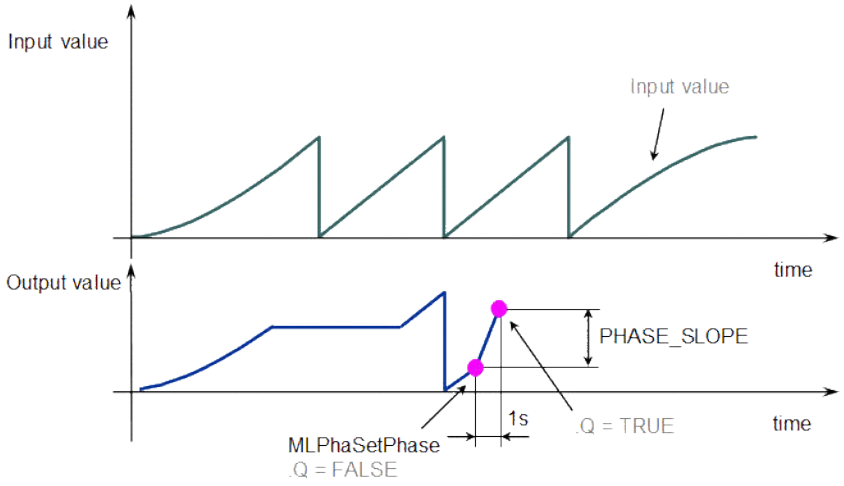
There is a delay when using an external encoder.
The delay is five cycles:
2 cycles to read the encoder from the AKD via EtherCAT,
1 cycle for computing,
2 cycles for sending the new position set point to the AKD).
This lag error is speed proportional (5 cycles * speed).
A Phaser block can be used to compensate for this lag.

When executing, the phaser block is in one of these states:

State	Description
Applying Phase	• Entered when the programmed Phase value is reached. • Exits to the **Changing** phase state whenever a new value is programmed via the "MLPhaWritePhase" (→ p. 358) function changes the Phase Offset target.
Changing Phase	• Entered when a new value is programmed. • Exits to the **Applying Phase** state when the programmed phase offset is reached. • The current Phase offset value is slewed to the new phase offset by the amount of the slew value.
Standby	• Entered when the Block is initialized. • Exits to the **Changing Phase** state when the Phase value is changed via the "MLPhaWritePhase" (→ p. 358) command.

4.1.13.1 Example: Phaser Functions

- Call "MLPhaWritePhase" (→ p. 358) function to modify the Phase value..
- Call "MLPhaWriteSlope" (→ p. 359) to modify the rate of change of phase, or slope, applied when the Phase value is changed.



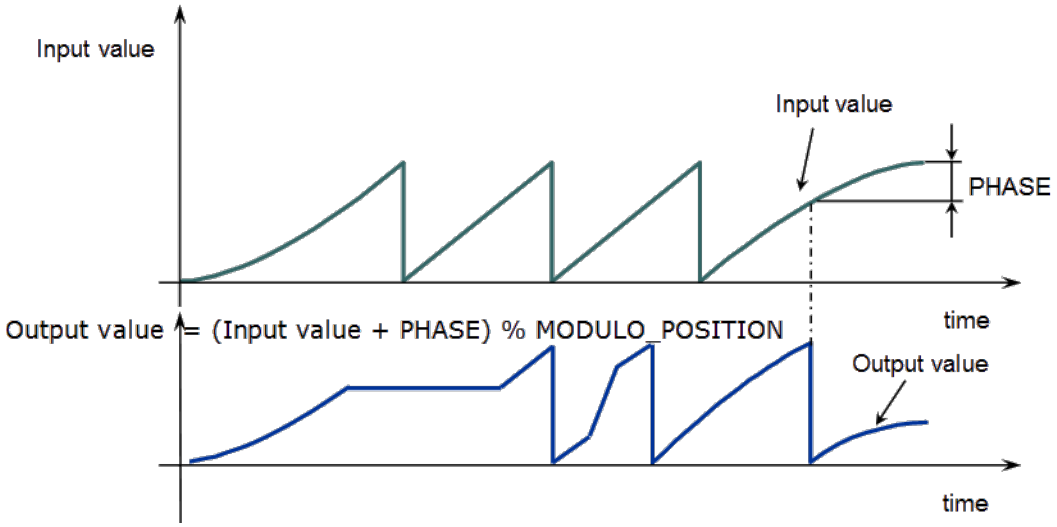


Figure 4-36: Phaser Functions Usage

NOTE

% MODULO_POSITION is in the equation to take into account the modulo (periodicity) of the value.

4.1.13.2 MLPhaInit

Pipe Network

Function - Initializes a phaser Pipe Block.

4.1.13.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a phaser function block in the Pipe Network.
ModuloPosition	LREAL	No range	User units	360.0	Rollover position of the phaser block.
Phase	LREAL	No range	N/A	0.0	Amount of phase adjustment.
UseUserSlope	BOOL	0, 1	N/A	Max Slope	Determines if the Max Slope or user-defined slope is used.
PhaseSlope	LREAL	No range	User unit/sec	0.0	User-defined slope for making the phase adjustment.
StandbyValue	LREAL	No range	N/A	0.0	This is the output from the Phaser Block, when the pipe is active, until the "MLPhaWritePhase" (→ p. 358) function is executed.

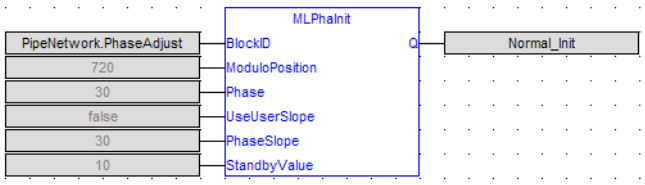
4.1.13.2.2 Outputs

Input	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE if the function or function block is successfully executing.

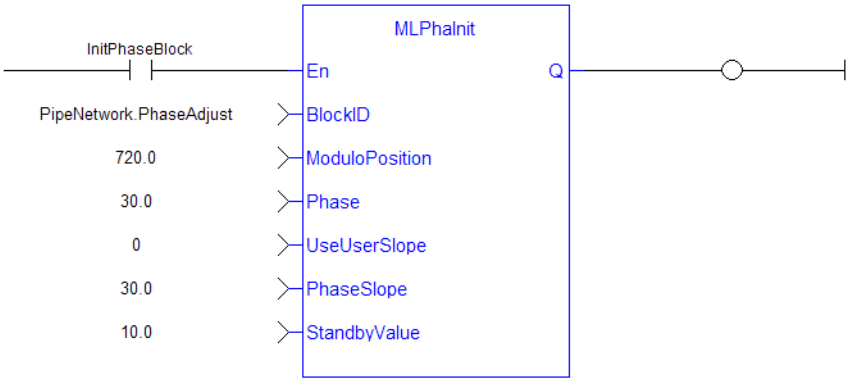
4.1.13.2.3 Remarks

- Returns TRUE if the function succeeded.
- This function block is automatically called by the Pipe Network(MLPN_CREATE_OBJECTS) if a Phaser block is added to the Pipe Network.
  - User-defined settings are entered in the Pipe Blocks Properties screen.
- The Phaser Pipe Block is assigned a Name, OUTPUT_PERIOD, PHASE, PHASE_SLOPE_TYPE, and STANDBY_VALUE.

4.1.13.2.4 FBD Language Example



4.1.13.2.5 FFLD Language Example



4.1.13.2.6 IL Language Example

Not available.

4.1.13.2.7 ST Language Example

```
//Initialize  a Phaser  Pipe Block named “PhaseAdjust” to a Modulo of 720,
phase offset value of 30, use the Max Slope
MLPhaInit( PipeNetwork.PhaseAdjust , 720, 30, false, 30 , 10 );
```

See Also

- "MLPhaReadActPhase" (→ p. 355)
- "MLPhaReadPhase" (→ p. 356)
- "MLPhaReadSlope" (→ p. 357)
- "MLPhaWritePhase" (→ p. 358)
- "MLPhaWriteSlope" (→ p. 359)

4.1.13.3 MLPhaReadActPhase

Pipe Network

Function - Gets the actual phase value of a phaser block.

4.1.13.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable					
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a phaser function block in the Pipe Network.

4.1.13.3.2 Outputs

Input	Data Type	Range	Unit	Description
OK				
Phase	LREAL			

4.1.13.3.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- If a PHASE_SLOPE_USER value is used, the new phase (refer to "MLPhaWritePhase" (→ p. 358)) isn't set immediately.
  - The phase is ramped with the slope value from the old phase value to the new phase value.
  - MLPhaReadActPhase returns this ramping value.
  - See "MLPhaReadSlope" (→ p. 357) and "MLPhaWriteSlope" (→ p. 359) for more information.
- "MLPhaReadPhase" (→ p. 356) returns the new value and this also when the phaser is still ramping.
  - If using Max Slope means no ramping.
  - **MLPhaReadActPhase** and **MLPhaReadPhase** always returns the same value.

4.1.13.3.4 FBD Language Example

Not available.

4.1.13.3.5 FFLD Language Example

Not available.

4.1.13.3.6 IL Language Example

Not available.

4.1.13.3.7 ST Language Example

Not available.

See Also

- "MLPhaInit" (→ p. 354)
- "MLPhaReadPhase" (→ p. 356)
- "MLPhaReadSlope" (→ p. 357)
- "MLPhaWritePhase" (→ p. 358)
- "MLPhaWriteSlope" (→ p. 359)

4.1.13.4 MLPhaReadPhase

Pipe Network ✓

Function - Gets the phase value of a phaser block.

4.1.13.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a phaser function block in the Pipe Network.

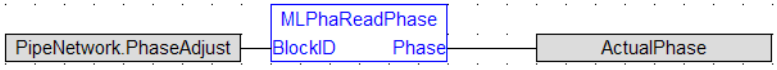
4.1.13.4.2 Outputs

Input	Data Type	Range	Unit	Description
Phase	LREAL			

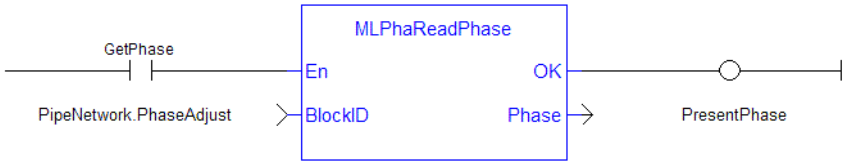
4.1.13.4.3 Remarks

None

4.1.13.4.4 FBD Language Example



4.1.13.4.5 FFLD Language Example



4.1.13.4.6 IL Language Example

Not available.

4.1.13.4.7 ST Language Example

```
PresentPhase := MLPhaReadPhase( PipeNetwork.PhaseAdjust );
```

See Also

- "MLPhaInit" (→ p. 354)
- "MLPhaReadActPhase" (→ p. 355)
- "MLPhaReadSlope" (→ p. 357)
- "MLPhaWritePhase" (→ p. 358)
- "MLPhaWriteSlope" (→ p. 359)

4.1.13.5 MLPhaReadSlope

Pipe Network

Function - Gets the phase slope value of a phaser block.

4.1.13.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a phaser function block in the Pipe Network.

4.1.13.5.2 Outputs

Input	Data Type	Range	Unit	Description
Slope	LREAL		User unit/sec	Present slope value. • Value defined in the setup of a phaser block in a Pipe Network. • Depending on the phase slope type setting, it is the VALUE in PHASE_SLOPE_USER, PHASE, or the Max Slope.

4.1.13.5.3 Remarks

None

4.1.13.5.4 FBD Language Example

4.1.13.5.5 FFLD Language Example

4.1.13.5.6 IL Language Example

Not available.

4.1.13.5.7 ST Language Example

```
PresentSlope :=MLPhaReadSlope( PipeNetwork.PhaseAdjust  );
```

See Also

- "MLPhaInit" (→ p. 354)
- "MLPhaReadActPhase" (→ p. 355)
- "MLPhaReadPhase" (→ p. 356)
- "MLPhaWritePhase" (→ p. 358)
- "MLPhaWriteSlope" (→ p. 359)

4.1.13.6 MLPhaWritePhase

Pipe Network

Function - Sets the phase value of a phaser block.

4.1.13.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a phaser function block in the Pipe Network.

Input	Data Type	Range	Unit	Default	Description
Phase	LREAL	No range	User unit/sec		Phase value. • Value defined in the setup of a phaser block in a Pipe Network. • It is in the PHASE field in the parameter tab.

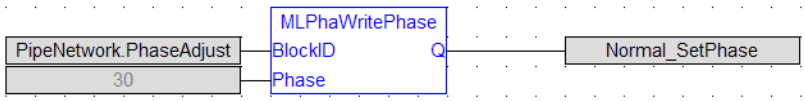
4.1.13.6.2 Outputs

Input	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE if the function or function block is successfully executing.

4.1.13.6.3 Remarks

None

4.1.13.6.4 FBD Language Example



4.1.13.6.5 FFLD Language Example

4.1.13.6.6 IL Language Example

Not available.

4.1.13.6.7 ST Language Example

```
MLPhaWritePhase( PipeNetwork.PhaseAdjust , 30 );
```

See Also

- "MLPhaInit" (→ p. 354)
- "MLPhaReadActPhase" (→ p. 355)
- "MLPhaReadPhase" (→ p. 356)
- "MLPhaReadSlope" (→ p. 357)
- "MLPhaWriteSlope" (→ p. 359)

4.1.13.7 MLPhaWriteSlope

Pipe Network

Function - Sets the phase slope value of a phaser block.

4.1.13.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a phaser function block in the Pipe Network.
Slope	LREAL	No range	User unit/sec		Sets the slope of the phase adjustment. • Value defined in the setup of a phaser block in a Pipe Network. • Depending on the phase slope type setting, it is the VALUE in PHASE_SLOPE_USER, PHASE, or the Max Slope.

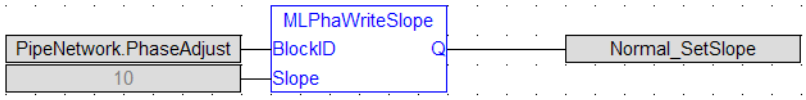
4.1.13.7.2 Outputs

Input	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE if the function or function block is successfully executing.

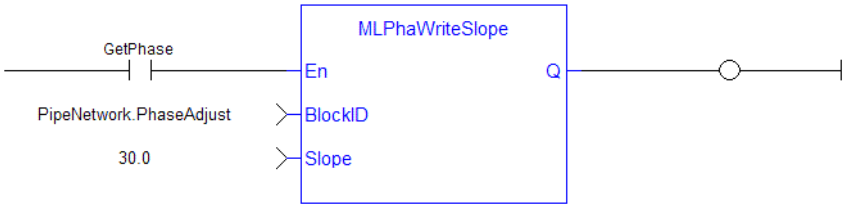
4.1.13.7.3 Remarks

Returns TRUE if the function succeeded.

4.1.13.7.4 FBD Language Example



4.1.13.7.5 FFLD Language Example



4.1.13.7.6 IL Language Example

Not available.

4.1.13.7.7 ST Language Example

```
MLPhaWriteSlope( PipeNetwork.PhaseAdjust , 10 );
```

See Also

- "MLPhaInit" (→ p. 354)
- "MLPhaReadActPhase" (→ p. 355)
- "MLPhaReadPhase" (→ p. 356)
- "MLPhaReadSlope" (→ p. 357)
- "MLPhaWritePhase" (→ p. 358)

4.1.14 PMP

Name	Description
MLPmpAbs	Moves to an absolute position using a parabolic acceleration profile.
MLPmpForcePos	Forces the position of a PMP block to a specified position.
MLPmpInit	Initializes a PMP block for use in a PLC program.
MLPmpReadAccel	Gets the Acceleration parameter of a PMP block.
MLPmpReadFstSpd	Gets the FirstTravelSpeed parameter of a PMP block.
MLPmpReadInitPos	Gets the Initial Position parameter of a PMP block.
MLPmpReadJerk	Gets the Jerk parameter of a PMP block.
MLPmpReadLstSpd	Gets the LastTravelSpeed parameter of a PMP block.
MLPmpRel	Used to perform two subsequent relative moves.
MLPmpRun	Jog the generator at the specified speed.
MLPmpStatus	Returns the status of the PMP block generator.
MLPmpWriteAccel	Sets the Acceleration parameter of a PMP block.
MLPmpWriteFstSpd	Sets the FirstTravelSpeed parameter of a PMP block.
MLPmpWriteJerk	Sets the Jerk parameter of a PMP block.
MLPmpWriteLstSpd	Sets the LastTravelSpeed parameter of a PMP block.

4.1.14.1 MLPmpAbs

Function - Moves to an absolute position using a parabolic acceleration profile.

4.1.14.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a PMP function block in the Pipe Network.
Position	LREAL	No range	User units	No default	Absolute position of the motor/load to be at after this function block is complete.

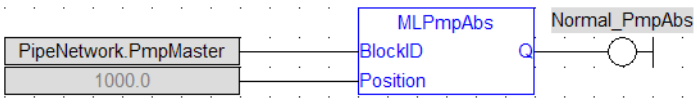
4.1.14.1.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE if the function or function block is successfully executing.

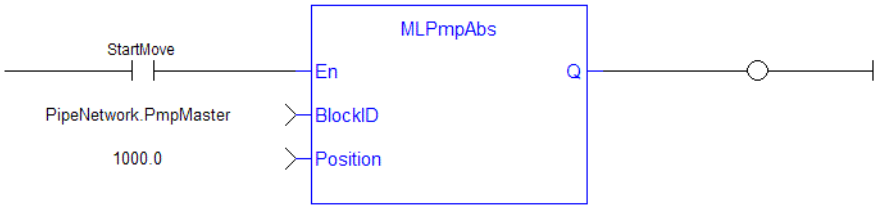
4.1.14.1.3 Remarks

- The FIRST_TRAVEL_SPEED is used as the velocity for the motion.
- JERK determines the level of parabolic acceleration.
- Returns TRUE if the function succeeded.

4.1.14.1.4 FBD Language Example



4.1.14.1.5 FFLD Language Example



4.1.14.1.6 IL Language Example

Not available.

4.1.14.1.7 ST Language Example

```
MLPmpAbs( PipeNetwork.PmpMaster, 1000.0 ) ;
```

See Also

- "MLPmpWriteAccel" (→ p. 374)
- "MLPmpWriteJerk" (→ p. 376)
- "MLPmpWriteFstSpd" (→ p. 375)

4.1.14.2 MLPmpForcePos

Pipe Network ✓

Function - Forces the position of a PMP block to a specified position.

4.1.14.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables execution.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a PMP function block in the Pipe Network.
Position	LREAL	No range	User units	No default	Defines the PMP starting position when the motion starts.

4.1.14.2.2 Outputs

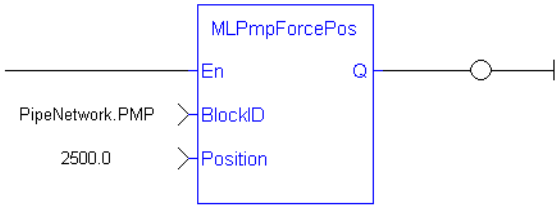
Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE when the function successfully executes.

4.1.14.2.3 Remarks

- This block can only be executed when motion is **NOT** occurring.
- It can be used to force the PMP starting position to the values to start motion from.

4.1.14.2.4 FBD Language Example

4.1.14.2.5 FFLD Language Example



NOTE

A pulse contact is required to start the FB.

4.1.14.2.6 IL Language Example

Not available.

4.1.14.2.7 ST Language Example

```
MLPmpForcePos( PipeNetwork.PMP, 2500.0 );
```

See Also

"MLPmpReadInitPos" (→ p. 367)

4.1.14.3 MLPmpInit

Pipe Network

Function - Initializes a PMP block for use in a PLC program.

4.1.14.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a PMP function block in the Pipe Network.
ModuloPosition	LREAL	No range	User units	360.0	Modulo Position for cyclic motion systems expressed in user logical units. Position Rollover Value.
Period	LREAL	No range	User units	1.0	Sampling period of the generator expressed according to the update cycle. Example: 2.0 means the sampling is done once every 2 cycles.
FirstTravelSpeed	LREAL	No range	User unit/sec	100.0	First travel speed of the motion.

Input	Data Type	Range	Unit	Default	Description
LastTravelSpeed	LREAL	No range	User unit/sec	100.0	Last travel speed of the motion.
Acceleration	LREAL	No range	User unit/sec^2	1000.0	Acceleration of the PMP block motion.
Jerk	LREAL	No range	User unit/sec^3	0 (zero)	Jerk.
InitialPosition	LREAL	No range	User units	0 (zero)	Initial position of the PMP block when the Pipe Network is started.
Modulo	BOOL	0, 1	N/A	No default	The available modes are: • TRUE = Modulo. • FALSE = No modulo.

4.1.14.3.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE if the function or function block is successfully executing.

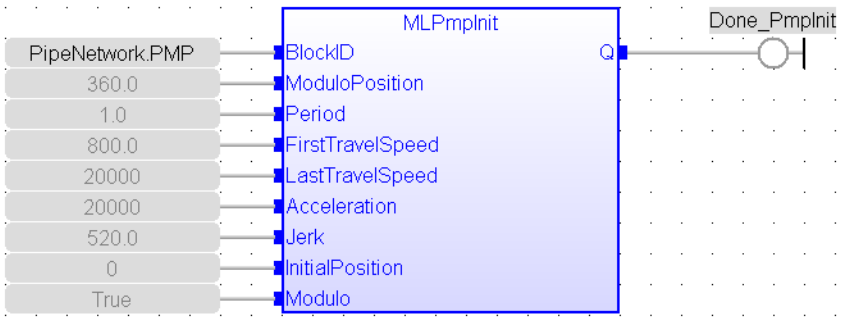
4.1.14.3.3 Remarks

- Initializes a PMP object (Parabolic Motion Profile generator) with user-defined settings.
- This function block is automatically called by the function Pipe Network(MLPN_CREATE_OBJECTS) if a PMP block is added to the Pipe Network.
  - User-defined settings are entered in the Pipe Blocks Properties screen.
- The PMP pipe block is assigned a Name, SAMPLING_PERIOD, MODULO_POSITION, FIRST_TRAVEL_SPEED, LAST_TRAVEL_SPEED, ACCELERATION, JERK, and INITIAL Position.
- Some of these parameters can be changes in an application program using other MLPmp function blocks.
- An **MLPmpRel** function block is used to make a bi-directional motion.
  - First movement in one direction, then a return motion back to the initial position.
- An **MLPmpAbs** function block is used to move one direction to an absolute position.

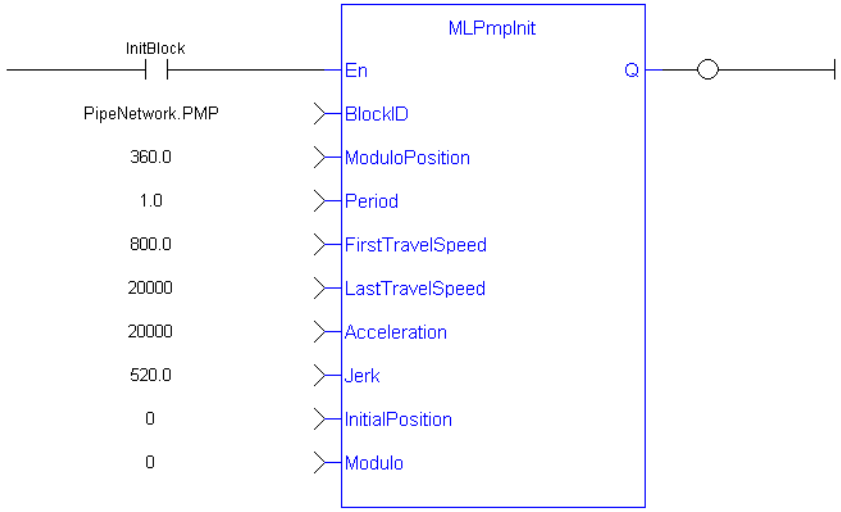
NOTE

PMP objects are normally created in the Pipe Network using the graphical engine.
You do not have to add **MLPmpInit** function blocks to their programs.
Parameters are entered directly in pop-up windows and the code is automatically added to the current project.

4.1.14.3.4 FBD Language Example



4.1.14.3.5 FFLD Language Example



4.1.14.3.6 IL Language Example

Not available.

4.1.14.3.7 ST Language Example

```
//Initialize  a PMP Pipe Block named "PMP" to a modulo roll over of 360,
motion generator sample period of 1,First Travel Speed of 800.0, Second
Travel Speed of 20000.0, Accel of 20000.0,Jerk of 520.0, Initial position of
0.0
MLPmpInit( PipeNetwork.Pmp , 360.0, 1.0, 800.0, 20000.0, 20000.0, 520.0, 0,
true  ) ;
```

See Also

- "MLPmpReadAccel" (→ p. 365)
- "MLPmpReadFstSpd" (→ p. 366)
- "MLPmpReadInitPos" (→ p. 367)
- "MLPmpReadJerk" (→ p. 368)
- "MLPmpReadLstSpd" (→ p. 369)

4.1.14.4 MLPmpReadAccel

Pipe Network

Function - Gets the Acceleration parameter of a PMP block.

4.1.14.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a PMP function block in the Pipe Network.

4.1.14.4.2 Outputs

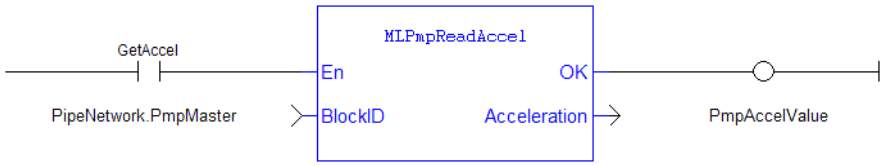
Output	Data Type	Range	Unit	Description
Acceleration	LREAL		User unit/sec^2	Present Acceleration of the PMP Pipe Network function block. • Value defined PMP block when creating a Pipe Network. • The value is entered in the ACCELERATION field in the Parameter tab.

4.1.14.4.3 Remarks

Used in both the **MLPmpAbs** and **MLPmpRel** function blocks.

4.1.14.4.4 FBD Language Example

4.1.14.4.5 FFLD Language Example



4.1.14.4.6 IL Language Example

Not available.

4.1.14.4.7 ST Language Example

```
PmpAccelValue :=  MLPmpReadAccel( PipeNetwork.PmpMaster )  ;
```

See Also

- "MLPmpReadFstSpd" (→ p. 366)
- "MLPmpReadInitPos" (→ p. 367)
- "MLPmpReadJerk" (→ p. 368)
- "MLPmpReadLstSpd" (→ p. 369)

4.1.14.5 MLPmpReadFstSpd

Pipe Network

Function - Gets the FirstTravelSpeed parameter of a PMP block.

4.1.14.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a PMP function block in the Pipe Network.

4.1.14.5.2 Outputs

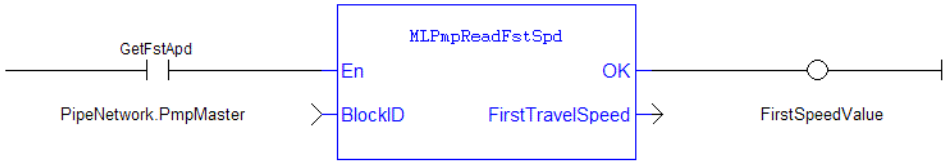
Output	Data Type	Range	Unit	Description
FirstTravelSpeed	LREAL		User unit/sec	Present first travel speed of the PMP Pipe Network function block. • Value defined in the setup of a PMP block in a Pipe Network. • The value is entered in the FIRST_TRAVEL_SPEED field in the Parameter tab.

4.1.14.5.3 Remarks

- This parameter is used as:
  - The first of two speeds in an **MLPmpRel** function block.
  - The speed in an **MLPmpAbs** function block.

4.1.14.5.4 FBD Language Example

4.1.14.5.5 FFLD Language Example



4.1.14.5.6 IL Language Example

Not available.

4.1.14.5.7 ST Language Example

```
FirstSpeedValue := MLPmpReadFstSpd( PipeNetwork.PmpMaster )  ;
```

See Also

- "MLPmpReadAccel" (→ p. 365)
- MLPmpReadFstSpd
- "MLPmpReadInitPos" (→ p. 367)
- "MLPmpReadJerk" (→ p. 368)
- "MLPmpReadLstSpd" (→ p. 369)
- "MLPmpWriteLstSpd" (→ p. 377)

4.1.14.6 MLPmpReadInitPos

Pipe Network

Function - Gets the Initial Position parameter of a PMP block.

4.1.14.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a PMP function block in the Pipe Network.

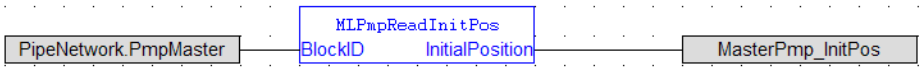
4.1.14.6.2 Outputs

Output	Data Type	Range	Unit	Description
InitialPosition	LREAL		User units	Present Initial Position of the PMP Pipe Network function block. • Value defined in the setup of a PMP block in a Pipe Network. • The value is entered in the INITIAL_ POSITION field in the Parameter tab.

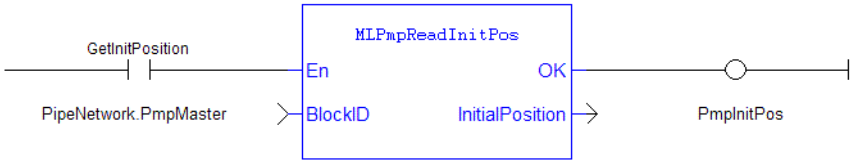
4.1.14.6.3 Remarks

- This value is the position the PMP block starts at when the Pipe Network is enabled.
- Can be set when adding a PMP block to a Pipe Network and defining the parameters for that block.

4.1.14.6.4 FBD Language Example



4.1.14.6.5 FFLD Language Example



4.1.14.6.6 IL Language Example

Not available.

4.1.14.6.7 ST Language Example

```
PmpInitPos := MLPmpReadInitPos( PipeNetwork.PmpMaster )  ;
```

See Also

"MLPmpInit" (→ p. 363)

4.1.14.7 MLPmpReadJerk

Pipe Network

Function - Gets the Jerk parameter of a PMP block.

4.1.14.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a PMP function block in the Pipe Network.

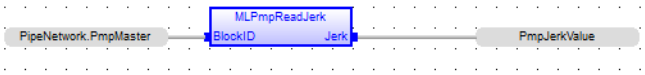
4.1.14.7.2 Outputs

Output	Data Type	Range	Unit	Description
Jerk	LREAL		User unit/sec^3	Jerk of the PMP Pipe Network function block. • Value defined in the setup of a PMP block in a Pipe Network. • The value is entered in the JERK field in the Parameter tab.

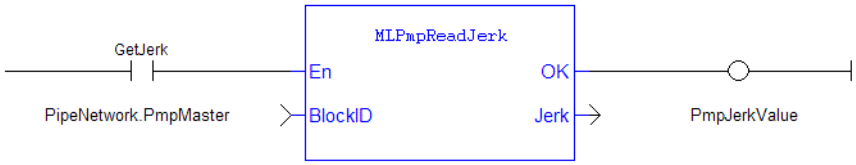
4.1.14.7.3 Remarks

Used in both the **MLPmpAbs** and **MLPmpRel** function blocks.

4.1.14.7.4 FBD Language Example



4.1.14.7.5 FFLD Language Example



4.1.14.7.6 IL Language Example

Not available.

4.1.14.7.7 ST Language Example

```
PmpJerkValue := MLPmpReadJerk( PipeNetwork.PmpMaster )  ;
```

4.1.14.8 MLPmpReadLstSpd

Pipe Network

Function - Gets the LastTravelSpeed parameter of a PMP block.

4.1.14.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a PMP function block in the Pipe Network.

4.1.14.8.2 Outputs

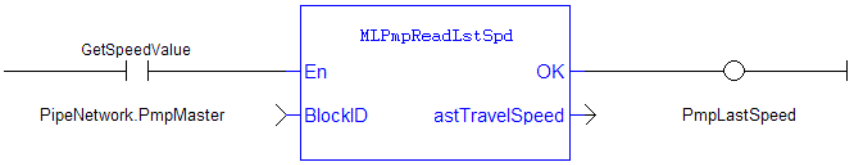
Output	Data Type	Range	Unit	Description
LastTravelSpeed	LREAL		User unit/sec	Last Travel Speed of the PMP Pipe Network function block. • Value defined in the setup of a PMP block in a Pipe Network. • The value is entered in the LAST_TRAVEL_SPEED field in the Parameter tab.

4.1.14.8.3 Remarks

Used in the **MLPmpRel** function block.

4.1.14.8.4 FBD Language Example

4.1.14.8.5 FFLD Language Example



4.1.14.8.6 IL Language Example

Not available.

4.1.14.8.7 ST Language Example

```
PmpLastSpeed := MLPmpReadLstSpd( PipeNetwork.PmpMaster )  ;
```

See Also

- "MLPmpReadAccel" (→ p. 365)
- "MLPmpReadFstSpd" (→ p. 366)
- "MLPmpReadInitPos" (→ p. 367)
- "MLPmpReadJerk" (→ p. 368)

4.1.14.9 MLPmpRel

Pipe Network

Function - Used to perform two subsequent relative moves.

4.1.14.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a PMP function block in the Pipe Network.
DeltaFirst	LREAL	No range	User units	Value defined in the setup of a PMP block in a Pipe Network.	Length of the first move. The value is entered in the FIRST_TRAVEL_SPEED field in the Parameter tab.
DeltaSecond	LREAL	No range	User units	Value defined in the setup of a PMP block in a Pipe Network.	Length of the second (return) move. The value is entered in the LAST_TRAVEL_SPEED field in the Parameter tab.

4.1.14.9.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE if the function or function block successfully completed.

4.1.14.9.3 Remarks

- Using the **MLPmpRel** function block, the PMP Generator produces forward-backward motions with a non-stop, jerk-free transition through zero speed.
  - See the example image "PMP Generator Forward and Backward Motion Profile" (→ p. 371).
  - This feature is useful for linear axes which must move back and forward without any pause at one end.
- This function is used to do a single relative move, ending in zero speed, by setting the **DeltaSecond** argument to zero (0.0).
  - If it is done, for the controlling speed to be the first move, the **Last_Travel_Speed** parameter must be set equal to or greater than the **First_Travel_Speed** parameter.
- The slower of the two speeds is utilized to optimize the S-curve behavior for the move whether it is a 2 or 1 delta move.
- If the **DeltaSecond** argument is non-zero, it must have the opposite sign than the sign of the **DeltaFirst** argument.

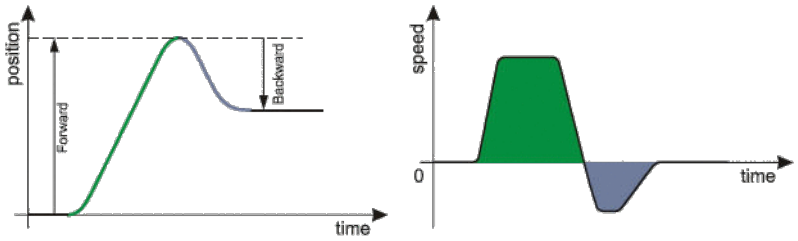
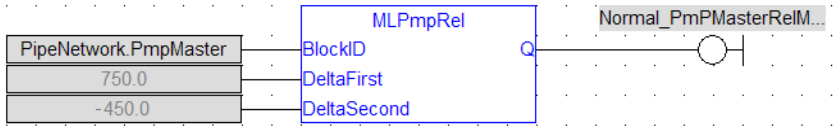
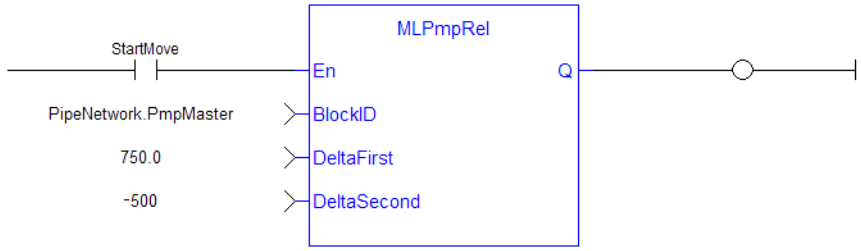


Figure 4-37: PMP Generator Forward and Backward Motion Profile

4.1.14.9.4 FBD Language Example



4.1.14.9.5 FFLD Language Example



4.1.14.9.6 IL Language Example

Not available.

4.1.14.9.7 ST Language Example

```
//Execute a Relative move on a PMP Block name "PmpMaster" with a First Travel
Speed of 750.0, Second Travel Speed of -500,
MLPmpRel( PipeNetwork.PmpMaster, 750 , -500);
```

See Also

- "MLPmpWriteAccel" (→ p. 374)
- "MLPmpWriteFstSpd" (→ p. 375)
- "MLPmpWriteJerk" (→ p. 376)
- "MLPmpWriteLstSpd" (→ p. 377)

4.1.14.10 MLPmpRun

Pipe Network ✓

Function - Jog the generator at the specified speed.

4.1.14.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables execution. Is only recognized if the PMP generator is Idle or at constant velocity as determined from the "MLPmpStatus" (→ p. 373) function.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a PMP function block in the Pipe Network.
Speed	LREAL	No range	User unit/sec	No default	The designated rate to Jog at. • If the speed is 0.0 User unit/sec, the PMP block decelerates to zero speed and switches to the Idle state (0). • See Set the Axis Block Position Units for more information.

4.1.14.10.2 Outputs

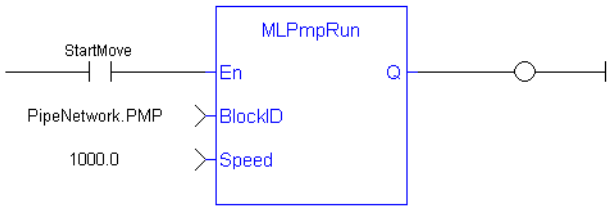
Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE if the function or function block successfully completed.

4.1.14.10.3 Remarks

None

4.1.14.10.4 FBD Language Example

4.1.14.10.5 FFLD Language Example



4.1.14.10.6 IL Language Example

Not available.

4.1.14.10.7 ST Language Example

```
//Execute a Relative move on a PMP Block name "PmpMaster" with a Jog speed of
1000.0
MLPmpRun( PipeNetwork.PmpMaster, 1000.0 ) ;
```

See Also

"MLPmpStatus" (→ p. 373)

4.1.14.11 MLPmpStatus

Pipe Network

Function - Returns the status of the PMP block generator.

4.1.14.11.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL	0, 1	N/A	No default	Enables execution.
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a PMP function block in the Pipe Network.

4.1.14.11.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL		N/A	Returns TRUE when the function successfully executes.
Default (.Q)	DINT		N/A	Returns the status of the PMP block generator.

Value	Description
0	Indicates the PMP block is idle. No command is currently running in the generator. It can be used to determine that a previous move is complete.
1	Indicates the PMP block is either accelerating to a position or speed, or is decelerating to a position or speed.
2	Indicates the PMP block is running at a constant speed.

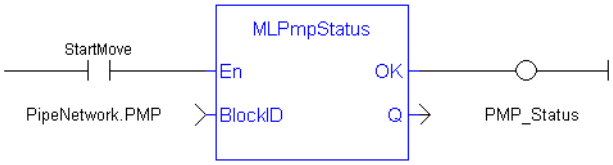
4.1.14.11.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

4.1.14.11.4 FBD Language Example

4.1.14.11.5 FFLD Language Example



4.1.14.11.6 IL Language Example

Not available.

4.1.14.11.7 ST Language Example

```
PMP_Status := MLPmpStatus ( PipeNetwork.PmpMaster )  ;
  Done :=TRUE;
```

4.1.14.12 MLPmpWriteAccel

Pipe Network

Function - Sets the Acceleration parameter of a PMP block.

4.1.14.12.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a PMP function block in the Pipe Network.
Acceleration	LREAL	> 0 (zero)	User unit/sec^2	Value defined in the setup of a PMP block in a Pipe Network.	Acceleration value. The value is entered in the ACCELERATION field in the Parameter tab.

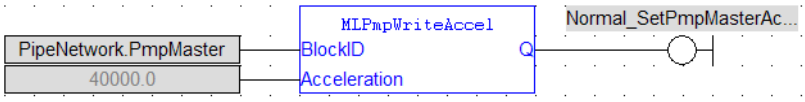
4.1.14.12.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE if the function or function block is successfully executing.

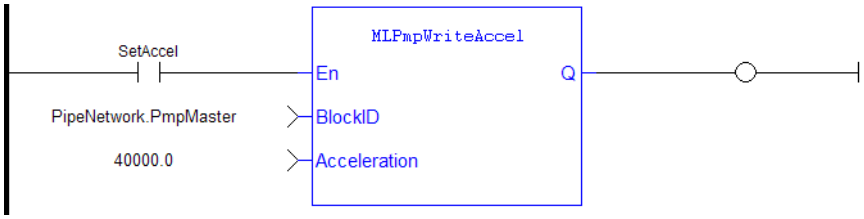
4.1.14.12.3 Remarks

- Can be set when adding a PMP block to a Pipe Network and defining the parameters for that block.
- Returns TRUE if the function succeeded.

4.1.14.12.4 FBD Language Example



4.1.14.12.5 FFLD Language Example



4.1.14.12.6 IL Language Example

Not available.

4.1.14.12.7 ST Language Example

```
MLPmpWriteAccel( PipeNetwork.PmpMaster, 40000.0 )  ;
```

See Also

- "MLPmpAbs" (→ p. 361)
- "MLPmpRel" (→ p. 370)
- "MLPmpWriteFstSpd" (→ p. 375)
- "MLPmpWriteJerk" (→ p. 376)
- "MLPmpWriteLstSpd" (→ p. 377)

4.1.14.13 MLPmpWriteFstSpd

Pipe Network

Function - Sets the FirstTravelSpeed parameter of a PMP block.

4.1.14.13.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a PMP function block in the Pipe Network.
FirstTravelSpeed	LREAL	> 0 (zero)	User unit/sec	Value defined in the setup of a PMP block in a Pipe Network.	First Travel Speed value. The value is entered in the FIRST_TRAVEL_ SPEED field in the Parameter tab.

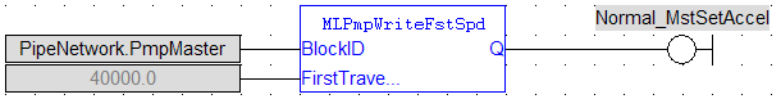
4.1.14.13.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE if the function or function block is successfully executing.

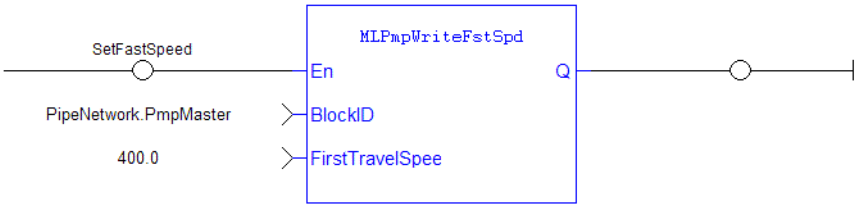
4.1.14.13.3 Remarks

- Can be set when adding a PMP block to a Pipe Network and defining the parameters for that block.
- Returns TRUE if the function succeeded.

4.1.14.13.4 FBD Language Example



4.1.14.13.5 FFLD Language Example



4.1.14.13.6 IL Language Example

Not available.

4.1.14.13.7 ST Language Example

```
MLPmpWriteFstSpd( PipeNetwork.PmpMaster, 300.0 )  ;
```

See Also

- "MLPmpAbs" (→ p. 361)
- "MLPmpRel" (→ p. 370)
- "MLPmpWriteAccel" (→ p. 374)
- "MLPmpWriteJerk" (→ p. 376)
- "MLPmpWriteLstSpd" (→ p. 377)

4.1.14.14 MLPmpWriteJerk

Pipe Network

Function - Sets the Jerk parameter of a PMP block.

4.1.14.14.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a PMP function block in the Pipe Network.
Jerk	LREAL	> 0 (zero)	User unit/sec^3	Value defined in the setup of a PMP block in a Pipe Network.	Jerk value. The value is entered in the JERK field in the Parameter tab.

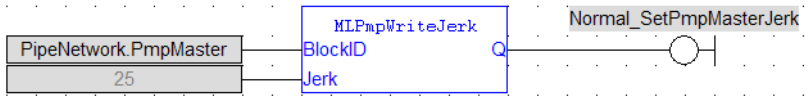
4.1.14.14.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE if the function or function block is successfully executing.

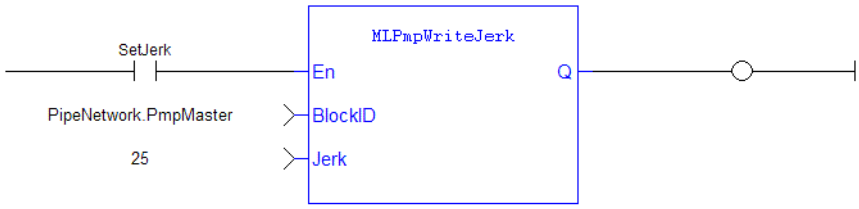
4.1.14.14.3 Remarks

- Can be set when adding a PMP block to a Pipe Network and defining the parameters for that block.
- Returns TRUE if the function succeeded.

4.1.14.14.4 FBD Language Example



4.1.14.14.5 FFLD Language Example



4.1.14.14.6 IL Language Example

Not available.

4.1.14.14.7 ST Language Example

```
MLPmpWriteJerk( PipeNetwork.PmpMaster, 15.0 )  ;
```

See Also

- "MLPmpAbs" (→ p. 361)
- "MLPmpReadJerk" (→ p. 368)
- "MLPmpRel" (→ p. 370)
- "MLPmpWriteAccel" (→ p. 374)
- "MLPmpWriteFstSpd" (→ p. 375)
- "MLPmpWriteLstSpd" (→ p. 377)

4.1.14.15 MLPmpWriteLstSpd

Pipe Network

Function - Sets the LastTravelSpeed parameter of a PMP block.

4.1.14.15.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of a PMP function block in the Pipe Network.
LastTravelSpeed	LREAL	> 0 (zero)	User unit/sec	Value defined in the setup of a PMP block in a Pipe Network.	Last Travel Speed value. The value is entered in the LAST_TRAVEL_ SPEED field in the Parameter tab.

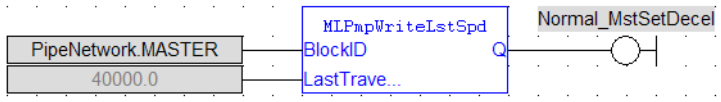
4.1.14.15.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE if the function or function block is successfully executing.

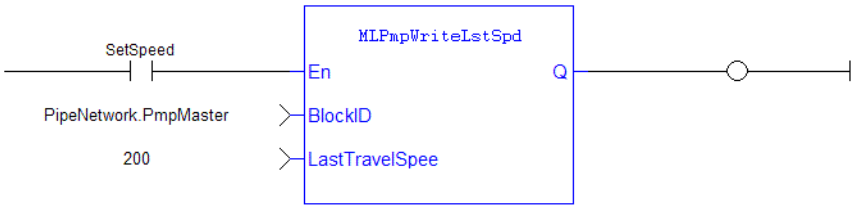
4.1.14.15.3 Remarks

- Can be set when adding a PMP block to a Pipe Network and defining the parameters for that block.
- Returns TRUE if the function succeeded.

4.1.14.15.4 FBD Language Example



4.1.14.15.5 FFLD Language Example



4.1.14.15.6 IL Language Example

Not available.

4.1.14.15.7 ST Language Example

```
MLPmpWriteLstSpd( PipeNetwork.PmpMaster, 650 )  ;
```

See Also

- "MLPmpAbs" (→ p. 361)
- "MLPmpReadLstSpd" (→ p. 369)
- "MLPmpRel" (→ p. 370)
- "MLPmpWriteAccel" (→ p. 374)
- "MLPmpWriteFstSpd" (→ p. 375)
- "MLPmpWriteJerk" (→ p. 376)

4.1.15 Sampler

Name	Description
"MLSmpConECAT" (→ p. 379)	Connects a sampler block to the specified CoE object in a PDO.
"MLSmpConnect" (→ p. 381)	Connects a sampler block to a pipe network axis or pipe block.
"MLSmpConPLCAxis" (→ p. 382)	Connects a sampler block to a specific PLCopen axis variable.
"MLSmpConPNAxis" (→ p. 383)	Connects a sampler block to a specific Pipe Network axis variable.
"MLSmpInit" (→ p. 384)	Initializes a sampler object.

TIP

There is a delay when using an external encoder.
The delay is five cycles:
2 cycles to read the encoder from the AKD via EtherCAT,

1 cycle for computing,
2 cycles for sending the new position set point to the AKD).
This lag error is speed proportional (5 cycles * speed).
A Phaser block can be used to compensate for this lag.

4.1.15.1 MLSmpConECAT

Function - Connects a sampler block to the specified CoE object in a PDO.

4.1.15.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID Name of the SMP function block in the Pipe Network.
DeviceAddr	INT	No range	N/A	No default	The EtherCAT address of the slave device. • The first node usually has the value 1001. • Alternately, use the members of the EtherCAT structure to specify a device's address.
Index	UINT	No range	N/A	No default	The CoE index of the object to be connected with the Sampler block.
SubIndex	UINT	No range	N/A	No default	The CoE sub-index of the object to be connected with the Sampler block.

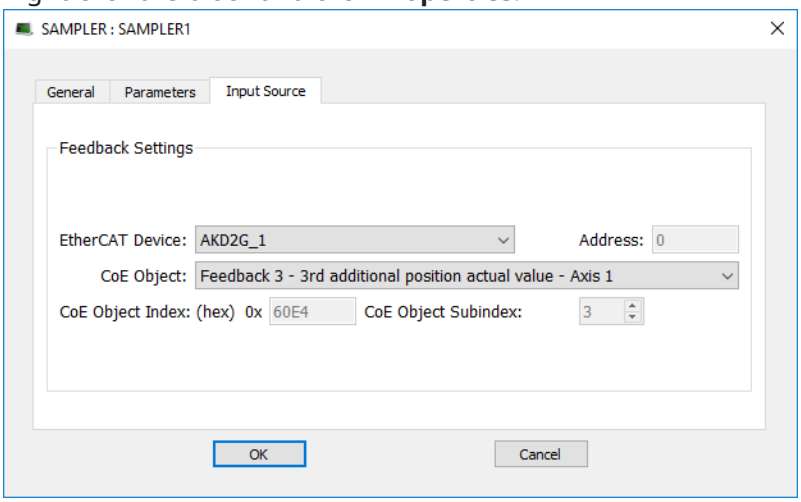
4.1.15.1.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Function block is operational.

4.1.15.1.3 Remarks

- The CoE object must be included in a PDO for the specified EtherCAT device.
- Use any EtherCAT data source as input for the specified sampler block.
- If the device providing the CoE Object for the Sampler block is removed:
  - The CoE object value becomes 0 (zero) if the EtherCAT device providing the CoE object for the Sampler block is removed using the "ECATDeviceAction" (→ p. 40) function block.
  - The CoE object automatically connects to the sampler block when the device is reconnected to the EtherCAT network.

- This function can be programmed from within the Pipe Network block. Right-click the block and click **Properties**.



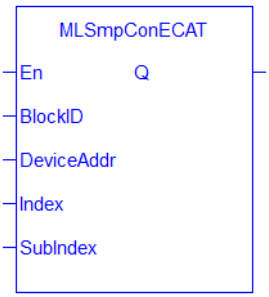
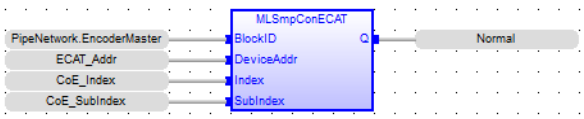
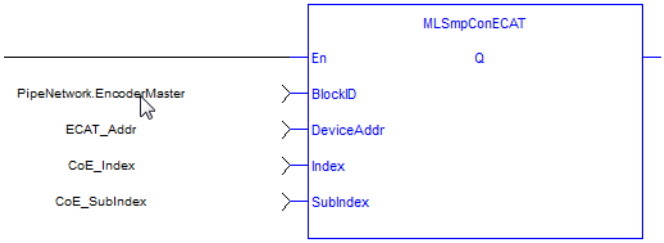


Figure 4-38: MLSmpConECAT

4.1.15.1.4 FBD Language Example



4.1.15.1.5 FFLD Language Example



4.1.15.1.6 IL Language Example

Not available.

4.1.15.1.7 ST Language Example

```
//Connect a Sampler pipe block named "EncoderMaster" to an ECAT Index Object
defined by  variable "CoE_SubIndex" with the SubIndex defined by variable
"CoE_SubIndex", from a device with  Ethercat Address defined by "ECAT_Addr"
MLSmpConECAT(PipeNetwork.EncoderMaster, ECAT_Addr, CoE_Index, CoE_SubIndex );
```

See Also

- "MLSmpConnect" (→ p. 381)
- "MLSmpInit" (→ p. 384)

4.1.15.2 MLSmpConnect

Pipe Network

Function - Connects a sampler block to a pipe network axis or pipe block.

4.1.15.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID Name of the SMP function block in the Pipe Network.
PipeBlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID Name of the pipe block the sampler is connected to.

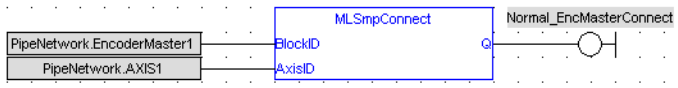
4.1.15.2.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE if the Sampler is connected.

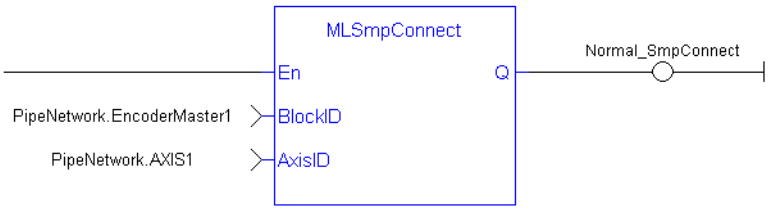
4.1.15.2.3 Remarks

None

4.1.15.2.4 FBD Language Example



4.1.15.2.5 FFLD Language Example



4.1.15.2.6 IL Language Example

Not available.

4.1.15.2.7 ST Language Example

```
//Connect a Sampler pipe block named "EncoderMaster1" to a PipeNetwork Axis
block named AXIS1
MLSmpConnect( PipeNetwork.EncoderMaster1, PipeNetwork.AXIS1 ) ;
```

See Also

- "MLSmpConECAT" (→ p. 379)
- "MLSmpConPLCAxis" (→ p. 382)
- "MLSmpConPNAxis" (→ p. 383)
- "MLSmpInit" (→ p. 384)

4.1.15.3 MLSmpConPLCAxis

Pipe Network

Function - Connects a sampler block to a specific PLCopen axis variable.

4.1.15.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID Name of the SMP function block in the Pipe Network.
AxisID	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function. See Modify PLCopen Axis for more information.
Variable	UINT	N/A Use available macros.	N/A	No default	Variable to be connected to. Use one of these Internal Defines: • MC_ACTUAL_POSITION • MC_COMMAND_POSITION • MC_NORMAL_COMMAND_POSITION • MC_PHASE_COMMAND_POSITION • MC_SUPERIMPOSED_COMMAND_POSITION

4.1.15.3.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Function block is operational.

4.1.15.3.3 Remarks

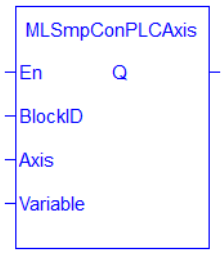
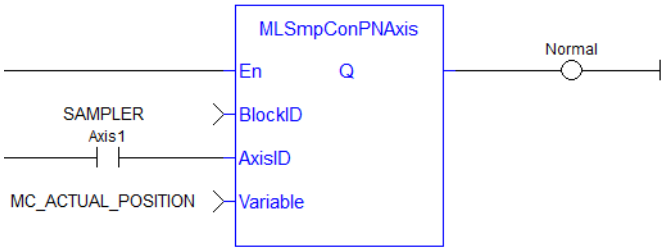


Figure 4-39: MLSmpConPLCAxis

4.1.15.3.4 FBD Language Example

4.1.15.3.5 FFLD Language Example



4.1.15.3.6 IL Language Example

Not available.

4.1.15.3.7 ST Language Example

```
//Connect a Sampler pipe block named "SAMPLER" to a variable named "MC_
ACTUAL_POSITION" from a PLCOpen Axis named Axis1.
MLSmpConPLCAxis( PipeNetwork.SAMPLER, Axis1, MC_ACTUAL_POSITION);
```

4.1.15.4 MLSmpConPNAxis

Pipe Network

Function - Connects a sampler block to a specific Pipe Network axis variable.

4.1.15.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID Name of the SMP function block in the Pipe Network.
AxisID	DINT	-2147483648 to 2147483647	N/A	No default	ID Name of the axis the sampler is connected to.
Variable	UINT	N/A Use available macros.	N/A	No default	Variable to be connected to. Use one of these Internal Defines: • ML_ACTUAL_POSITION • ML_ACTUAL_TORQUE • ML_ACTUAL_VELOCITY • ML_CURRENT_POSITION • ML_FEEDBACK_POSITION • ML_FOLLOWING_ERROR • ML_GENERATOR_POSITION • ML_PIPE_POSITION • ML_REFERENCE_POSITION

4.1.15.4.2 Outputs

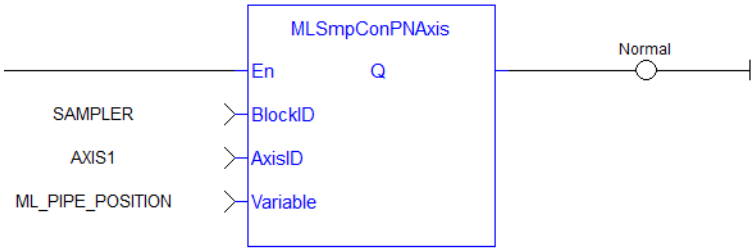
Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Function block is operational.

4.1.15.4.3 Remarks

Figure 4-40: MLSmpConPNAxis function

4.1.15.4.4 FBD Language Example

4.1.15.4.5 FFLD Language Example



4.1.15.4.6 IL Language Example

Not available.

4.1.15.4.7 ST Language Example

```
//Connect a sampler block named "SAMPLER" to a variable named ML_PIPE_
POSITION  from a Pipe Network Axis named PipeNetwork.AXIS1
MLSmpConPNAxis( PipeNetwork.SAMPLER , PipeNetwork.AXIS1, ML_PIPE_POSITION );
```

4.1.15.5 MLSmpInit

Pipe Network

Function - Initializes a sampler object.

4.1.15.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID Name of the SMP function block in the Pipe Network.
SamplingPeriod	LREAL	0.25 to ?	Millisecond	1.0	Period that the device is sampled.
Mode	DINT	1, 2 Position or Speed	N/A	Position	Sampled output can be either Position or Velocity.

Input	Data Type	Range	Unit	Default	Description
InputModuloPosition	LREAL	No range	User units	360.0	Period of the input signal. The value set depends upon the device used. • **AKD**: This should be set equal to 232 (4294967296.0). • **AKD2G**: The value should be set based on the feed constant value assigned in the EtherCAT object 0x60E4 subindex 1-5. • The default feed constant value is 216 (65536)
OutputModuloPosition	LREAL	No range	User units	360.0	Period of the output signal.

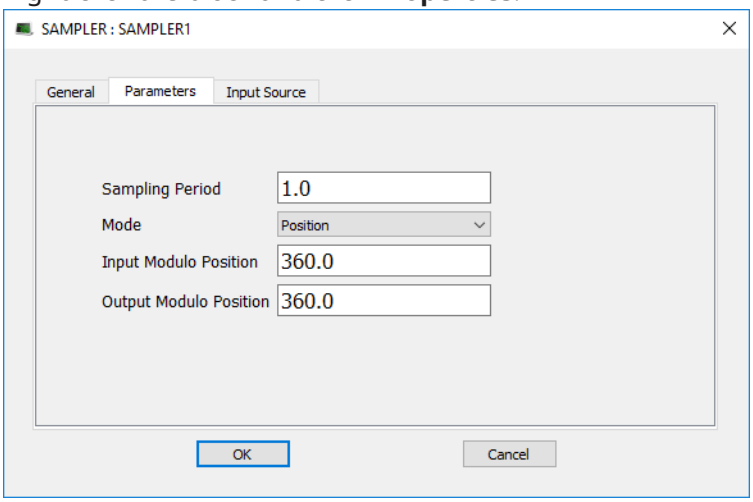
4.1.15.5.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	SMP Block successfully initiated.

4.1.15.5.3 Remarks

- This sampler block is used to periodically sample and place into a pipe some output of a source object.
  - The sampled output can be the POSITION or SPEED of a source object measured by a resolver, an encoder, or some other types of sensor.
- The sampler implements the logical connection between:
  - An encoder on a physical master axis (the source object).
  - One or more pipes.
  - It performs the function of periodically sampling the source and placing the sampled values into the pipe.
- This function block is automatically called by the Function PipeNetwork(MLPN_CREATE_OBJECTS) if a Smp Block is added to the Pipe Network, with user-defined settings entered in the Pipe Blocks Properties screen.
- The Smp Pipe Block is assigned a Name, SAMPLING_PERIOD, MODE, INPUT_VALUE_PERIOD, and OUTPUT_VALUE_PERIOD.

- This function can be programmed from within the Pipe Network block. Right-click the block and click **Properties**.

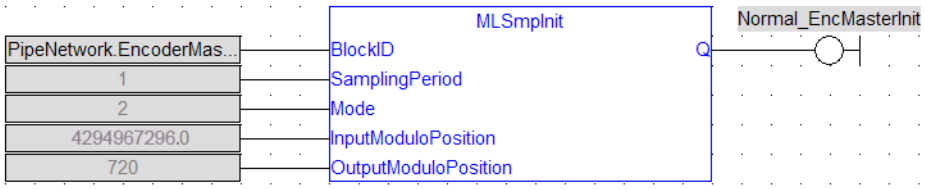


- To offset the Sampler Block Output Position in the Pipe Network either:
  - Place a Phaser Block (and write "MLPhaWritePhase" (→ p. 358) in the application code).
  - Place a Gear Block (and write "MLGearWriteOff" (→ p. 323)) after the Sampler Block.

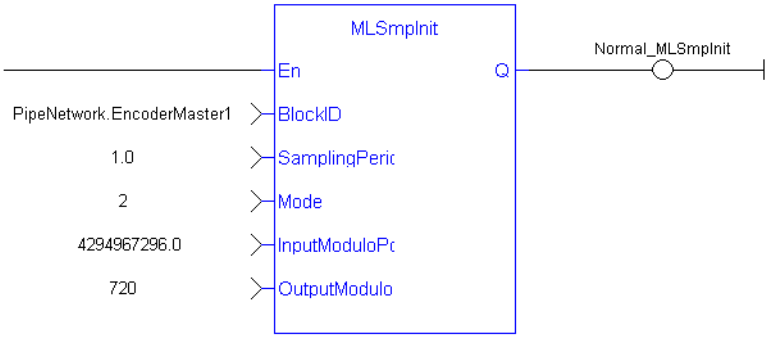
Using AKD Secondary Feedback	Using AKD2G Additional Feedback

• The Sampler can be connected to the secondary feedback on the AKD using "MLSmpConPNAxis" (→ p. 383).
• The scaling for the AKD Secondary Feedback is setup using AKD Parameters: DRV.HANDWHEEL and FB2.ENCRES .
• The feedback signal comes through EtherCAT in object 0x2050.
• The scaling for this position signal is 0 to 4294967296 = 0 to FB2.ENCRES.
  • Object 0x2050 rolls over to 0 when reaching 4294967296.
	• The Sampler can be connected to the additional feedback (1-5) on the AKD2G using "MLSmpConPNAxis" (→ p. 383).
	  • See the AKD2G Feedback Settings section for setting up the additional feedback type and resolution.
	• The default feed constant value in the KAS-IDE is scaled to 65536.
	  • This feed constant value can be changed using EtherCAT object 0x60E9 subindex 1-5.
	• The feedback signal comes through EtherCAT in object 0x60E4 subindex 1-5.

4.1.15.5.4 FBD Language Example



4.1.15.5.5 FFLD Language Example



4.1.15.5.6 IL Language Example

Not available.

4.1.15.5.7 ST Language Example

```
//Initialize  a Sampler Pipe Block named "EncoderMaster1" to a Sample Period
of 1 millisec, Mode of Operation to 2(Velocity), Input Modulo of 4294967296,
and Output Modulo of 720
MLSmpInit( PipeNetwork.EncoderMaster1, 1.0,2,4294967296,720);
```

See Also

- "MLSmpConECAT" (→ p. 379)
- "MLSmpConnect" (→ p. 381)
- "MLSmpConPLCAxis" (→ p. 382)
- "MLSmpConPNAxis" (→ p. 383)

4.1.16 Synchronizer

TIP

- See "Example: Synchronizer Functions" (→ p. 387) for more information.

Name	Description
MLSyncInit	Initializes a synchronizer Pipe Block.
MLSyncReadDeltaS	Gets the output phasing value of a synchronizer block.
MLSyncStart	Starts a synchronization of a synchronizer Pipe Block.
MLSyncStop	De-synchronizes a synchronizer Pipe Block.
MLSyncWriteDeltaS	Sets the output phasing value of a synchronizer block.

4.1.16.1 Example: Synchronizer Functions

When **MLSyncStart** function is called, the output value is adapted to catch up with the input value.

The OUTPUT_PHASING parameter is also used to define a curve in order to smooth the output value.

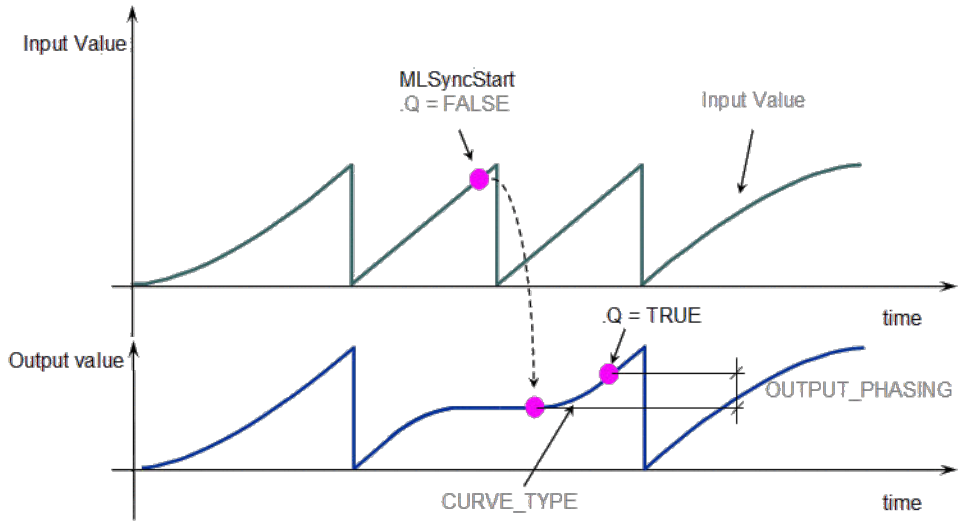


Figure 4-41: Get Output Phasing after **MLSyncStart**

When the **MLSyncStop** function is called, the output value is adapted according to the specified Stop-Position (point B).

The OUTPUT_PHASING parameter is used to define point A, where the flow follows a curve in order to smooth the output value.

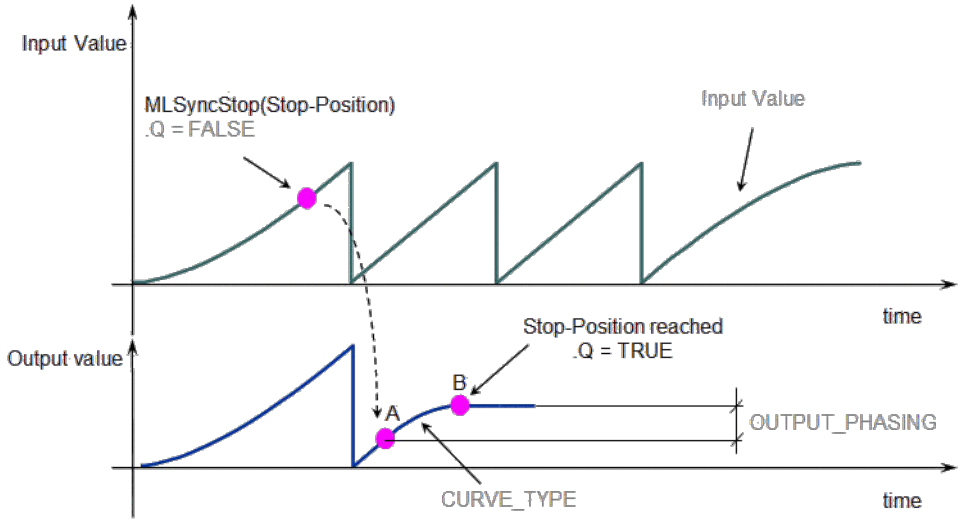


Figure 4-42: Get Output Phasing after **MLSyncStop**

4.1.16.2 MLSyncInit

Pipe Network

Function - Initializes a synchronizer Pipe Block.

4.1.16.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Pipe Network Block.

Input	Data Type	Range	Unit	Default	Description
ModuloPosition	LREAL	No range	User units	No default	The modulo distance.
CurveType	DINT	1, 2	N/A	No default	The curve type to the motion when starting and stopping synchronization. The options are: • 1 = Parabolic • 2 = Polynomial
DeltaS	LREAL	No range	User units	No default	The Distance to get in or out of synchronization. This parameter is used in "MLSyncStart" (→ p. 391) and "MLSyncStop" (→ p. 392).

4.1.16.2.2 Outputs

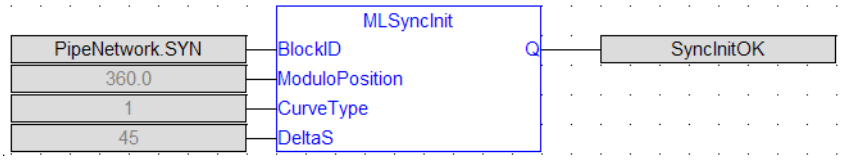
Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Function block executed successfully.

4.1.16.2.3 Remarks

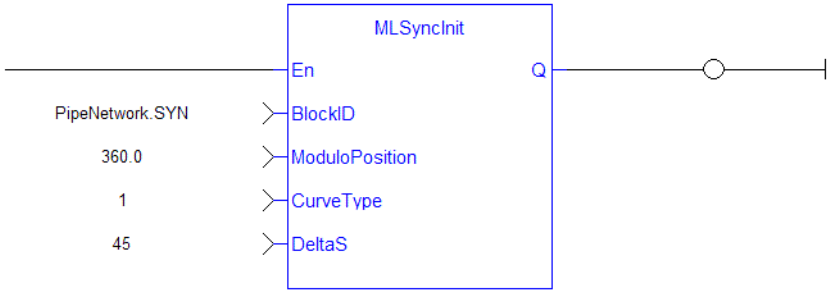
- Returns TRUE if the function succeeded.
- This FB is automatically created in the compiled code of a Pipe Network.
- This function block is part of the MLPN_CREATE_OBJECT to initialize the Pipe Network.
  - It is called at the beginning of an application program with the function call:

```
PipeNetwork(MLPN_CREATE_OBJECTS);
```

4.1.16.2.4 FBD Language Example



4.1.16.2.5 FFLD Language Example



4.1.16.2.6 IL Language Example

Not available.

4.1.16.2.7 ST Language Example

```
//Initialize  a synchronizer  Pipe Block named " SYN" with a modulo of 360,
Curve Type of Parabolic, and a distance (DeltaS) of 30 to get in and out of
synchronization
MLSyncInit( PipeNetwork.SYN, 360, 1, 30 );
```

See Also

"MLSyncWriteDeltaS" (→ p. 393)

4.1.16.3 MLSyncReadDeltaS

Pipe Network ✓

Function - Gets the output phasing value of a synchronizer block.

4.1.16.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Pipe Network Block.

4.1.16.3.2 Outputs

Output	Data Type	Range	Unit	Description
DeltaS	LREAL		User units	Present Delta Slope value.

4.1.16.3.3 Remarks

- Output phasing is the distance or the slope the output takes to synchronize with the input when "MLSyncStart" (→ p. 391) block is executed.
  - See the "Get Output Phasing after MLSyncStart" (→ p. 390) image.
- It also affects the distance or the slope the output takes to de-synchronize with the input and come to a stop when "MLSyncStop" (→ p. 392) block is executed.
  - See the "Get Output Phasing after MLSyncStop" (→ p. 391) image.

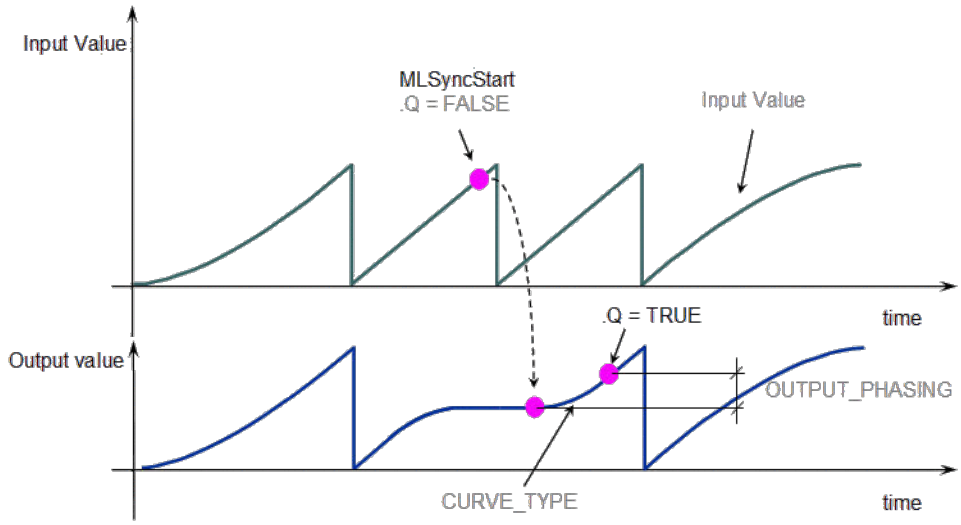


Figure 4-43: Get Output Phasing after **MLSyncStart**

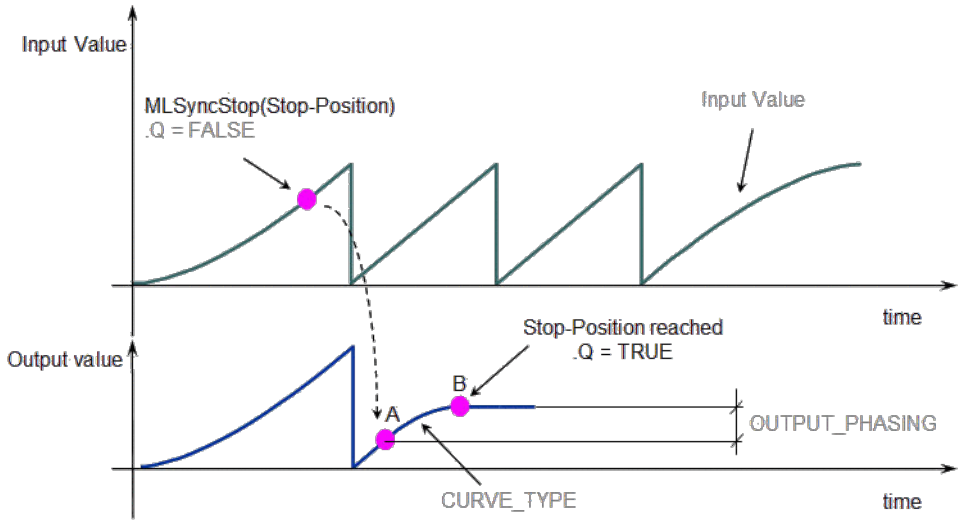
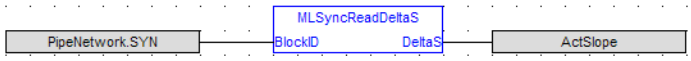
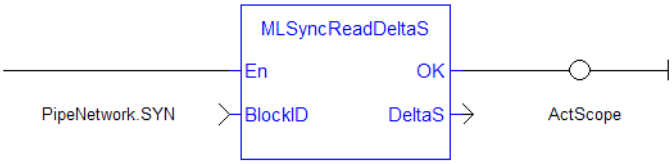


Figure 4-44: Get Output Phasing after **MLSyncStop**

4.1.16.3.4 FBD Language Example



4.1.16.3.5 FFLD Language Example



4.1.16.3.6 IL Language Example

Not available.

4.1.16.3.7 ST Language Example

```
ActScope := MLSyncReadDeltaS( PipeNetwork.SYN );
```

See Also

"MLSyncWriteDeltaS" (→ p. 393)

4.1.16.4 MLSyncStart

Pipe Network

Function - Starts a synchronization of a synchronizer Pipe Block.

4.1.16.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Pipe Network Block.

4.1.16.4.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Function block executed successfully.

4.1.16.4.3 Remarks

Returns TRUE if the function succeeded.

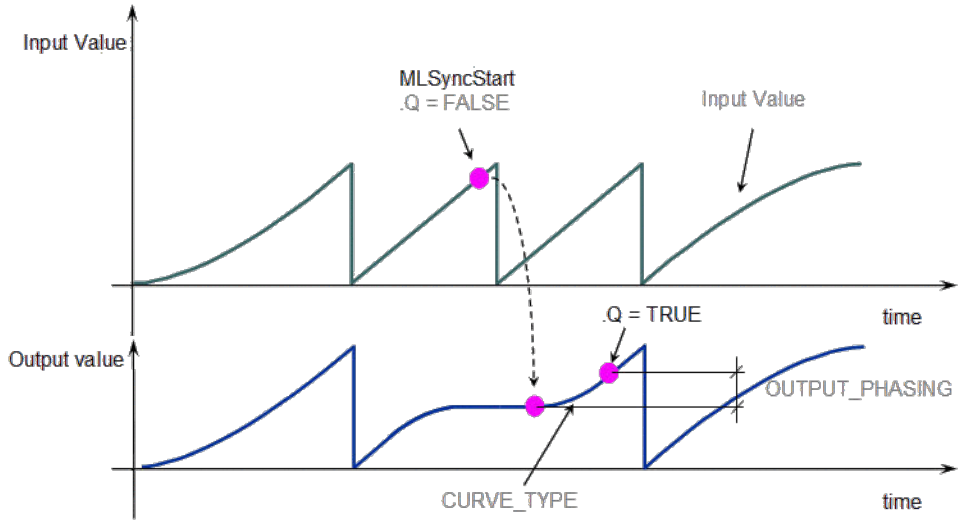
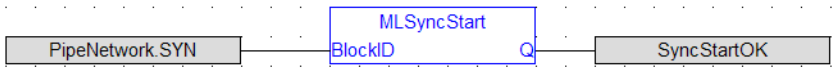


Figure 4-45: MLSyncStart

4.1.16.4.4 FBD Language Example



4.1.16.4.5 FFLD Language Example

4.1.16.4.6 IL Language Example

Not available.

4.1.16.4.7 ST Language Example

```
MLSyncStart( PipeNetwork.SYN );
```

4.1.16.5 MLSyncStop

Pipe Network

Function - De-synchronizes a synchronizer Pipe Block.

4.1.16.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Position	LREAL	No range	User units	No default	Motion Stop Position.

4.1.16.5.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Function block executed successfully.

4.1.16.5.3 Remarks

Returns TRUE if the function succeeded.

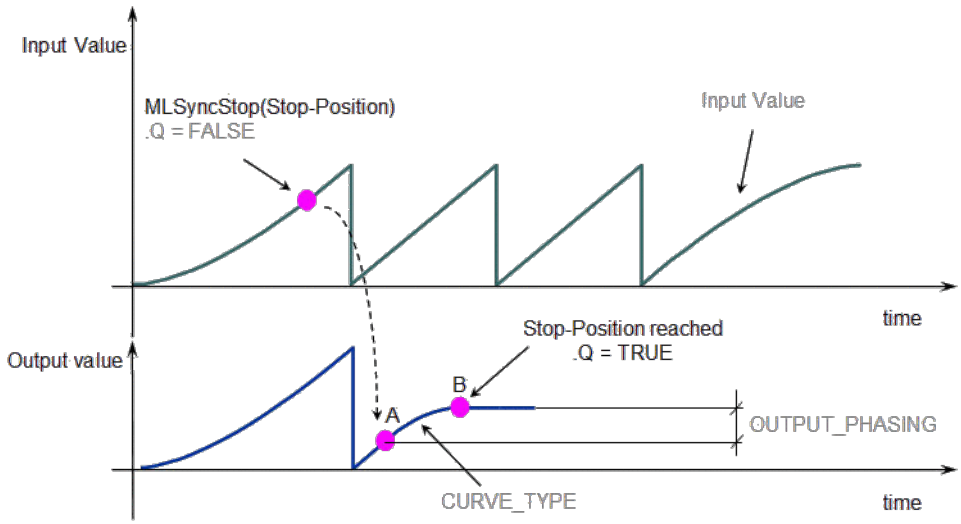
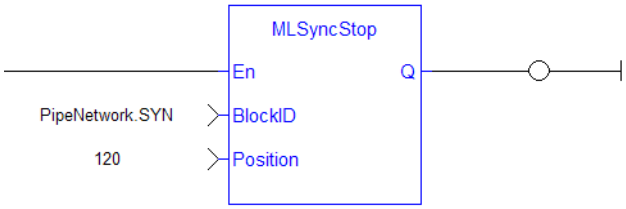


Figure 4-46: MLSyncStop

4.1.16.5.4 FBD Language Example

4.1.16.5.5 FFLD Language Example



4.1.16.5.6 IL Language Example

Not available.

4.1.16.5.7 ST Language Example

```
MLSyncStop( PipeNetwork.SYN , 120 );
```

4.1.16.6 MLSyncWriteDeltaS

Pipe Network

Function - Sets the output phasing value of a synchronizer block.

4.1.16.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID name of the Pipe Network Block.
DeltaS	LREAL	No range	User units	No default	Slope to be used during Start and stop of Synchronization. This parameter is used in "MLSyncStart" (→ p. 391) and "MLSyncStop" (→ p. 392).

4.1.16.6.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Function block executed successfully.

4.1.16.6.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- Returns TRUE if the function succeeded.
- Output phasing is the distance or the slope the output takes to synchronize with the input when MLSyncStart block is executed.
- It also affects the distance or the slope the output takes to de-synchronize with the input and come to a stop when MLSyncStop block is executed.

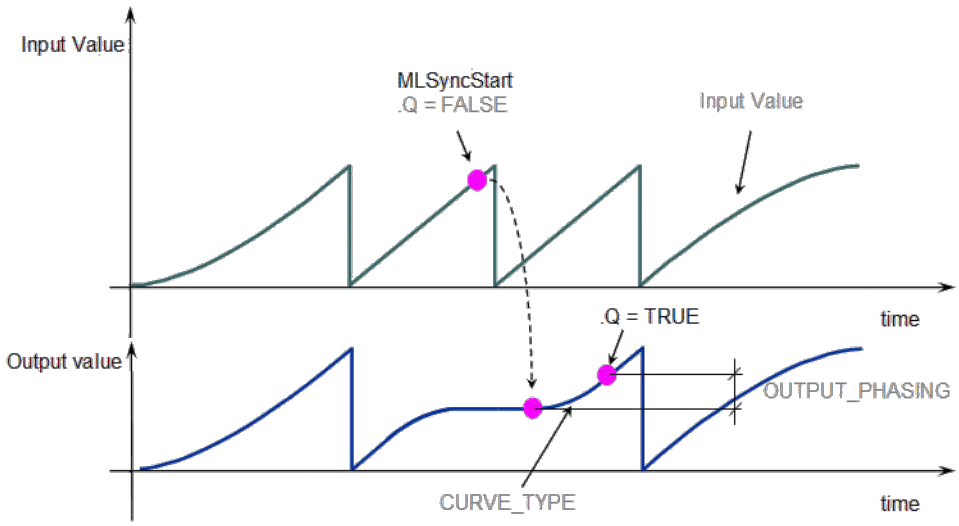


Figure 4-47: Set output phasing after MLSyncStart

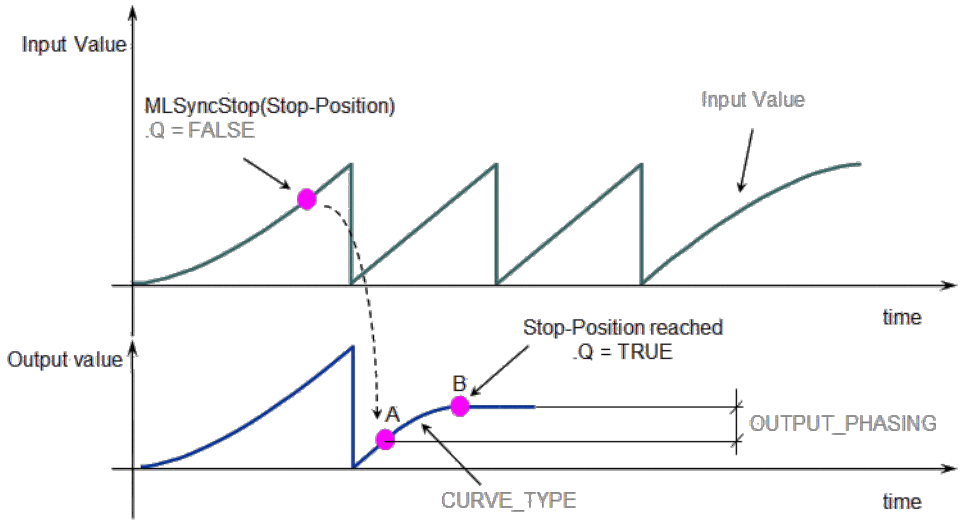
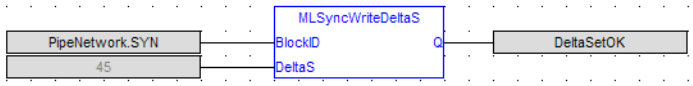
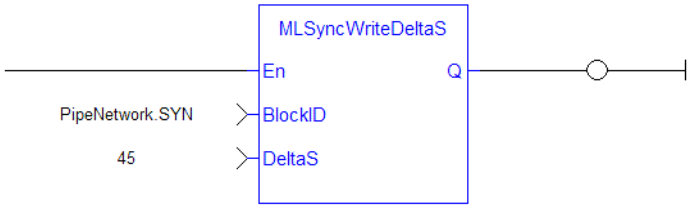


Figure 4-48: Set output phasing after MLSyncStop

4.1.16.6.4 FBD Language Example



4.1.16.6.5 FFLD Language Example



4.1.16.6.6 IL Language Example

Not available.

4.1.16.6.7 ST Language Example

```
MLSyncWriteDeltaS( PipeNetwork.SYN, 45 );
```

4.1.17 Trigger

TIP

- See "Example: Trigger Functions" (→ p. 396) for an example of Trigger functions.

Name	Description
MLTrigClearFlag	Clears the flag of an initiated Trigger block.
MLTrigInit	Initializes a Trigger object for use in a PLC program.
MLTrigIsTrigged	Checks if the selected block has been triggered.
MLTrigReadDelay	Returns the time the trigger block uses to compensate the delay of the sensor that captures the triggering signal.
MLTrigReadPos	Returns the position of the block at the moment when it was triggered.

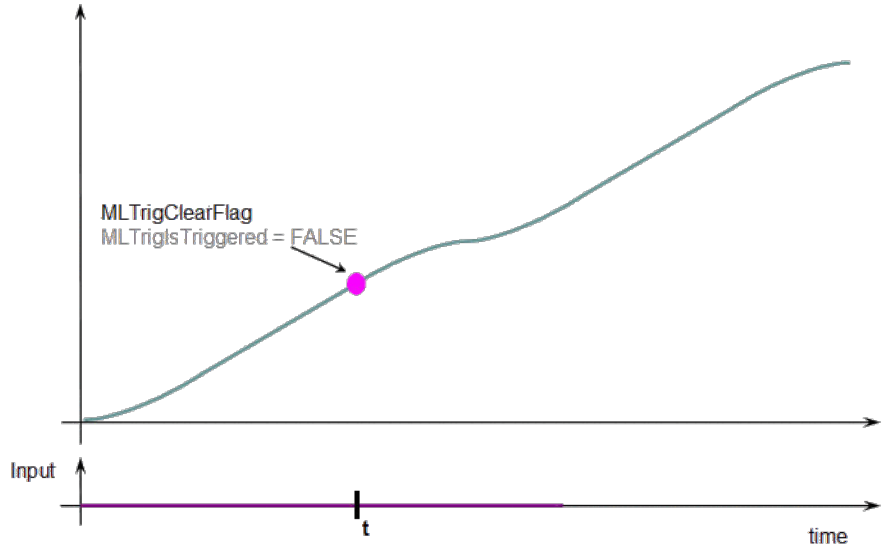
Name	Description
MLTrigReadTime	Returns the time of the moment where the block was triggered in milliseconds.
MLTrigSetEdge	Sets the edge configuration for a trigger block (e.g., rising, falling, etc.)
MLTrigWriteDelay	Sets the time the trigger block uses to compensate for the delay introduced by the sensor that captures the triggering signal.

4.1.17.1 Example: Trigger Functions

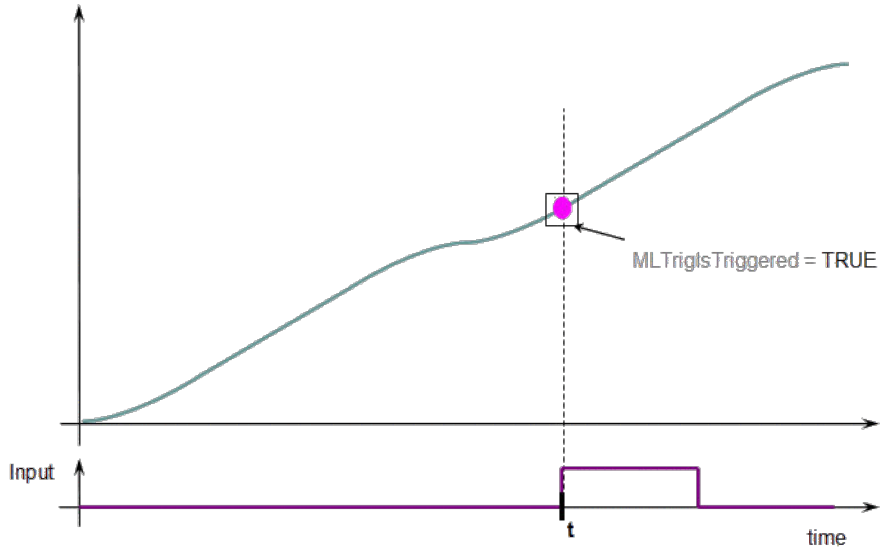
IMPORTANT

The trigger delay must be calculated by **you** and set with the "MLTrigWriteDelay" (→ p. 408) function block.
This delay belongs to the sensor and it is additional to the "MLTrigReadTime" (→ p. 405) / "MLTrigReadPos" (→ p. 403).

When you call the **MLTrigClearFlag** function, the flag for trigger is reset to False.



When a Fast Input is set, the **MLTrigIsTriggered** function returns TRUE.



Then you can call the **MLTrigReadPos** and **MLTrigReadTime** functions to get more details.

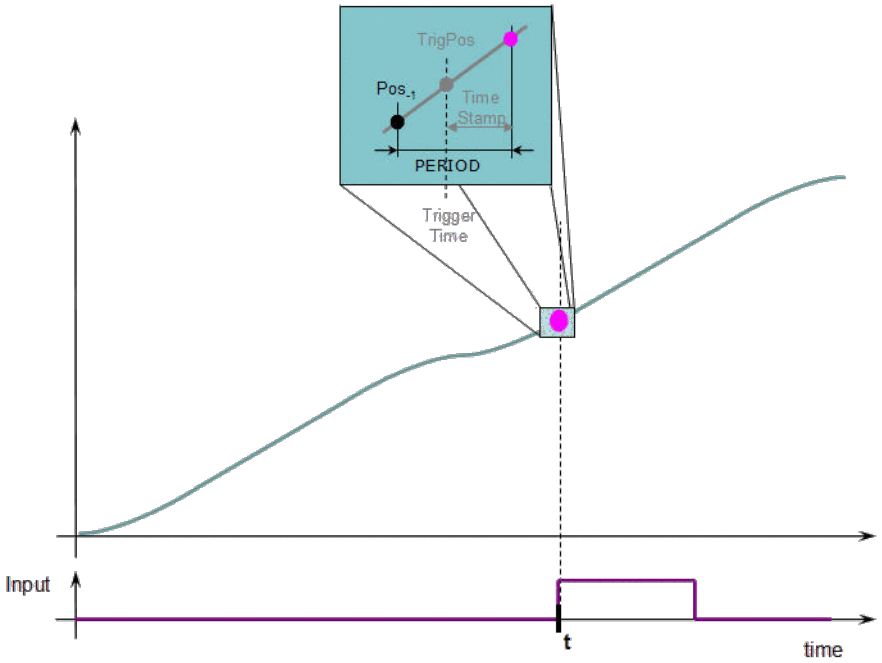


Figure 4-49: Trigger Functions Usage

4.1.17.2 MLTrigClearFlag

Pipe Network

Function - Clears the flag of an initiated Trigger block.

4.1.17.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Trigger object.

4.1.17.2.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE if the function block is executed.

4.1.17.2.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- The block captures the position and time of the next event.
- Once triggered, a block has to be reset before it can be triggered again.
  - All events sent to a block while in a triggered state are ignored.
  - The position and time information is lost.

IMPORTANT

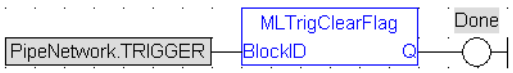
The Fast Input assigned to a Trigger block has to be reset before information on a new event can be captured.
MLAxisRstFastIn is used at the same time as **MLTrigClearFlag**.

4.1.17.2.3.1 Fast Homing - Inputs

See these topics for more information:

- Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block
- Fast Homing Example with the PLCopen Motion Engine
- Fast Inputs with Pipe Network Motion
- Pipe Network Registration and Fast Homing
- Registration Position Capture Example with Pipe Network Trigger Block

4.1.17.2.4 FBD Language Example



4.1.17.2.5 FFLD Language Example

4.1.17.2.6 IL Language Example

Not available.

4.1.17.2.7 ST Language Example

```
//Clear Trigger Flag
MLTrigClearFlag( PipeNetwork.TRIGGER );
```

See Also

- "MLAxisRstFastIn" (→ p. 266)
- "MLTrigIsTrigged" (→ p. 400)
- "MLTrigReadPos" (→ p. 403)
- "MLTrigReadTime" (→ p. 405)

4.1.17.3 MLTrigInit

Pipe Network

Function - Initializes a Trigger object for use in a PLC program.

4.1.17.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Trigger object.
Input_Axis	STRING	No range	N/A	No default	Name of the axis where the Fast Input is located.

Input	Data Type	Range	Unit	Default	Description
InputID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of the Fast Input. **InputID** INT Range = 0, 1 • 0 = Touch Probe 1 / Capture Engine 0 • 1 = Touch Probe 2 / Capture Engine 1
EdgeID	DINT	0, 2	N/A	1 (rising edge)	The edge to trigger on. • 0 = disable the Fast Input. • 1 = rising edge. • 2 = falling edge.

4.1.17.3.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE if the function block is executed.

4.1.17.3.3 Remarks

- This function block is automatically called if a Trigger Block is added to the Pipe Network, with user-defined settings entered in the Pipe Blocks Properties screen.
- The Trigger object monitors a selected Fast Input and captures the time of a rising or falling edge event.
  - With the time and pipe position information, the Trigger object extrapolates the axis position when the Fast Input event occurred.

- Parameters to enter include:
  - The name of the Pipe Block.
  - The Axis where the Fast Input is located.
  - The number of the desired Fast Input.
  - Whether to trigger on the rising or falling edge of the input.

NOTE

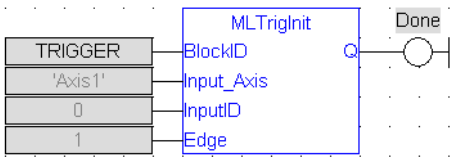
Trigger objects are normally created in the Pipe Network using the graphical engine.
You do not have to add MLTrigInit function blocks to their programs.
Parameters are entered directly in pop-up windows and the code is automatically added to the current project.

4.1.17.3.3.1 Fast Homing - Inputs

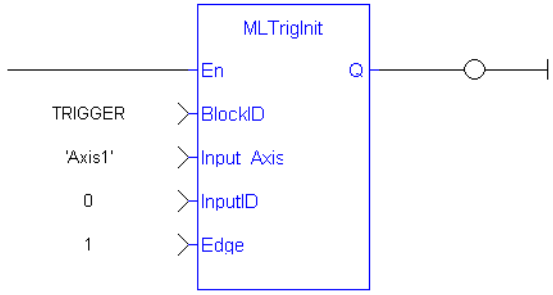
See these topics for more information:

- Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block
- Fast Homing Example with the PLCopen Motion Engine
- Fast Inputs with Pipe Network Motion
- Pipe Network Registration and Fast Homing
- Registration Position Capture Example with Pipe Network Trigger Block

4.1.17.3.4 FBD Language Example



4.1.17.3.5 FFLD Language Example



4.1.17.3.6 IL Language Example

Not available.

4.1.17.3.7 ST Language Example

```
//Create and Initiate a Trigger Pipe Block named "Trigger" and set it up to
receive the trigger signal from Axis1, capture engine 0, and the rising edge
of the signal
TRIGGER := MLBlkCreate( 'TRIGGER', 'TRIGGER' );
MLTrigInit( TRIGGER, 'Axis1', 0, 1 );
```

See Also

- "MLAxisRstFastIn" (→ p. 266)
- "MLTrigClearFlag" (→ p. 397)
- "MLTrigIsTrigged" (→ p. 400)
- "MLTrigReadPos" (→ p. 403)

4.1.17.4 MLTrigIsTrigged

Pipe Network

Function - Checks if the selected block has been triggered.

4.1.17.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Trigger object.

4.1.17.4.2 Outputs

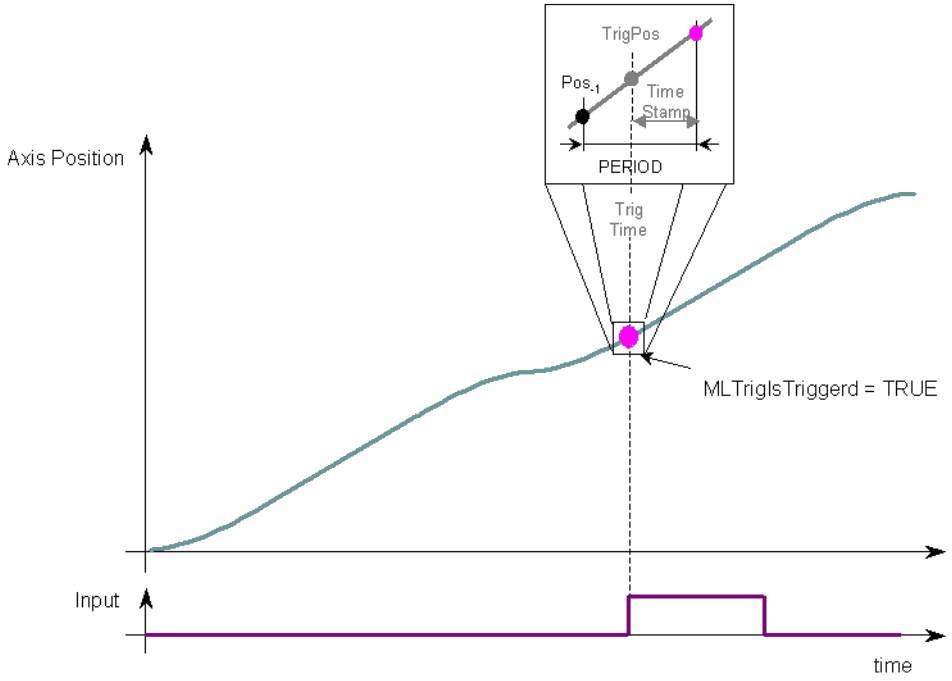
Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE if the selected Trigger Object has triggered.

4.1.17.4.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- When a block has been triggered, it contains the time and position when a Fast Input event occurred.
  - The application has to reset the block before the block can be triggered again.
- All trigger events that are sent to the block during its triggered state are lost.
- Once triggered, a block has to be reset before it can be triggered again.
  - All events sent to a block while in a triggered state are ignored.
  - The position and time information is lost.



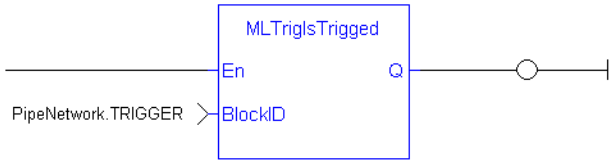
4.1.17.4.3.1 Fast Homing - Inputs

See these topics for more information:

- Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block
- Fast Homing Example with the PLCopen Motion Engine
- Fast Inputs with Pipe Network Motion
- Pipe Network Registration and Fast Homing
- Registration Position Capture Example with Pipe Network Trigger Block

4.1.17.4.4 FBD Language Example

4.1.17.4.5 FFLD Language Example



4.1.17.4.6 IL Language Example

Not available.

4.1.17.4.7 ST Language Example

```
//Check if a Trigger Block has been triggered, then save position
IF MLTrigIsTrigged( PipeNetwork.TRIGGER ) THEN
Trig_Position := MLTrigReadPos( PipeNetwork.TRIGGER );
END_IF
```

See Also

- "MLTrigReadPos" (→ p. 403)
- "MLTrigReadTime" (→ p. 405)

4.1.17.5 MLTrigReadDelay

Function - Returns the time the trigger block uses to compensate the delay of the sensor that captures the triggering signal.

4.1.17.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	Identifier of the trigger block.
En	BOOL	No range	N/A	No default	Enables execution.

4.1.17.5.2 Outputs

Output	Data Type	Range	Unit	Description
Delay	LREAL	No range	Microseconds	Value of the delay compensation currently applied by the trigger block.
OK	BOOL	No range	N/A	Returns TRUE when the function successfully executes.

4.1.17.5.3 Remarks

- Returns the delay programmed in a trigger block by the "MLTrigWriteDelay" (→ p. 408) function to compensate for the reaction time required by the sensor.
- Electronic sensors are not able to respond immediately to a signal.
- Sensors usually require a certain amount of time to process a change of state in their input signal.

4.1.17.5.3.1 Fast Homing - Inputs

See these topics for more information:

- Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block
- Fast Homing Example with the PLCopen Motion Engine
- Fast Inputs with Pipe Network Motion
- Pipe Network Registration and Fast Homing
- Registration Position Capture Example with Pipe Network Trigger Block

4.1.17.5.4 FBD Language Example

Not available.

4.1.17.5.5 FFLD Language Example

Not available.

4.1.17.5.6 IL Language Example

Not available.

4.1.17.5.7 ST Language Example

Not available.

See Also

"MLTrigWriteDelay" (→ p. 408)

4.1.17.6 MLTrigReadPos

Pipe Network ✓

Function - Returns the position of the block at the moment when it was triggered.

4.1.17.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Trigger object.

4.1.17.6.2 Outputs

Output	Data Type	Range	Unit	Description
Position	LREAL	No range	User units	Returns the position of the selected block's Axis at the moment when it was triggered.

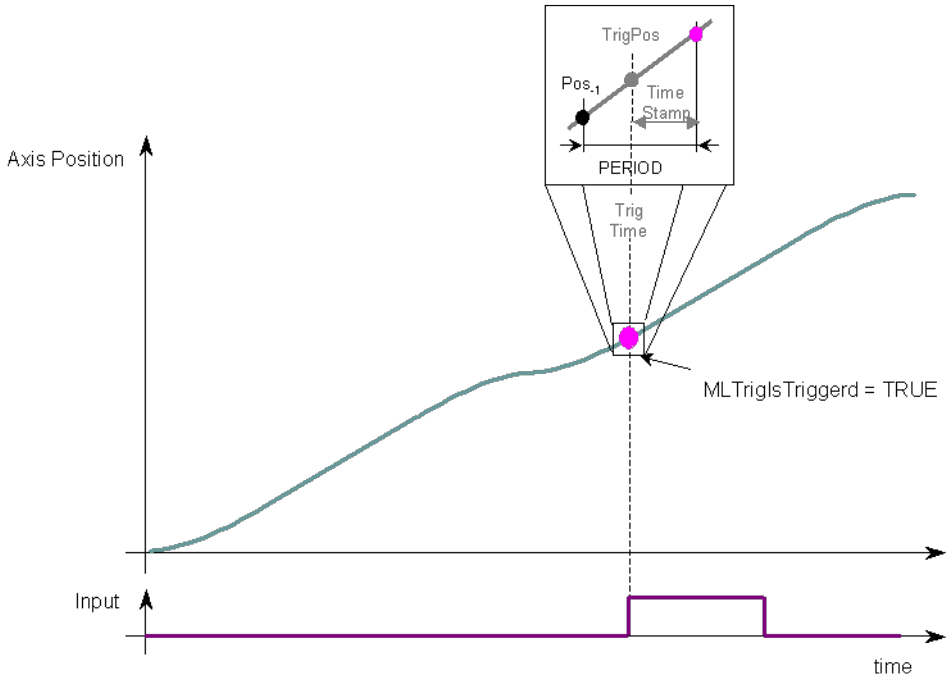
4.1.17.6.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- Returns the modulo-applied position of the pipe at the moment it is triggered by the Trigger Block's selected Fast Input.
- This value is only valid when TrigIsTrigged() returns TRUE.
  - The Trigger block extrapolates the output value based on the timestamp of the Fast Input event to provide an accurate position even if the event occurs in the middle of a program cycle.

- Once triggered, a block has to be reset before it can be triggered again.
  - All events sent to a block while in a triggered state are ignored.
  - The position and time information is lost.



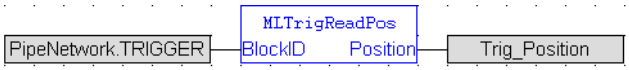
- **Modulo Calculation**: MLTrigReadPos uses the Output Modulo Position value of the previous block in the pipe, even if the previous pipe is configured for No Modulo mode.
  - The previous block must specify a 0 (zero) value for Output Modulo Position before setting the Mode to No Modulo to prevent a modulo operation for MLTrigReadPosPipe.
- Previous Function Name: MLTrigGetPos

4.1.17.6.3.1 Fast Homing - Inputs

See these topics for more information:

- Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block
- Fast Homing Example with the PLCopen Motion Engine
- Fast Inputs with Pipe Network Motion
- Pipe Network Registration and Fast Homing
- Registration Position Capture Example with Pipe Network Trigger Block

4.1.17.6.4 FBD Language Example



4.1.17.6.5 FFLD Language Example

4.1.17.6.6 IL Language Example

Not available.

4.1.17.6.7 ST Language Example

```
//Save position of Axis when Fast Input event occurs
Trig_Position := MLTrigReadPos( PipeNetwork.TRIGGER );
```

See Also

- "MLAxisRstFastIn" (→ p. 266)
- "MLTrigClearFlag" (→ p. 397)
- "MLTrigIsTrigged" (→ p. 400)
- "MLTrigReadTime" (→ p. 405)

4.1.17.7 MLTrigReadTime

Function - Returns the time of the moment where the block was triggered in milliseconds.

4.1.17.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initiated Trigger object.

4.1.17.7.2 Outputs

Output	Data Type	Range	Unit	Description
Time	LREAL	No range	Microseconds	Returns the time the Trigger Block's selected Fast Input was triggered.

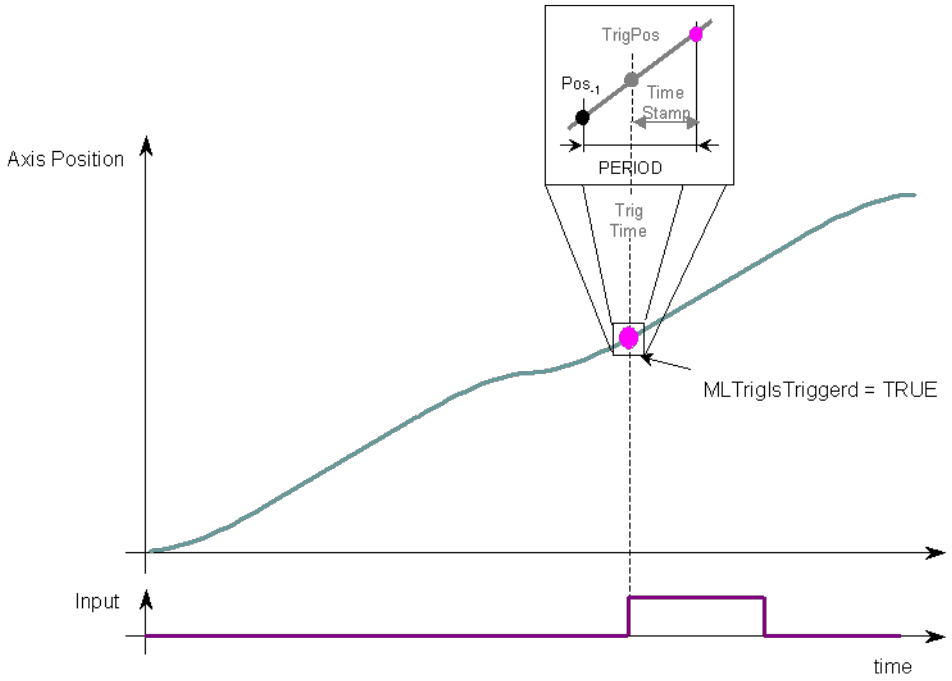
4.1.17.7.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- This value is only valid when TrigIsTrigged() returns TRUE.
  - The output is computed from the timestamp of a Fast Input time event.

- Once triggered, a block has to be reset before it can be triggered again.
  - All events sent to a block while in a triggered state are ignored.
  - The position and time information is lost.



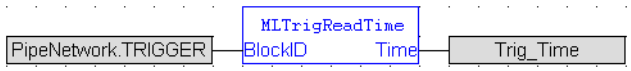
- Previous Function Name: MLTrigGetTime

4.1.17.7.3.1 Fast Homing - Inputs

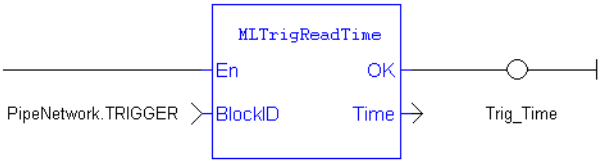
See these topics for more information:

- Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block
- Fast Homing Example with the PLCopen Motion Engine
- Fast Inputs with Pipe Network Motion
- Pipe Network Registration and Fast Homing
- Registration Position Capture Example with Pipe Network Trigger Block

4.1.17.7.4 FBD Language Example



4.1.17.7.5 FFLD Language Example



4.1.17.7.6 IL Language Example

Not available.

4.1.17.7.7 ST Language Example

```
//Save time when Fast Input event occurs
Trig_Time := MLTrigReadTime( PipeNetwork.TRIGGER );
```

See Also

- "MLAxisRstFastIn" (→ p. 266)
- "MLTrigClearFlag" (→ p. 397)
- "MLTrigIsTrigged" (→ p. 400)
- "MLTrigReadPos" (→ p. 403)

4.1.17.8 MLTrigSetEdge

Pipe Network

Function - Sets the edge configuration for a trigger block (e.g., rising, falling, etc.)

4.1.17.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of the trigger block.
Edge	DINT	0, 2	N/A	1 (rising edge)	The edge to trigger on. • 0 = disable the Fast Input. • 1 = rising edge. • 2 = falling edge.

4.1.17.8.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL	No range	N/A	• TRUE if block executed successfully. • FALSE if execution is not successful.

4.1.17.8.3 Remarks

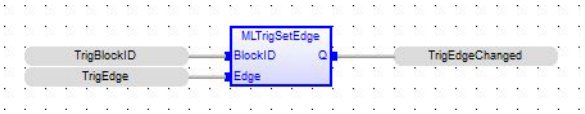
- This block should be called prior to calling "MLAxisCfgFastIn" (→ p. 233).
- The value at the Edge input must match the value at MLAxisCfgFastIn's Mode input.

4.1.17.8.3.1 Fast Homing - Inputs

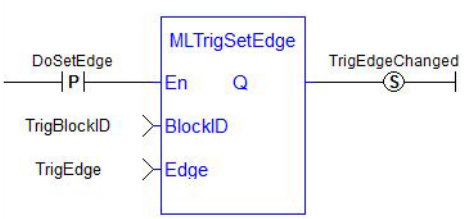
See these topics for more information:

- Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block
- Fast Homing Example with the PLCopen Motion Engine
- Fast Inputs with Pipe Network Motion
- Pipe Network Registration and Fast Homing
- Registration Position Capture Example with Pipe Network Trigger Block

4.1.17.8.4 FBD Language Example



4.1.17.8.5 FFLD Language Example



4.1.17.8.6 IL Language Example

Not available.

4.1.17.8.7 ST Language Example

```
TrigEdgeChanged := MLTrigSetEdge (TrigBlockID,TrigEdge);
```

4.1.17.9 MLTrigWriteDelay

Function - Sets the time the trigger block uses to compensate for the delay introduced by the sensor that captures the triggering signal.

4.1.17.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BlockID	DINT	-2147483648 to 2147483647	N/A	No default	Identifier of the trigger block.
Delay	LREAL	No range	Microseconds	No default	Reaction time of the sensor that the trigger block has to compensate.

4.1.17.9.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL	No range	N/A	Returns TRUE if the delay is successfully set. .

4.1.17.9.3 Remarks

- Allows the trigger block to calculate the exact moment at which a signal was triggered by the specified compensation.
- The delay compensation should include drive processing time, sensor delay, and the communication latency through the EtherCAT network.
- Electronic sensors are not able to respond immediately to a signal.
- Sensors usually require a certain amount of time to process a change of state in their input signal.

4.1.17.9.3.1 Fast Homing - Inputs

See these topics for more information:

- Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block
- Fast Homing Example with the PLCopen Motion Engine
- Fast Inputs with Pipe Network Motion
- Pipe Network Registration and Fast Homing
- Registration Position Capture Example with Pipe Network Trigger Block

4.1.17.9.4 FBD Language Example

Not available.

4.1.17.9.5 FFLD Language Example

Not available.

4.1.17.9.6 IL Language Example

Not available.

4.1.17.9.7 ST Language Example

Not available.

See Also

"MLTrigReadDelay" (→ p. 402)

4.2 Motion / PLCopen Library

These are the Motion/PLCopen functions and function blocks.

- "Control" (→ p. 409)
- "Information" (→ p. 409)
- "I/O" (→ p. 409)
- "PLCopenMotion" (→ p. 410)
- "Profile" (→ p. 410)
- "Reference" (→ p. 411)
- "Registration" (→ p. 411)

4.2.1 Control

Name	Description
MC_ClearFaults	Sends a request to the drive to clear any drive faults.
MC_CreatePLCAxis	Creates a PLCopen Axis.
MC_EStop	Causes an emergency stop (E-stop).
MC_InitAxis	Initializes a PLCopen Axis' data.
MC_InitAxisFeedback	Initializes a PLCopen digitizing Axis' position data.
MC_Power	Requests to either enable the drive and close the loop, or disable the drive and open the loop.
MC_ResetError	Resets the errors of the specified axis.
MC_Stop	Aborts the active move, removes the next move from the queue, performs a controlled stop, and switches the axis to Stopping state.

4.2.2 Information

Name	Description
MC_ReadActPos	Reads the actual position of the axis.
MC_ReadActVel	Reads the actual velocity of the axis.
MC_ReadAxisErr	Returns the error status of the specified axis.
MC_ReadBoolPar	Returns the value of the specified Boolean axis parameter.
MC_ReadParam	Returns the value of the specified axis parameter.
MC_ReadStatus	Returns the state of the specified axis.
MC_WriteBoolPar	Writes the specified axis Boolean parameter.
MC_WriteParam	Writes the specified axis parameter.

4.2.3 I/O

Name	Description
MC_AbortTrigger	When the Execute input transitions from low to high, this function block aborts an MC_TouchProbe function block.
MC_TouchProbe	Arms a Fast Input and returns the latched position when the Fast Input event occurs.

4.2.4 PLCopenMotion

Name	Description
MC_AddSuperAxis	Adds a Superimposed Axis to the Axis's list of assigned superimposed axes.
MC_Halt	Decelerates an axis to zero velocity.
MC_MoveAbsolute	Performs a single-axis move to a specified endpoint position.
MC_MoveAdditive	Performs a single-axis move for a specified distance from the endpoint of the previous move.
MC_MoveContVel	Performs a single-axis non-ending move at a specified velocity with the option of continually updating the ongoing motion with the current input parameters.
MC_MoveRelative	Performs a single-axis move for a specified distance to perform incremental motion.
MC_MoveSuperimp	Performs a single-axis move superimposed upon the currently executing move.
MC_MoveVelocity	Performs a single-axis, non-ending move at a specified velocity.
MC_RemSuperAxis	Remove an axis from the axis's list of assigned, superimposed axes.
MC_SetOverride	Writes velocity and acceleration override factors.

4.2.5 Profile

Name	Description
MC_CamIn	Performs a slave axis move which follows the master axis based on the Cam Table specified by CamTableID.
MC_CamOut	Disengages the slave axis from a MC_CamIn move.
MC_ CamResumePos	Returns the slave axis position for resuming an "MC_CamIn" (➔ p. 476) move.
MC_CamStartPos	Returns the slave axis position for starting an "MC_CamIn" (➔ p. 476) move.
MC_ CamTblSelect	Defined to read and initialize the specified profile.
MC_GearIn	Performs a slave axis move which follows the master axis based on the ratio specified by RatioNumerator and RatioDenominator.
MC_GearInPos	Performs a slave axis move which follows the master axis based on the ratio specified by RatioNumerator and RatioDenominator.
MC_GearOut	Disengages the slave axis from a MC_GearIn or MC_GearInPos move.
MC_Phasing	Performs a master position phase shift for the slave axis.

Name	Description
MC_SyncSlaves	Allows the application to specify what slave axes are to be synchronized and which master they follow.

4.2.6 Reference

Name	Description
MC_Reference	Used to execute a fast home to a switch.
MC_SetPos	Changes the present actual position of the axis (as reported by "MC_ReadActPos" (→ p. 436)) to the position specified by the **Position** and **Mode** inputs.
MC_SetPosition	Deprecated - See "MC_SetPos" (→ p. 514).

4.2.7 Registration

Name	Description
MC_MachRegist	Enables Mark-to-Machine registration.
MC_MarkRegist	Enables Mark-to-Mark registration.
MC_StopRegist	Turns off registration for the specified axis and disarms the specified fast input.

4.2.8 Control

These functions provide general controls to drives and axes.

Name	Description
MC_ClearFaults	Sends a request to the drive to clear any drive faults.
MC_CreatePLCAxis	Creates a PLCopen Axis.
MC_EStop	Causes an emergency stop (E-stop).
MC_InitAxis	Initializes a PLCopen Axis' data.
MC_InitAxisFeedback	Initializes a PLCopen digitizing Axis' position data.
MC_Power	Requests to either enable the drive and close the loop, or disable the drive and open the loop.
MC_ResetError	Resets the errors of the specified axis.
MC_Stop	Aborts the active move, removes the next move from the queue, performs a controlled stop, and switches the axis to Stopping state.

4.2.8.1 MC_ClearFaults

Function - Sends a request to the drive to clear any drive faults.

4.2.8.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Function enable - execute function. This Input must be on shot.
Axis	AXIS_REF	1, 256	N/A	No default	AXIS_REF.AXIS_NUM is the master axis number.

4.2.8.1.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	0, 1	N/A	Boolean output to indicate successful request. • This output does not indicate the faults are cleared. • It does indicate the request was made. •

4.2.8.1.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- The condition causing the drive fault has to be corrected before calling this function.
- If the fault condition still exists when this function is called, this function sends a request to the drive but the drive faults remain.
- This function does **not** reset axis errors.
  - MC_ResetError is required to reset axis errors and possibly to re-enable or turn power on to the servo axis after the fault condition is cleared.

4.2.8.1.3.1 Usage

When there is a positive transition of the EN input, this function requests a Fault Reset of the Drive for the Axis defined in the axis input of this function.

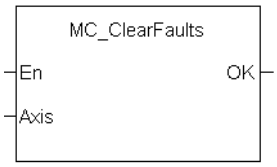
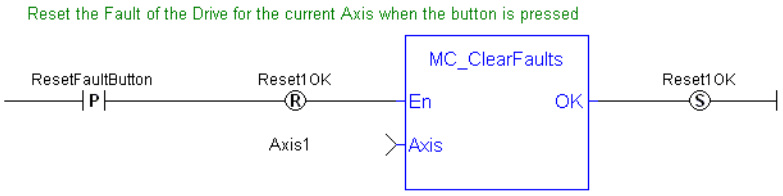


Figure 4-50: MC_ClearFaults

4.2.8.1.4 FBD Language Example

4.2.8.1.5 FFLD Language Example



4.2.8.1.6 IL Language Example

Not available.

4.2.8.1.7 ST Language Example

```
(* MC_ClearFaults ST example *)
MC_ClearFaults( Axis1);    //clear drive faults for Axis 1
```

See Also

"MC_ResetError" (→ p. 422)

4.2.8.2 MC_CreatePLCAxis

PLCopen

Function - Creates a PLCopen Axis.

4.2.8.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Requests to create a PLCopen axis.
AxisName	STRING	No range	N/A	No default	Axis name.
BusInterface	STRING	No range	N/A	No default	Bus interface identifier: • EtherCATDriver = EtherCAT interface. • MSBusDriver = KAS Simulator interface.
BusAddress	DINT	Bus dependent	N/A	No default	Address of the drive on the bus.
AxisNumber	UINT	1, 256	N/A	No default	Axis number.

<table>
<tr><th>Input</th><th>Data Type</th><th>Range</th><th>Unit</th><th>Default</th><th>Description</th></tr>
<tr><td>AxisType</td><td>USINT</td><td>0, 1</td><td>N/A</td><td>No default</td><td>
<table>
<tr><th>USINT</th><th>Type</th><th>Description</th></tr>
<tr><td>0</td><td>MC_AXIS_TYPE_SERVO_STEPPER</td><td>A Servo or Stepper axis can be mapped to a physical or simulated drive.
Either a servo or stepper drive is supported.</td></tr>
<tr><td>1</td><td>MC_AXIS_TYPE_DIGITIZING</td><td>A digital position input from a drive or other device.
It is useful as an input for gearing, camming, etc.</td></tr>
<tr><td>2</td><td>MC_AXIS_TYPE_VIRTUAL</td><td>A virtual axis cannot be mapped to a physical or simulated drive.
It is useful for generating motion trajectory as an input for gearing, camming, etc.</td></tr>
</table>
</td></tr>
<tr><td>DriveAxisNumber</td><td>UINT</td><td>1, 256</td><td>N/A</td><td>No default</td><td>This one-based number specifies the axis on the drive.
For a single-axis drive this number should be 1.</td></tr>
<tr><td>UserUnits</td><td>DINT</td><td>1, 2147483647</td><td>User units</td><td>No default</td><td>User unit portion of the user unit/feedback unit ratio.</td></tr>
</table>

Input	Data Type	Range	Unit	Default	Description
FeedbackUnits	DINT	1, 2147483647	Feedback units	No default	Feedback unit portion of the user unit/feedback unit ratio.
Rollover	LREAL	0, 4294967296	User units	No default	Rollover position. 0 (zero) = no rollover.
UpdateRate	UINT	3, 9	N/A	No default	Servo update rate. • (0, 1, and 2 are reserved for future enhancements) • 3 = 125 µsec • 4 = 250 µsec • 5 = 500 µsec • 6 = 1 msec • 7 = 2 msec • 8 = 4 msec • 9 = 8 msec

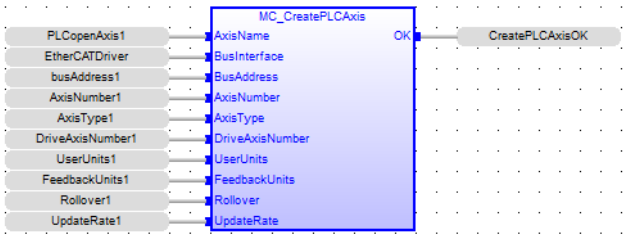
4.2.8.2.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Indicates the axis has been created.

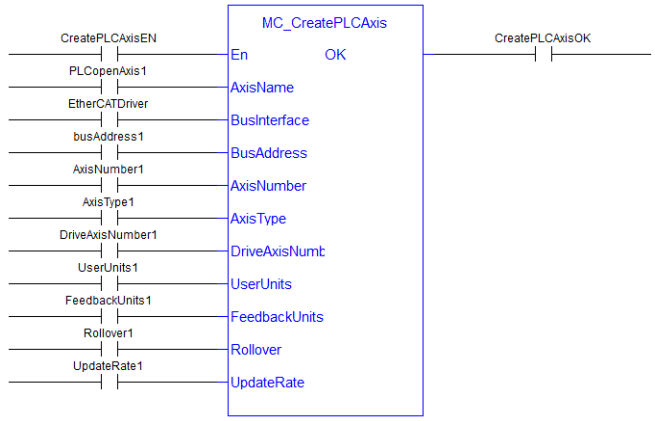
4.2.8.2.3 Remarks

- A call to this function is automatically generated when the application is compiled, based on the data entered in the PLCopen Axis Data dialog.
- **MC_CreateAxis** must be called between "MLMotionInit" (→ p. 549) and "MLMotionStart" (→ p. 552).

4.2.8.2.4 FBD Language Example



4.2.8.2.5 FFLD Language Example



4.2.8.2.6 IL Language Example

Not available.

4.2.8.2.7 ST Language Example

```
(* MC_CreatePLCAxis ST Example *)
AxisName1          := 'PLCOpenAxis1';
BusName1           := 'EtherCATDriver';
BusAddress1        := 1001;
AxisNumber1        := 1;
AxisType1          := MC_AXIS_TYPE_SERVO_STEPPER;
DriveAxisNumber1   := 1;
UserUnits1         := 360;
FeedbackUnits1     := 1048576;
Rollover1          := 0;
UpdateRate1        := 3;
MC_CreateAxis(AxisName1, BusName1, BusAddress1, AxisNumber1, AxisType1,
DriveAxisNumber1, UserUnits1, FeedbackUnits1, Rollover1, UpdateRate1);
```

4.2.8.3 MC_EStop

PLCopen ✓

Function - Causes an emergency stop (E-stop).

4.2.8.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	A positive transition of this input causes an E-stop on the specified axis.
Axis	AXIS_REF	1, 256	N/A	No default	The AXIS_NUM element of the AXIS_REF structure.

4.2.8.3.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Indicates the E-stop was executed. If an invalid Axis input was specified, this output is not energized and no E-stop is performed.

4.2.8.3.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

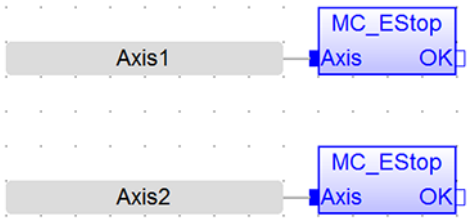
- Call **MC_EStop** to generate an emergency stop for an axis.
- Call "MC_ResetError" (→ p. 422) to reset the emergency stop.
- Actions of an E-stop:
  - Stops motion interpolation.
  - Clears all moves from the queue (active and next).
  - Changes the axis state to ErrorStop.
    - See PLCopen State Machine for more information.

- Request the drive to open the position loop and disable the drive.

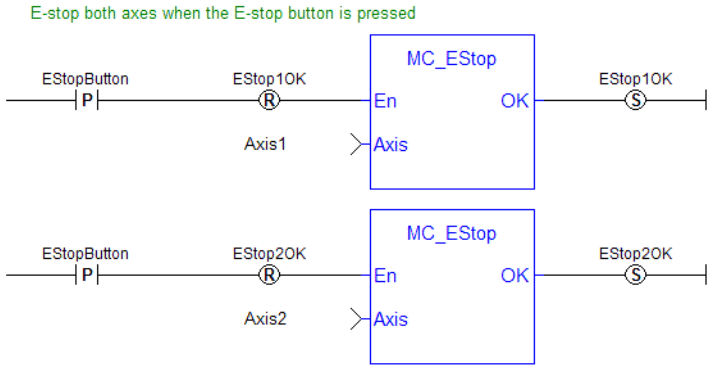
- The E-stop remains in effect until the application calls "MC_ResetError" (→ p. 422) to reset the E-stop.

Figure 4-51: MC_EStop

4.2.8.3.4 FBD Language Example



4.2.8.3.5 FFLD Language Example



4.2.8.3.6 IL Language Example

Not available.

4.2.8.3.7 ST Language Example

```
(* MC_EStop ST example *)
ON EStopButton DO
MC_EStop( Axis1 );
MC_EStop( Axis2 );
END_DO;
```

See Also

"MC_ResetError" (→ p. 422)

4.2.8.4 MC_InitAxis

PLCopen

Function - Initializes a PLCopen Axis' data.

4.2.8.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Requests to create a PLCopen axis.
AxisNumber	UINT	1, 256	N/A	No default	Axis number.
VelocityLimit	LREAL	No range	User unit/sec	No default	Reserved for future use.
LowerLimit	LREAL	No range	User units	No default	Reserved for future use.
UpperLimit	LREAL	No range	User units	No default	Reserved for future use.
LimitControl	UINT	0, 2	N/A	No default	Reserved for future use.
PosErrorLimit	LREAL	No range	User units	No default	Position error limit: When the Position Error (command position - actual position) exceeds this value, an E-stop is generated.
InPositionBand	LREAL	No range	User units	No default	In-position bandwidth: When the axis actual position is within this distance from its programmed endpoint, the axis is considered in position.

4.2.8.4.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Indicates the initialization is complete.

4.2.8.4.3 Remarks

- A call to this function is automatically generated when the application is compiled.
- This is based on the data entered in the PLCopen Axis Data dialog.

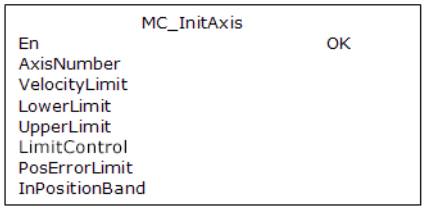
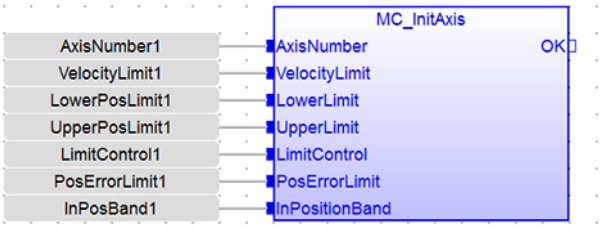
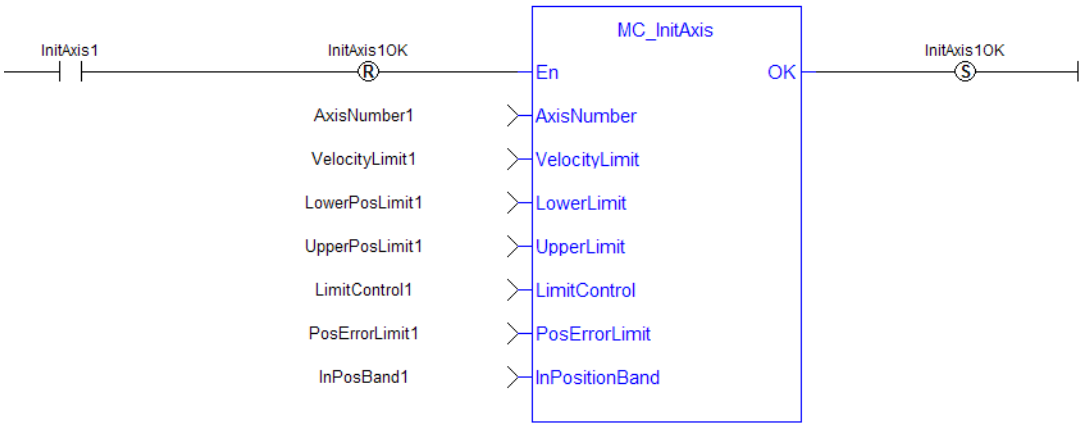


Figure 4-52: MC_InitAxis

4.2.8.4.4 FBD Language Example



4.2.8.4.5 FFLD Language Example



4.2.8.4.6 IL Language Example

Not available.

4.2.8.4.7 ST Language Example

```
(* MC_InitAxis ST example *)
AxisNumber1    := 1;
VelocityLimit1 := 10000; (*User unit/second*)
LowerPosLimit1 := 0;
UpperPosLimit1 := 0;
LimitControl1  := 0; (* Ignore lower and upper pos limit*)
PosErrorLimit1 := 10; (*User unit*)
InPosBand1     := 0;
MC_InitAxis(AxisNumber1, VelocityLimit1, LowerPosLimit1, UpperPosLimit1,
LimitControl1, PosErrorLimit1, InPosBand1);
```

4.2.8.5 MC_InitAxisFeedback

PLCopen

Function - Initializes a PLCopen digitizing Axis' position data.

4.2.8.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Request to Initialize a PLCopen Digitizing Axis. This input is only applicable to FFLD programs.
AxisNumber	UINT	1, 256	N/A	No default	Digitizing Axis number.
FeedbackPDOIndex	UINT	0, 65536	N/A	No default	The object Index through which the feedback data is sent.
FeedbackPDOSubIndex	UINT	0, 255	N/A	No default	The sub index of the object through which the feedback data is sent to the controller.
FeedbackDataResolution	UINT	1, 32	N/A	No default	The resolution of the Encoder.

4.2.8.5.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Indicates the initialization is complete. This output is only applicable to FFLD programs.
ErrorID	DINT	No range	N/A	Indicates if the call failed or succeeded. If the call succeeds, this is set to 0 (zero).

4.2.8.5.3 Remarks

- A call to this function is automatically generated when the application is compiled.
  - This call is based on the data entered in the PLCopen Axis Data dialog.

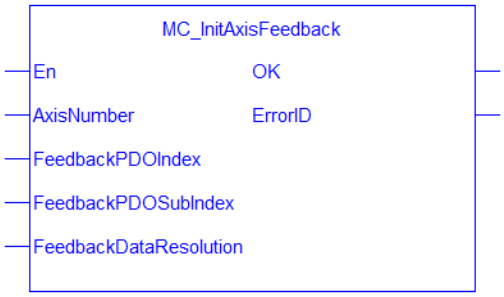
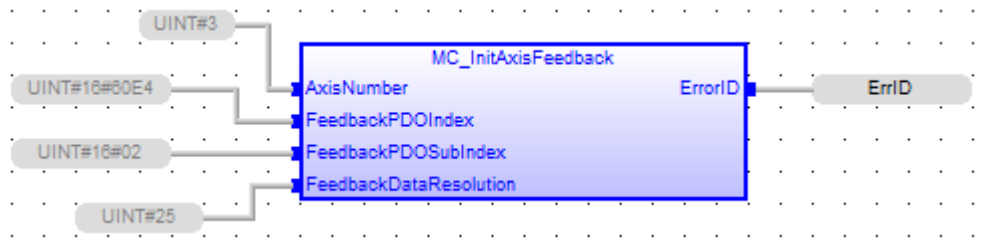


Figure 4-53: MC_InitAxisFeedback

4.2.8.5.4 FBD Language Example



4.2.8.5.5 FFLD Language Example

Not available.

4.2.8.5.6 IL Language Example

Not available.

4.2.8.5.7 ST Language Example

```
(* MC_InitAxisFeedback ST example *)
// Initialize the digitizing Axis (Axis #3) with the Feedback object 0x60E4
subIndex 2.
Encoder resolution is 25bits.
ErrID := MC_InitAxisFeedback(3, 16#60E4, 16#02, 25);
```

4.2.8.6 MC_Power

PLCopen ✓

Function Block - Requests to either enable the drive and close the loop, or disable the drive and open the loop.

4.2.8.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	When this transitions goes to high: • The control **closes** the servo loop. • Sends a command to the drive to **enable**. When this transitions go to low: • The control **opens** the servo loop. • Sends a command to the drive to **disable**.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
EnablePositive	BOOL	0, 1	N/A	No default	For future enhancement.
EnableNegative	BOOL	0, 1	N/A	No default	For future enhancement.
BufferMode	SINT	0 (zero)	N/A	No default	Unused.

4.2.8.6.2 Outputs

Output	Data Type	Range	Unit	Description
Status	BOOL			Indicates the enabled/disabled state of the drive.
Busy	BOOL			Always FALSE. For future enhancement.
Error	BOOL			Indicates an invalid input was specified.
ErrorID	INT			Indicates the error if **Error** output is TRUE.

4.2.8.6.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- The Status output indicates the state of the position loop.
  - If the position loop is open, the axis command position is set to the actual position of the axis and tracks the actual position.

NOTE

Be careful if there is more than one instance of **MC_Power** FB for the same drive, scanned in the same cycle.
The problem arises when one instance requests the drive to enable and the other requests the same drive to disable.
To avoid this trap, it is recommended to have only one instance of **MC_Power** for all active programs.

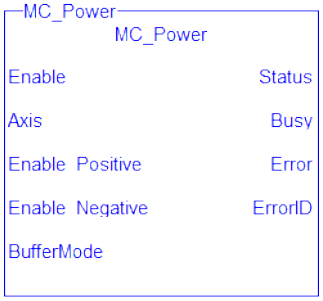
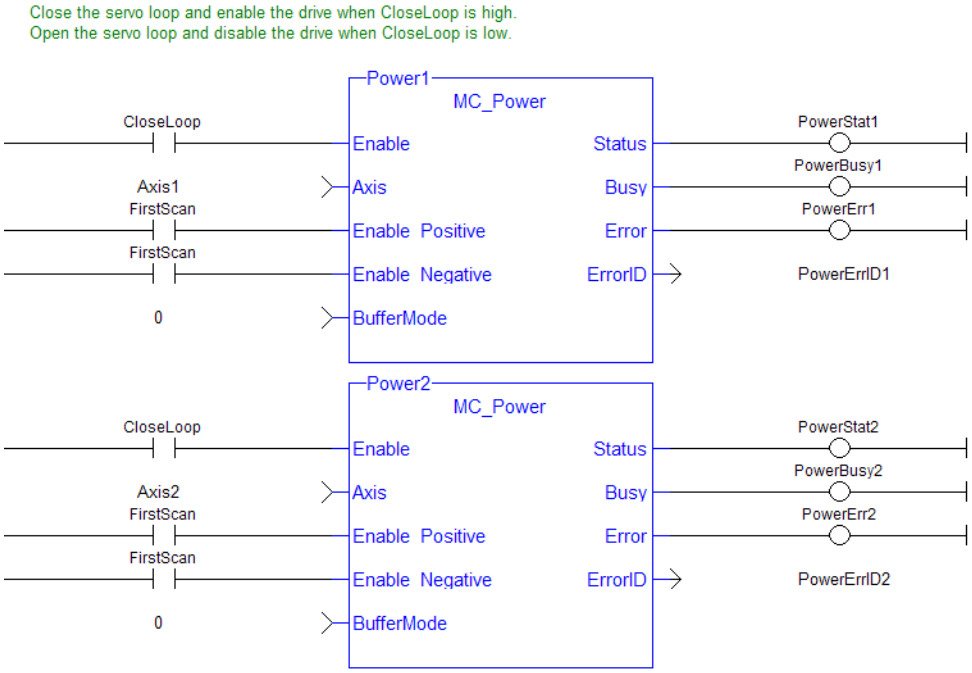


Figure 4-54: MC_Power

4.2.8.6.4 FBD Language Example

Not available.

4.2.8.6.5 FFLD Language Example



4.2.8.6.6 IL Language Example

Not available.

4.2.8.6.7 ST Language Example

```
(* MC_Power ST example *)
Inst_MC_Power( CloseLoopReq, Axis1, TRUE, TRUE, 0 );
//Inst_MC_Power is an instance of MC_Power function block
DriveIsOn := Inst_MC_Power.Status;     //store the Status output into a user
defined variable
```

4.2.8.7 MC_ResetError

PLCopen

Function - Resets the errors of the specified axis.

4.2.8.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Requests to reset the axis errors.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.

4.2.8.7.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Function block executed successfully.

4.2.8.7.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

This function performs these tasks in sequence:

1. Sends a request to the drive to clear any drive faults that exists
2. Resets the axis errors

NOTE

The condition causing the axis error has to be corrected before calling this function. The axis error still remains until the error condition exists when this function is called.

See Transition Events and Actions, #15 in the status machine of the CANopen protocol.

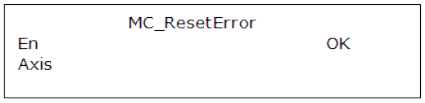
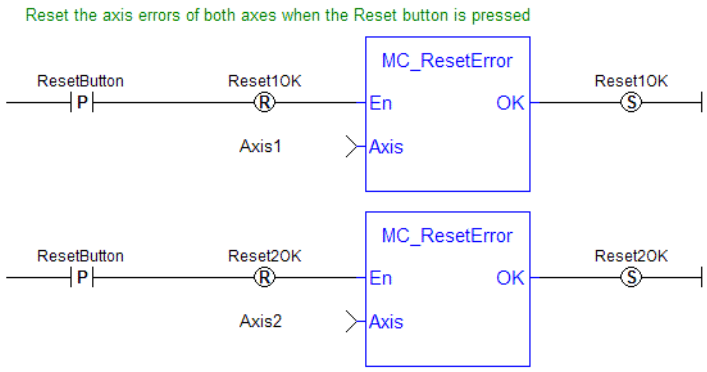


Figure 4-55: MC_ResetError

4.2.8.7.4 FBD Language Example

Not available.

4.2.8.7.5 FFLD Language Example



4.2.8.7.6 IL Language Example

Not available.

4.2.8.7.7 ST Language Example

```
//reset the axis and drive errors for Axis 1

MC_ResetError( Axis1 );
```

4.2.8.8 MC_Stop

PLCopen

Function Block - Aborts the active move, removes the next move from the queue, performs a controlled stop, and switches the axis to Stopping state.

4.2.8.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to stop the axis. It can be held high to prevent any other moves from being queued.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
Deceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Deceleration rate. S-curve: Unused.
Jerk	LREAL	No range	User unit/sec^3	No default	Trapezoidal: 0 (zero). S-curve: Constant jerk.

4.2.8.8.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			Indicates both: • The axis has reached zero velocity. • The Execute input is low.
Busy	BOOL			High from the time both: • The Execute input goes high until the axis reaches zero velocity. • The Execute input is low.
Active	BOOL			High from the time both: • The MC_Stop move becomes the active move until the axis reaches zero velocity. • The Execute input is low.
Error	BOOL			Indicates an invalid input was specified.
ErrorID	INT			Indicates the error if the **Error** output is set to TRUE.

4.2.8.8.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

-
- An MC_Stop cannot be aborted.
  - This means that, while in Stopping state, no function block can command any motion on the axis.
  - The axis remains in Stopping state until it reaches zero velocity and the Execute input is low.
- The application program can hold the axis in Stopping state even after it reaches zero velocity by leaving the Execute input high.

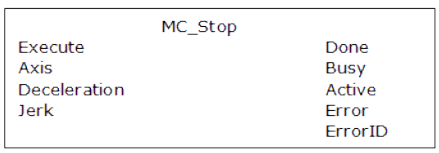


Figure 4-56: MC_Stop

4.2.8.8.3.1 Time Diagram

This example shows the behavior of the combination of a MC_Stop FB with a MC_MoveVelocity FB.

- A rotating axis is ramped down with FB2 MC_Stop
- The axis rejects motion commands as long as MC_Stop parameter “Execute” = TRUE

FB1 MC_MoveVelocity reports an error indicating the busy MC_Stop command.

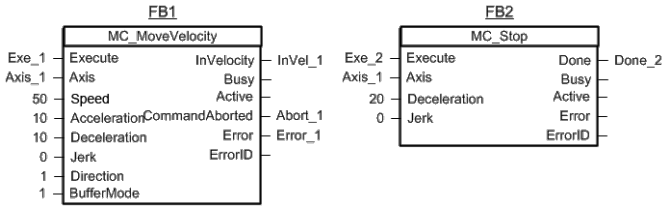


Figure 4-57: Time Diagrams: First and Second FBs

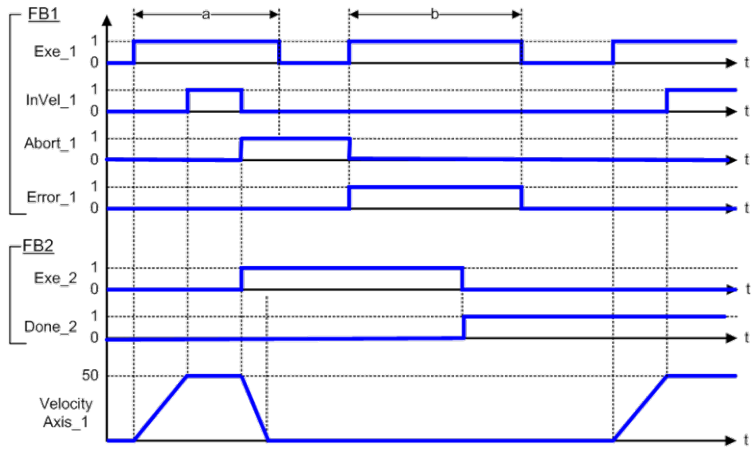
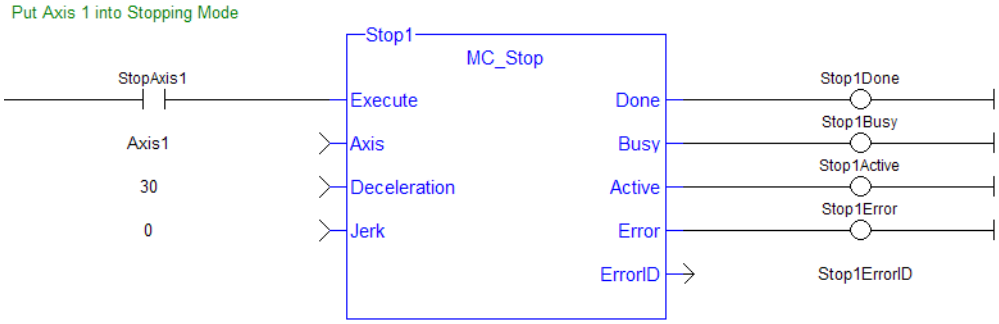


Figure 4-58: Time Diagram

4.2.8.8.4 FBD Language Example

Not available.

4.2.8.8.5 FFLD Language Example



4.2.8.8.6 IL Language Example

Not available.

4.2.8.8.7 ST Language Example

```
            (* MC_Stop S
            T example *)

Inst_MC_Stop( StopRequest , Axis1, 100.0, 100.0 ); //Inst_MC_Stop is an
instance of MC_Stop function block
```

```
StopComplete := Inst_MC_Stop.Done;      //store the Done output into a user
defined variable

StopActive := Inst_MC_Stop.Active;      //store the Active output into a user
defined variable

StopError := Inst_MC_Stop.Error;        //store the Error output into a user
defined variable
```

4.2.9 I/O

These functions provides I/O control over TouchProbe functions.

Name	Description
MC_AbortTrigger	When the Execute input transitions from low to high, this function block aborts an MC_TouchProbe function block.
MC_TouchProbe	Arms a Fast Input and returns the latched position when the Fast Input event occurs.

4.2.9.1 MC_AbortTrigger

PLCopen

Function Block - When the Execute input transitions from low to high, this function block aborts an MC_TouchProbe function block.

4.2.9.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to abort the "MC_TouchProbe" (→ p. 428) function block.
Axis	AXIS_REF	1, 256	N/A	No default	Identifies the axis specified in the **MC_TouchProbe** function block which is to be aborted.

Input	Data Type	Range	Unit	Default	Description
TriggerInput	TRIGGER_REF	INT	N/A	No default	Identifies the fast input specified in the **MC_TouchProbe** function block which is to be aborted. The structure elements are: **Direction** INT Range = 1, 2. • 1 = Rising edge. • 2 = Falling edge. **InputID** INT Range = 0, 1 • 0 = Touch Probe 1 / Capture Engine 0 • 1 = Touch Probe 2 / Capture Engine 1 **Trigid**INT Range = 0, 256. • The axis number of the input. • 0 (zero) indicates the trigger axis is to be the same as Axis.AXIS_NUM. **NOTE** **TrigMode** INT (TriggerInput.TrigMode) is not presently supported by this function.

4.2.9.1.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Function block has completed.
Busy	BOOL	No range	N/A	Indicates this function block is executing.
Error	BOOL	No range	N/A	Indicates the function block did not complete due to an error. The **ErrorID** output indicates the type of error when this output is high.
ErrorID	INT	No range	N/A	When the Error output is: • high, this output indicates the type of error. • low, this output is undefined.

4.2.9.1.3 Remarks

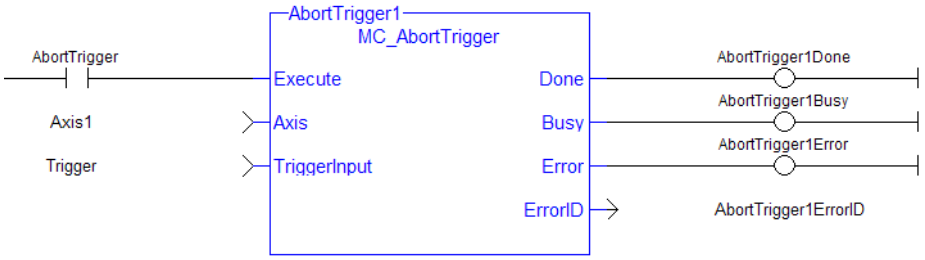
NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

4.2.9.1.4 FBD Language Example

Not available.

4.2.9.1.5 FFLD Language Example



4.2.9.1.6 IL Language Example

Not available.

4.2.9.1.7 ST Language Example

```
(* MC_AbortTrigger ST example *)
Inst_MC_AbortTrigger( AbortReq, Axis1, TriggerInputRef );
//Inst_MC_AbortTrigger is an instance of MC_AbortTrigger
```

See Also

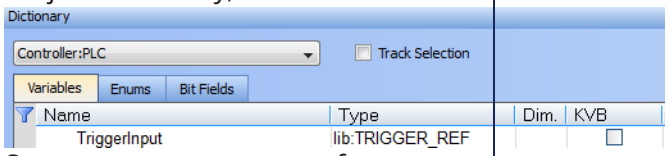
"MC_TouchProbe" (→ p. 428)

4.2.9.2 MC_TouchProbe

PLCopen

Function Block - Arms a Fast Input and returns the latched position when the Fast Input event occurs.

4.2.9.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to arm the input to return the latched position.
Axis	AXIS_REF	1, 256	N/A	N/A	Selects the axis the position is latched to.
TriggerInput	TRIGGER_REF	See Description.	N/A	No default	• Sets up the mechanism on the controller for the capture input signal. • An instance of the TRIGGER_REF function must first be setup in the Project Dictionary, as seen here:  • See "Elements" (→ p. 429) for information about Ranges.

Input	Data Type	Range	Unit	Default	Description
WindowOnly	BOOL	No range	N/A	No default	Enables a position latching window. • When this input is set, a window is defined by the FirstPosition and LastPosition inputs. • Any Fast Input event that occurs outside the window is ignored. • The first Fast Input event that occurs within the window latches the axis position.
FirstPosition	LREAL	No range	User units	No default	See the "WindowOnly" (→ p. 429) Description about how this input and the LastPosition input define the window. • This input is only applicable when the **WindowOnly** input is high. • If the **WindowOnly** input is low, this input is ignored.
LastPosition	LREAL	No range	User units	No default	See the "WindowOnly" (→ p. 429) Description about how this input and the FirstPosition input define the window. • This input is only applicable when the **WindowOnly** input is high. • If the **WindowOnly** input is low, this input is ignored.

4.2.9.2.1.1 Elements

Capture Engine

This is the drive capture engine to be used.

INT TriggerInput.InputID

InputID INT

Range = 0, 1

- 0 = Touch Probe 1 / Capture Engine 0
- 1 = Touch Probe 2 / Capture Engine 1

Trigger Direction

This is the input signal's edge to capture.
INT TriggerInput.Direction
Range = 1, 2

- 1 = Rising edge.
- 2 = Falling edge.

TIP

Trigger Direction is also sent to the servo drive and is shown in the WorkBench Position Capture screen **Edge** setup.

Axis Number

This is where the input comes from.

INT TriggerInput.TrigID
Range = 0, 256.
0 = Trigger axis is to be the same as Axis.AXIS_NUM.

Trigger Mode

This is the capture method.

INT TriggerInput.TrigMode
Range = 0, 1

- 0 = Time based capture.
- 1 = Position based capture.
- For position based capture, the TrigID must be the same as the AXIS_REF.

NOTE

The Mode (either Time or Position) must be configured the same in the servo drive.
This can be done either in:

- COE Init commands and executed when the EtherCAT network is initialized (0x3460, subindex 3 and 4).
- The WorkBench Positon Capture screen.

4.2.9.2.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			• Function block has completed. • The RecordedPosition output is valid.
Busy	BOOL			Indicates the specified input is: • Arming or is armed. • Waiting for the trigger and recording of the position to occur.
CommandAborted	BOOL			Indicates the move was aborted. A TriggerAbort function block has executed and canceled this function.
Error	BOOL			The function block has not completed successfully due to an error. The ErrorID output indicates the type of error.
ErrorID	INT			• When the Error output is high, this output indicates the type of error. • When the Error output is low, this output is undefined.
RecordedPosition	LREAL		User units	• When the Done output goes high, this output returns the latched position. • When the Done output is low, this output is undefined.

4.2.9.2.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- This function or function block does not generate any motion.
-
- This function block captures an axis position
- When the Execute input transitions from low to high, the control requests the drive to arm its Fast Input to latch the axis position when a Fast Input occurs.
- The Axis input specifies which axis's position to latch and the TriggerInput input specifies which Fast Input to use and whether to trigger on the rising or falling edge of the Fast Input.
- When the Fast Input event occurs, the drive latches the axis's position.

  - This function block then returns the latched position at the RecordedPosition output and set the Done output high.
  - This process can be canceled with the AbortTrigger function block.

- If the WindowOnly input is high, the FirstPosition input and the LastPosition input define a window in which a Fast Input is accepted.
- Any Fast Input events that occur outside the window is ignored.

IMPORTANT

There are differences in how **MC_TouchProbe** interacts with AKD and AKD2G drives.
See these topics for drive-specific information:

- "AKD Support With MC_TouchProbe" (→ p. 432)
- "AKD2G Support With MC_TouchProbe" (→ p. 433)

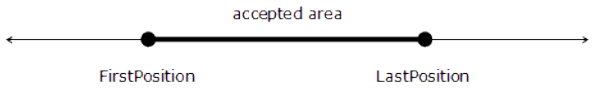
TIP

The accuracy of captured position data depends on the travel velocity. Please see the article MC_TouchProbe and Time-Based Capture on KDN for more information and how to correct for timing.

Example 1

If First Position ≤ LastPosition, the window in which a Fast Input is accepted is:

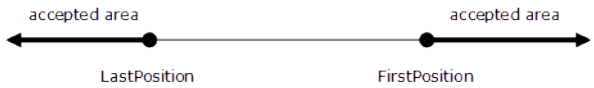
`FastInputPosition ≥ FirstPosition` AND `FastInputPosition ≤ LastPosition`



Example 2

If First Position > LastPosition, the window in which a Fast Input is accepted is:

`FastInputPosition ≥ FirstPosition` OR `FastInputPosition ≤ LastPosition`



This image shows the ladder diagram view of the MC_TouchProbe function block:

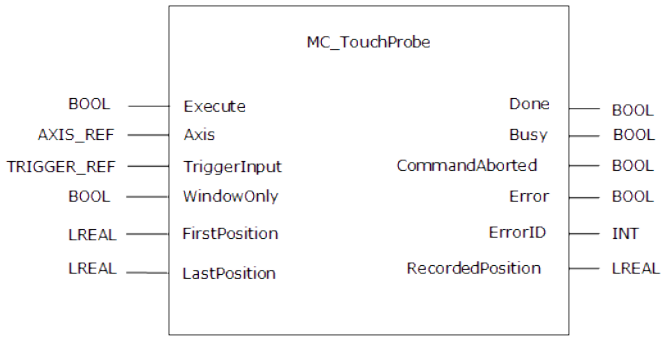


Figure 4-59: MC_TouchProbe

4.2.9.2.3.1 Fast Homing - Inputs

See these topics for more information:

- Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block
- Fast Homing Example with the PLCopen Motion Engine
- Fast Inputs with Pipe Network Motion

- Pipe Network Registration and Fast Homing
- Registration Position Capture Example with Pipe Network Trigger Block

4.2.9.2.3.2 Usage

This function block can be used to:

- Perform registration
- Determine the position of a product
- Measure product length

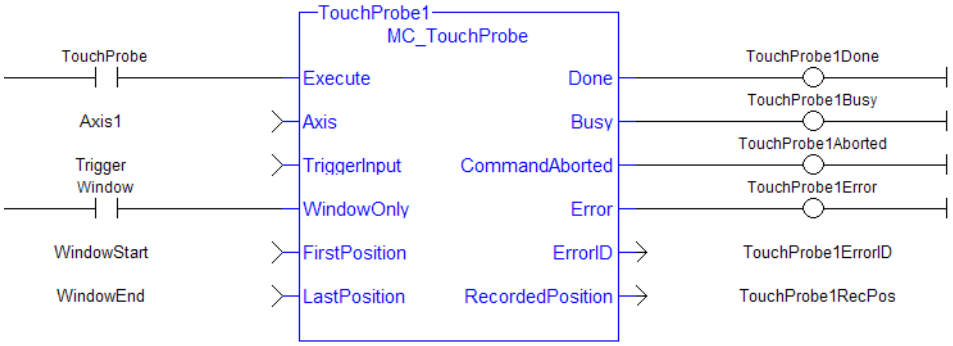
4.2.9.2.3.3 Limitations

- Both high speed inputs cannot be used at the same time.
- The TrigMode option is only used by MC_TouchProbe.

4.2.9.2.4 FBD Language Example

Not available.

4.2.9.2.5 FFLD Language Example



4.2.9.2.6 IL Language Example

Not available.

4.2.9.2.7 ST Language Example

```
(* MC_TouchProbe ST example *)
TriggerInputRef.InputID := 1;    //configure InputID
TriggerInputRef.Direction := 1;  //configure Direction
TriggerInputRef.TrigID := 0;     //configure TrigID
TriggerInputRef.TrigMode := 0;     //Capture trigger based on distributed
clock time
Inst_MC_TouchProbe( ArmProbe, Axis1, TriggerInputRef, FALSE,0.0, 0.0 );
//Inst_MC_TouchProbe is an instance of MC_TouchProbe function block
ProbeIsDone := Inst_MC_TouchProbe.Done;            //store Done output into
a user defined variable
ProbeValue := Inst_MC_TouchProbe.RecordedPosition; //store RecordedPosition
output into a user defined variable
```

See Also

"MC_AbortTrigger" (→ p. 426)

4.2.9.2.8 AKD Support With MC_TouchProbe

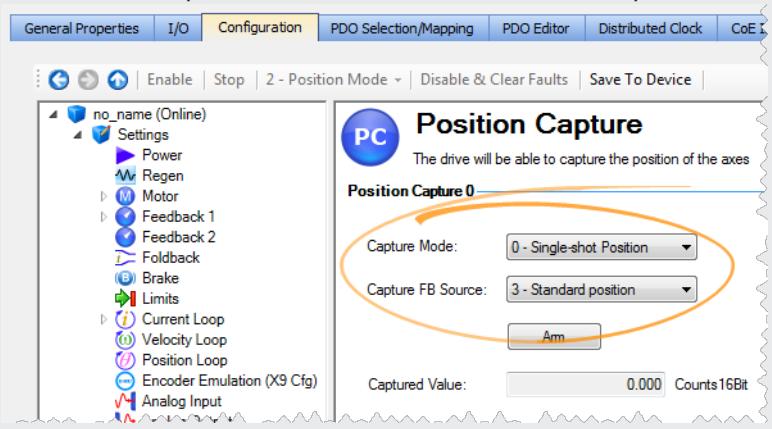
Following are several tips related to using MC_TouchProbe with AKD drives.

TIP

To use Capture Engine 1, modify the input PDOs that are used and add the Latch Position 1 parameter.

TIP

When using position-based capture, the proper Capture Mode and FB Source may need to be set up in the drive. One place to do that is in the Position Capture Screen in the KAS-IDE embedded WorkBench:



TIP

When setting up Position Capture, check the CoE-Init Command settings shown below. This is to verify they do not overwrite the corresponding drive parameters with unwanted values when the EtherCAT network initializes.

Pre Operational -> Safe Operational

Index	Subindex	Value	Comment	Direction	Source
0x6060	0	7	Opmode	Write	ESI File
0x60C2	1	1	Cycle time	Write	ESI File
0x60C2	2	-3	Cycle exp	Write	ESI File
0x3460	1	0	Latching engine 0 config to FI0, CAP0.TRIGGER=0, 0x3460-1:=0 (1 byte)	Write	ESI File
0x3460	2	1	Latching engine 1 config to FI1, CAP1.TRIGGER=1, 0x3460-2:=1 (1 byte)	Write	ESI File
0x60FE	2	196608	Digital Outputs Mask, 0x60fe:=0x30000 (4 bytes)	Write	ESI File
0x36E6	0	1	Set FBUS.PARAM02 to activate synchronization with the interrupt	Write	ESI File
0x36E8	0	1	Set FBUS.PARAM04 to disabled the drive on a motion error	Write	ESI File
0x3506	0	1	Set DRV.HWENMODE to disabled the rising edge of the hardware enable from clearing the drive faults	Write	ESI File
0x35CA	0	1048576	Set UNIT.PIN to set gear IN to the correct unit conversion.	Write	ESI File
0x365F	0	0	Set UNIT.VROTARY to set the velocity units to RPM.	Write	ESI File
0x3460	3	2	Capture mode to distributed clock time (DCT), CAP0.MODE=2, 0x3460-3:=0 (1 byte)	Write	ESI File
0x3460	4	2	Capture mode to distributed clock time (DCT), CAP1.MODE=2, 0x3460-4:=0 (1 byte)	Write	ESI File
0x50E2	0	1000	Set IL.KBUSFF (1.0 = 1000) UINT32	Write	ESI File
0x3498	0	1	Set FBUS.PROTECTION (available only since FW 01-07-03-000)	Write	ESI File

4.2.9.2.9 AKD2G Support With MC_TouchProbe

Following are several tips related to using MC_TouchProbe with AKD2G drives.

NOTE

Currently, AKD2G only supports position capture.

- AKD2G does not use CAP1 or CAP2 to provide the EtherCAT touch probes. AKD2G supports ETG6010 and DS402 for Touch Probe objects

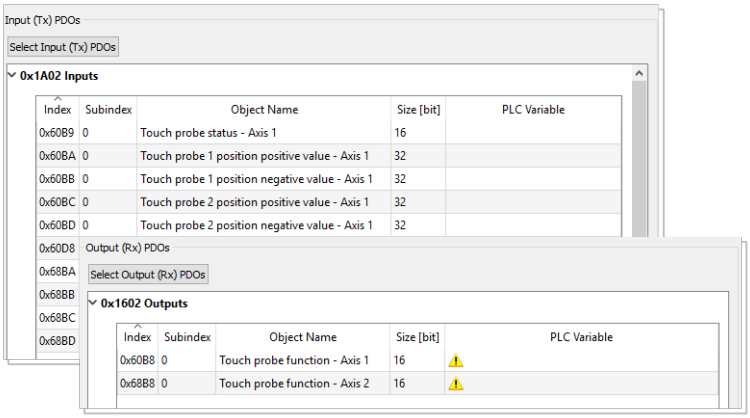
  Two touch probes per axis are supported over EtherCAT with their own dedicated hardware in the drive. Each touch probe can capture two positions, the position on the rising and the position of the falling edge of the trigger input.

  Following are the standards-compliant ETG6010 and DS402, EtherCAT / CANopen objects AKD2G supports.

Axis 1 Index	Axis 2 Index	Name	Note
60B8h	68B8h	Touch probe function / control	

Axis 1 Index	Axis 2 Index	Name	Note
60B9h	68B9h	Touch probe status	
60BAh	68BAh	Touch probe position 1 positive value	AXIS#.PL.FB, Scaling same as axis
60BBh	68BBh	Touch probe position 1 negative value	
60BCh	68BCh	Touch probe position 2 positive value	
60BDh	68BDh	Touch probe position 2 negative value	
60D0H	68D0h	Touch probe source	

- The KAS-IDE prepopulates the following PDOs with the required Touch probe objects by default.
  - Rx PDO 0x1602 with the required Touch Probe control objects
  - Tx PDO 0x1A02 with the required Touch Probe status and position value objects.



- The Trigger input source can be set by sending a SDO command.
  - Axis1:

    0x60D0 sub Index 1 for Touch Probe 1 Source

    0x60D0 sub Index 2 for Touch Probe 2 Source

  - Axis2:

    0x68D0 sub Index 1 for Touch Probe 1 Source

    0x68D0 sub Index 2 for Touch Probe 2 Source

- 6#D0h, Touch Probe Source. The following table shows how AKD2G signals are mapped to the touch probe source entry in the object dictionary. Note that a few sources appear in both the standard and the manufacture ranges to provide some consistency.

DS402 & ETG6010 Values	Text from Standard	AKD2G Values for 6#D0h	Equivalent CAP#.TRIGGER	AKD2G Note
-32768 to -1	Manufacturer specific	-41 to -42	41 to 42	Z pulse for Axis 1 to 2
		-31 to -35	31 to 35	Z pulse for Feedback 1 to 5 As FB1, 2, 4, and 5 do not support Z pulses then these will not be shown. When we support SFA on FB 1 and 2 then Z pulse may be possible. X23 is optional so if not fitted then -33 will not be valid.
		-21 to -26	21 to 26	DIO1 to DIO6 When X22 is not fitted options -21 and -22 will not be valid. When X23 is not fitted options -23 to -26 will not be valid.
		-1 to -12	1 to 12	DIN1 to DIN12 When X22 is not fitted options -9 to -12 will not be valid.
0	Reserved			Not valid
1	Digital Input 1 (Touch Probe input)	1	1	DIN1. Fast Opto
2	Digital Input 2 (Touch Probe input)	2	2	DIN2. Fast Opto
3	Digital Input 3 (Touch Probe input)			Not valid
4	Digital Input 4 (Touch Probe input)			Not valid
5	Hardware zero pulse signal of position encoder	5	41 for Axis 1 42 for Axis 2	Valid if PL.FBSOURCE is using a feedback that supports a Z pulse.
6	Software zero pulse encoder			Not valid
7 to 32767	Reserved			Not valid

4.2.10 Information

These functions provide feedback and are used to write parameters.

Name	Description
MC_ReadActPos	Reads the actual position of the axis.
MC_ReadActVel	Reads the actual velocity of the axis.
MC_ReadAxisErr	Returns the error status of the specified axis.
MC_ReadBoolPar	Returns the value of the specified Boolean axis parameter.
MC_ReadParam	Returns the value of the specified axis parameter.
MC_ReadStatus	Returns the state of the specified axis.
MC_WriteBoolPar	Writes the specified axis Boolean parameter.
MC_WriteParam	Writes the specified axis parameter.

4.2.10.1 MC_ReadActPos

PLCopen

Function Block - Reads the actual position of the axis.

4.2.10.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	Request to read the axis's actual position. Keeps continuously reading the actual position every PLC cycle, as long as the Enable remains high.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.

4.2.10.1.2 Outputs

Output	Data Type	Range	Unit	Description
Valid	BOOL			Indicates the value at the Position output is available.
Busy	BOOL			Indicates this function block is executing.
Error	BOOL			Indicates an invalid input was specified.
ErrorID	INT			Indicates the error if **Error** output is TRUE.
Position	LREAL		User units	Actual position of the axis.

4.2.10.1.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

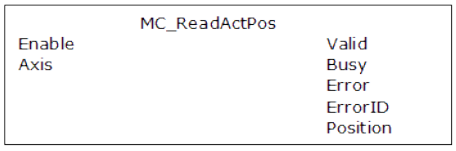
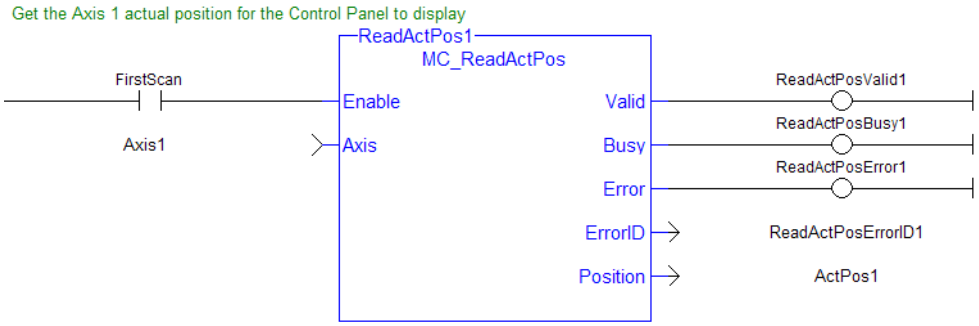


Figure 4-60: MC_ReadActPos

4.2.10.1.4 FBD Language Example

Not available.

4.2.10.1.5 FFLD Language Example



4.2.10.1.6 IL Language Example

Not available.

4.2.10.1.7 ST Language Example

```
            (* MC_ReadActPos S
            T example *)
Inst_MC_ReadActPos( TRUE, Axis1 );
//Inst_MC_ReadActPos is an instance of MC_ReadActPos function block

ActualPos := Inst_MC_ReadActPos.Position;
//store Position output into a user defined variable
```

4.2.10.2 MC_ReadActVel

PLCopen

Function Block - Reads the actual velocity of the axis.

4.2.10.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	Request to read the axis's actual velocity.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.

4.2.10.2.2 Outputs

Output	Data Type	Range	Unit	Description
Valid	BOOL			Indicates the value at the ActualVelocity output is available.
Busy	BOOL			Indicates this function block is executing.
Error	BOOL			Indicates an invalid input was specified.
ErrorID	INT			Indicates the error if **Error** output is TRUE.
ActualVelocity	LREAL		User unit/sec	Actual velocity of the axis. Oscillations may be seen because this is an instant velocity, not an average velocity.

4.2.10.2.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

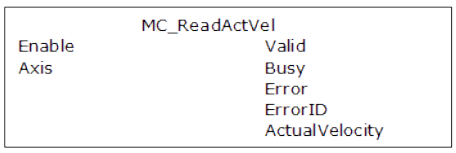
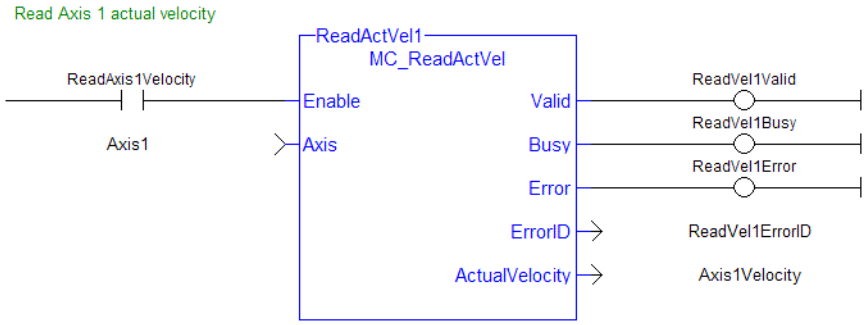


Figure 4-61: MC_ReadActVel

4.2.10.2.4 FBD Language Example

Not available.

4.2.10.2.5 FFLD Language Example



4.2.10.2.6 IL Language Example

Not available.

4.2.10.2.7 ST Language Example

```
            * MC_ReadActVel S
            T example *);
Inst_MC_ReadActVel( TRUE, Axis1 ); //Inst_MC_ReadActVel is an instance of MC_
ReadActVel function block

ActualVel := Inst_MC_ReadActVel.ActualVelocity; // store ActualVelocity
output into a user defined variable
```

4.2.10.3 MC_ReadAxisErr

PLCopen

Function Block - Returns the error status of the specified axis.

4.2.10.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	Request to read the error status of the axis.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.

4.2.10.3.2 Outputs

Output	Data Type	Range	Unit	Description
Valid	BOOL			Indicates the AxisErrorID output is valid.
Busy	BOOL			Indicates this function block is executing.
Error	BOOL			Indicates an invalid input was specified.
ErrorID	INT			Indicates the error if **Error** output is TRUE.
AxisErrorID	INT			Indicates the error status of the axis. • Each bit indicates a specific error. • Both emergency-stop (E-stop) and controlled-stop (C-stop) errors are indicated. • The "Output Bits" (→ p. 439) table defines the bits of this output.

4.2.10.3.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

4.2.10.3.3.1 Output Bits

Hexadecimal	Decimal	Description
0000H	0	No Error
0001H	1	User-set E-stop via MC_EStop, E-stop
0002H	2	Loss of Feedback, E-stop
0004H	4	Drive Fault, E-stop
0008H	8	Drive Communication Failure, E-stop
0400H	1024	Synchronization Error, C-stop
0700H	7192	Drive Overtravel Limit Exceeded, Cstop. See Overtravel Conditions.

NOTE

Multiple errors can be active at the same time.
Example: If a User-set E-stop and an Excess Position Error E-stop are both active, the value is 00000011H (17 decimal).

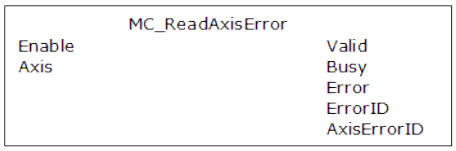
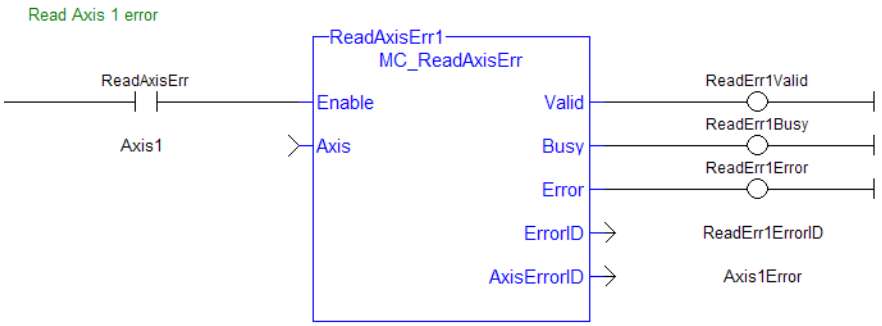


Figure 4-62: MC_ReadAxisErr

4.2.10.3.4 FBD Language Example

Not available.

4.2.10.3.5 FFLD Language Example



4.2.10.3.6 IL Language Example

Not available.

4.2.10.3.7 ST Language Example

```
            (* MC_ReadAxisErr ST example *)

Inst_MC_ReadAxisErr( TRUE, Axis1 );
//Inst_MC_ReadAxisErr is an instance of MC_ReadAxisErr function block
```

```
AxisErrorBits := Inst_MC_ReadAxisErr.AxisErrorID; //AxisErrorID contains the
error bits
```

4.2.10.4 MC_ReadBoolPar

PLCopen

Function Block - Returns the value of the specified Boolean axis parameter.

4.2.10.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	Requests to read a Boolean axis parameter.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
ParameterNumber	INT	No range	N/A	No default	Parameter number. See Axis Parameters for more information.

4.2.10.4.2 Outputs

Output	Data Type	Range	Unit	Description
Valid	BOOL			Indicates the Value output is valid.
Busy	BOOL			Indicates this function block is executing.
Error	BOOL			Indicates an invalid input was specified.
ErrorID	INT			Indicates the error if **Error** output is TRUE.
Value	BOOL			State of the Boolean parameter.

4.2.10.4.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

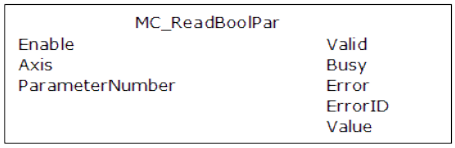
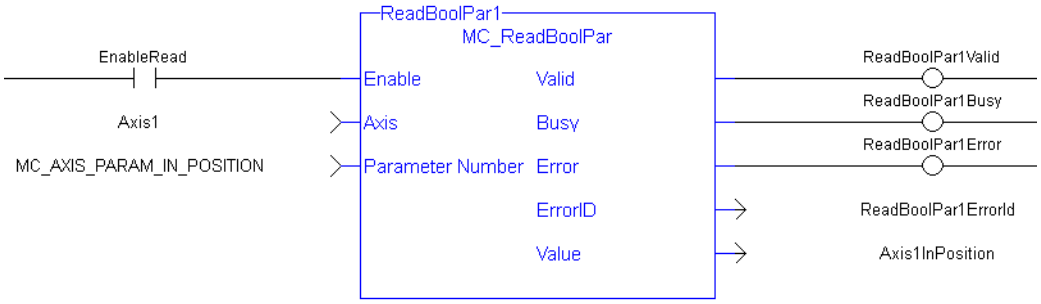


Figure 4-63: MC_ReadBoolPar

4.2.10.4.4 FBD Language Example

Not available.

4.2.10.4.5 FFLD Language Example



4.2.10.4.6 IL Language Example

Not available.

4.2.10.4.7 ST Language Example

```
            (* MC_ReadBoolPar ST example *)
Inst_MC_ReadBoolPar( EnableRead, Axis1, MC_AXIS_PARAM_IN_POSITION );
Axis1InPosition := Inst_MC_ReadBoolPar.Value;
```

4.2.10.5 MC_ReadParam

PLCopen

Function Block - Returns the value of the specified axis parameter.

4.2.10.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	Request to read the axis parameter.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
ParameterNumber	INT	No range	N/A	No default	Parameter number. See Axis Parameters for more information.

4.2.10.5.2 Outputs

Output	Data Type	Range	Unit	Description
Valid	BOOL			Indicates the Value output is valid.
Busy	BOOL			Indicates this function block is executing.
Error	BOOL			Indicates an invalid input was specified.
ErrorID	INT			Indicates the error if **Error** output is TRUE.
Value	LREAL			Value of the parameter.

4.2.10.5.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

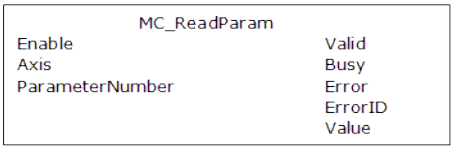
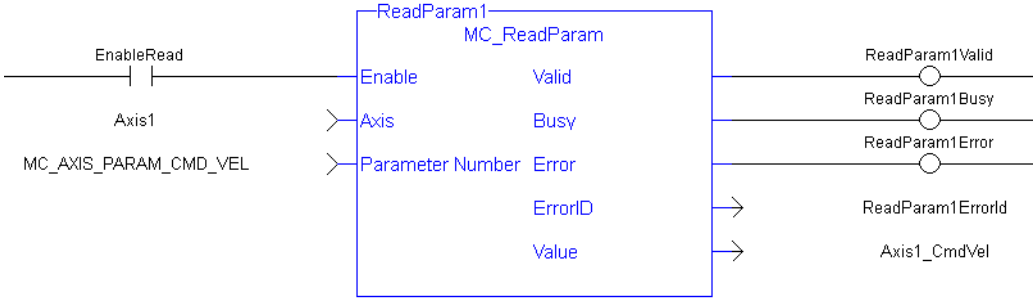


Figure 4-64: MC_ReadParam

4.2.10.5.4 FBD Language Example

Not available.

4.2.10.5.5 FFLD Language Example



4.2.10.5.6 IL Language Example

Not available.

4.2.10.5.7 ST Language Example

```
            (* MC_ReadParam ST example *)
Inst_MC_ReadParam( EnableRead, Axis1, MC_AXIS_PARAM_CMD_VEL );
Axis1_CmdVel := Inst_MC_ReadParam.Value;
```

4.2.10.6 MC_ReadStatus

PLCopen

Function Block - Returns the state of the specified axis.

4.2.10.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	Request to read and return the axis status.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.

4.2.10.6.2 Outputs

Output	Data Type	Range	Unit	Description
Valid	BOOL			Indicates the outputs are valid.
Busy	BOOL			Indicates this function block is executing.
Error	BOOL			Indicates an invalid input was specified.
ErrorID	INT			Indicates the error if **Error** output is TRUE.

Output	Data Type	Range	Unit	Description
ErrorStop	BOOL			Indicates Error Stop state. Either E-stop or C-stop.
Disabled	BOOL			Indicates Disabled state. Open loop and drive is disabled.
Stopping	BOOL			Indicates Stopping state. MC_Stop command.
StandStill	BOOL			Indicates Stand Still state. No move, closed loop, drive enabled.
DiscreteMotion	BOOL			Indicates Discrete Motion state. Pprogrammed endpoint move is active.
ContinuousMotion	BOOL			Indicates Continuous Motion state. Unending, single-axis move is active.
SynchronizedMotion	BOOL			Indicates Synchronized Motion state Slave move is active.
Homing	BOOL			Indicates Homing state. A homing cycle is currently executing.
ConstantVelocity	BOOL			Indicates the axis is moving at a constant velocity.
Accelerating	BOOL			Indicates the axis is accelerating.
Decelerating	BOOL			Indicates the axis is decelerating.

4.2.10.6.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

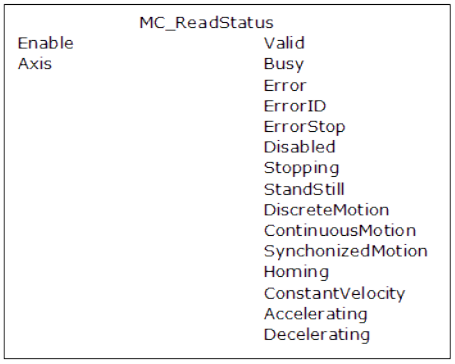
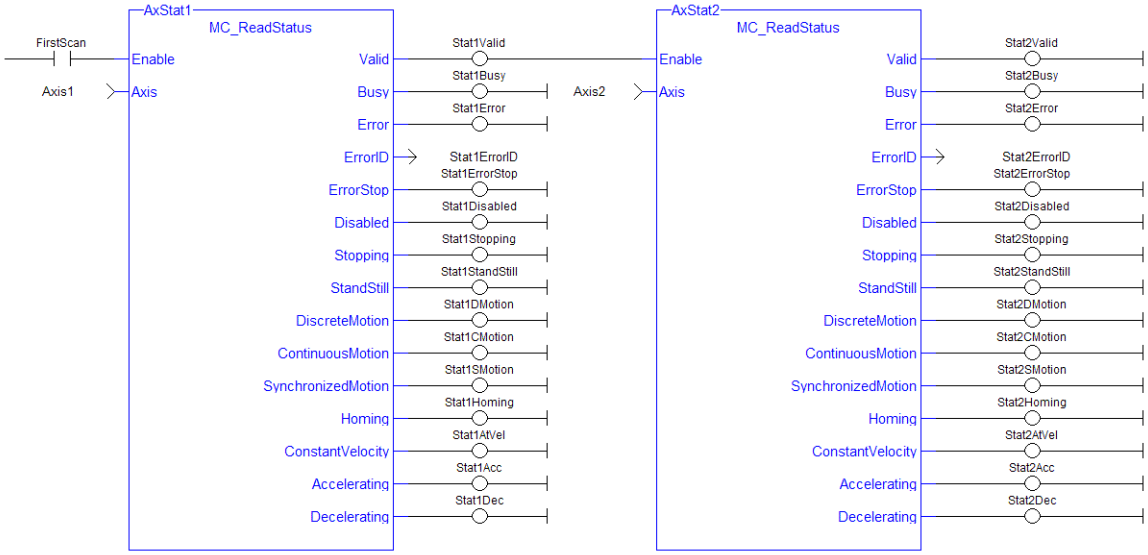


Figure 4-65: MC_ReadStatus

4.2.10.6.4 FBD Language Example

Not available.

4.2.10.6.5 FFLD Language Example



4.2.10.6.6 IL Language Example

Not available.

4.2.10.6.7 ST Language Example

```
            (* MC_ReadStatus ST example *)

Inst_MC_ReadStatus( EnableRead, Axis1 );
//Inst_MC_ReadStatus is an instance of MC_ReadStatus function block
AxisStopping := Inst_MC_ReadStatus.Stopping; // store Stopping output to a
user defined variable
AxisAccelerating := Inst_MC_ReadStatus.Accelerating; // store Accelerating
output to a user defined variable
```

4.2.10.7 MC_WriteBoolPar

PLCopen

Function Block - Writes the specified axis Boolean parameter.

4.2.10.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	Requests to write a Boolean axis parameter.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
ParameterNumber	INT	No range	N/A	No default	Parameter number. See Axis Parameters for more information.
Value	BOOL	0, 1	N/A	No default	State to write.

4.2.10.7.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			Indicates the Boolean parameter has been written.
Busy	BOOL			Indicates this function block is executing.
Error	BOOL			Indicates an invalid input was specified.
ErrorID	INT			Indicates the error if the **Error** output is set to TRUE.

4.2.10.7.3 Remarks

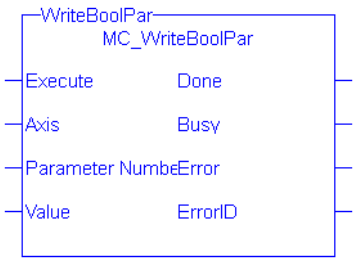
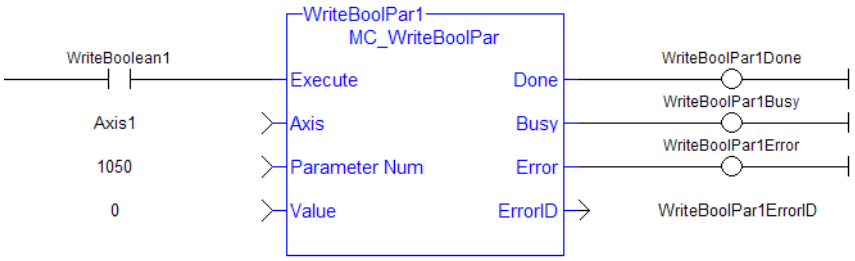


Figure 4-66: MC_WriteBoolPar

4.2.10.7.4 FBD Language Example

Not available.

4.2.10.7.5 FFLD Language Example



NOTE

Currently, MC_WriteBoolPar does not support any parameters (1050 is an arbitrary number chosen for example).

4.2.10.7.6 IL Language Example

Not available.

4.2.10.7.7 ST Language Example

```
            (* MC_WriteBoolPar ST example *)

WriteBool := FALSE;
Inst_MC_WriteBoolPar( WriteReq, Axis1, 1050, WriteBool );
```

4.2.10.8 MC_WriteParam

PLCopen

Function Block - Writes the specified axis parameter.

4.2.10.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	Requests to write an axis parameter.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
ParameterNumber	INT	No range	N/A	No default	Parameter number. See Axis Parameters for more information.
Value	LREAL	No range	N/A	No default	Value to write.

4.2.10.8.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			Indicates the parameter has been written.
Busy	BOOL			Indicates this function block is executing.
Error	BOOL			Indicates an invalid input was specified.
ErrorID	INT			Indicates the error if the **Error** output is set to TRUE.

4.2.10.8.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

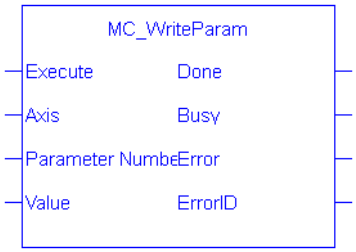
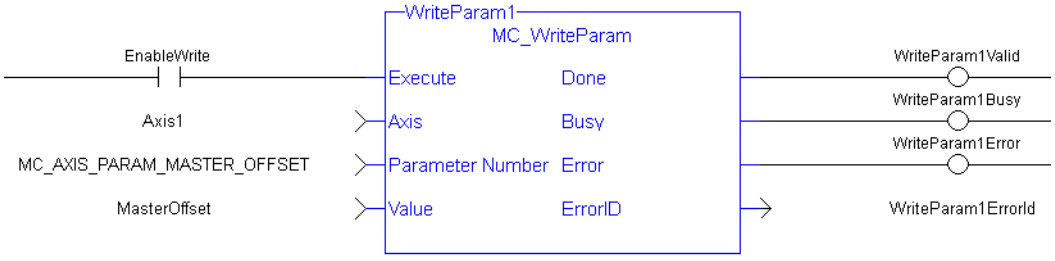


Figure 4-67: The MC_WriteParam function block

4.2.10.8.4 FBD Language Example

Not available.

4.2.10.8.5 FFLD Language Example



4.2.10.8.6 IL Language Example

Not available.

4.2.10.8.7 ST Language Example

```
            (* MC_WriteParam ST example *)

MasterOffset := 12.34;
Inst_MC_WriteParam( EnableWrite, Axis1, MC_AXIS_PARAM_MASTER_OFFSET,
MasterOffset);
```

4.2.11 PLCopenMotion

These functions provide control over an axis.

Name	Description
MC_AddSuperAxis	Adds a Superimposed Axis to the Axis's list of assigned superimposed axes.
MC_Halt	Decelerates an axis to zero velocity.
MC_MoveAbsolute	Performs a single-axis move to a specified endpoint position.
MC_MoveAdditive	Performs a single-axis move for a specified distance from the endpoint of the previous move.
MC_MoveContVel	Performs a single-axis non-ending move at a specified velocity with the option of continually updating the ongoing motion with the current input parameters.
MC_MoveRelative	Performs a single-axis move for a specified distance to perform incremental motion.
MC_MoveSuperimp	Performs a single-axis move superimposed upon the currently executing move.
MC_MoveVelocity	Performs a single-axis, non-ending move at a specified velocity.
MC_RemSuperAxis	Remove an axis from the axis's list of assigned, superimposed axes.
MC_SetOverride	Writes velocity and acceleration override factors.

4.2.11.1 MC_AddSuperAxis

Function Block - Adds a Superimposed Axis to the Axis's list of assigned superimposed axes.

4.2.11.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	N/A	N/A	N/A	Enables execution.
Axis	AXIS_REF	AXIS_NUM 1, 256	N/A	N/A	Axis to receive the additional superimposed axis's command delta.
SuperimposedAxis	UINT	1, 256	N/A	N/A	Axis number of the superimposed axis whose command delta is added to delta of `Axis`.

4.2.11.1.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Execution successful.

4.2.11.1.3 Remarks

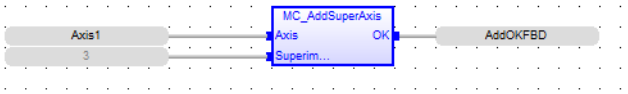
NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

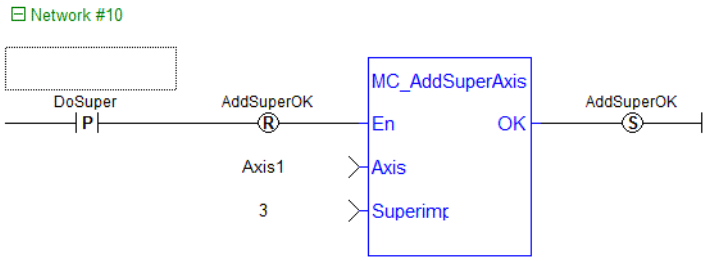
This feature allows the application program to superimpose the moves of multiple axes ("Superimposed Axes") on top of the move of another axis ("Receiving Axis").

- This is performed internally by adding the command deltas of the Superimposed Axes to the command delta of the Receiving Axis.
- A maximum of four different Superimposed Axes can be superimposed upon a Receiving Axis.
- While the Superimposed Axis is on this list, its command deltas are added to the Axis's command deltas.
- The **Axis** and the **SuperimposedAxis** must have the same update rate.
- The **OK** output goes high to indicate the function executed successfully.
- If the **OK** output **does not** go high, one of these errors is detected:
  - Axis and SuperimposedAxis do not have the same update rate.
  - Four different superimposed axes have already been assigned to Axis.
  - The axis is not a valid axis.
  - The axis is not a servo or virtual axis.
  - SuperimposedAxis is not:
    - A valid axis number.
    - A servo or virtual axis.
  - Axis could not acquire PLC motion engine lock.

4.2.11.1.4 FBD Language Example



4.2.11.1.5 FFLD Language Example



4.2.11.1.6 IL Language Example

Not available.

4.2.11.1.7 ST Language Example

```
AddOKST := MC_AddSuperAxis( Axis1, 3);
```

See Also

"MC_RemSuperAxis" (→ p. 473)

4.2.11.2 MC_Halt

Function Block - Decelerates an axis to zero velocity.

4.2.11.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to queue the move.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
Deceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Deceleration rate. S-curve: Unused.
Jerk	LREAL	No range	User unit/sec^3	No default	Trapezoidal: 0 (zero). S-curve: Constant jerk.
BufferMode	SINT	0, 5	N/A	No default	• 0 = Abort. • 1 = Buffer. • 2 = Blend to active. • 3 = Blend to next. • 4 = Blend to low velocity. • 5 = Blend to high velocity.

4.2.11.2.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates the move completed successfully. The Command Position has reached the endpoint.
Busy	BOOL	No range	N/A	High from the moment the Execute input goes high until the time the move is ended.
Active	BOOL	No range	N/A	Indicates this move is the Active move.
CommandAborted	BOOL	No range	N/A	Indicates the move was aborted.
Error	BOOL	No range	N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
ErrorID	INT	No range	N/A	Indicates the error if the **Error** output is set to TRUE.

4.2.11.2.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- It is a queued single-axis move.
  - The move is complete when the axis reaches zero velocity.
- It is typically used with Abort at the BufferMode input to terminate a move.
- See "MC_Stop" (→ p. 423) to execute a stop that cannot be aborted.

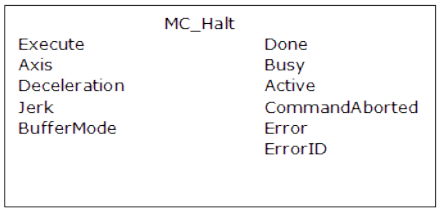


Figure 4-68: MC_Halt

4.2.11.2.3.1 Time Diagram

This example shows the behavior in combination with a "MC_MoveVelocity" (→ p. 470).

- A rotating axis is ramped down with FB2 MC_Halt.
- Another motion command overrides the MC_Halt command.
  - MC_Halt allows this, in contrast to MC_Stop.
- The axis can accelerate again without reaching standstill.

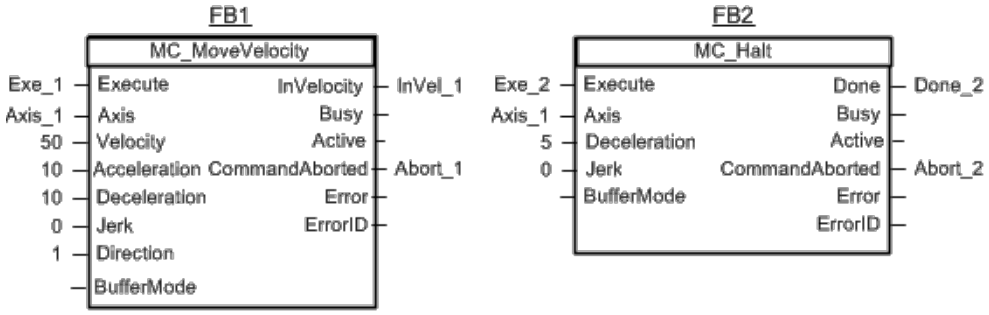


Figure 4-69: Time Diagrams: First and Second FBs

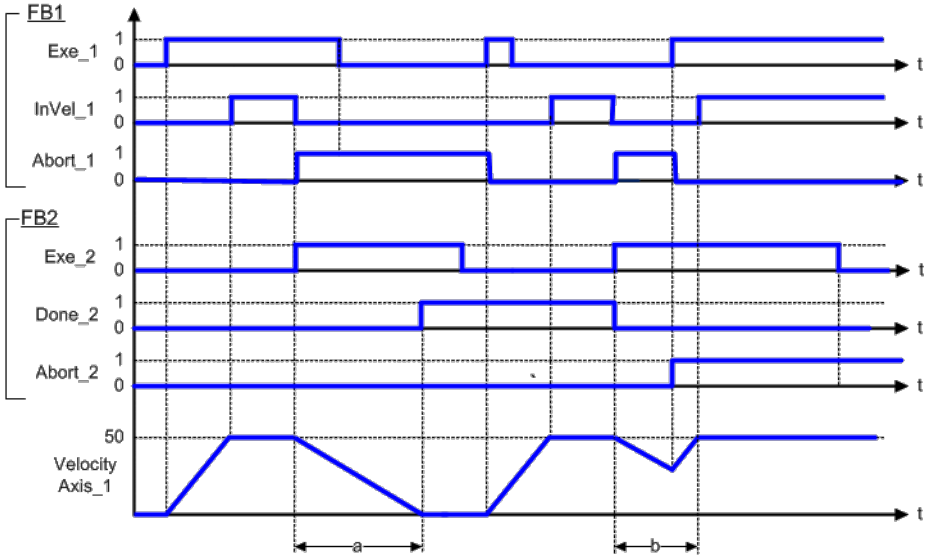
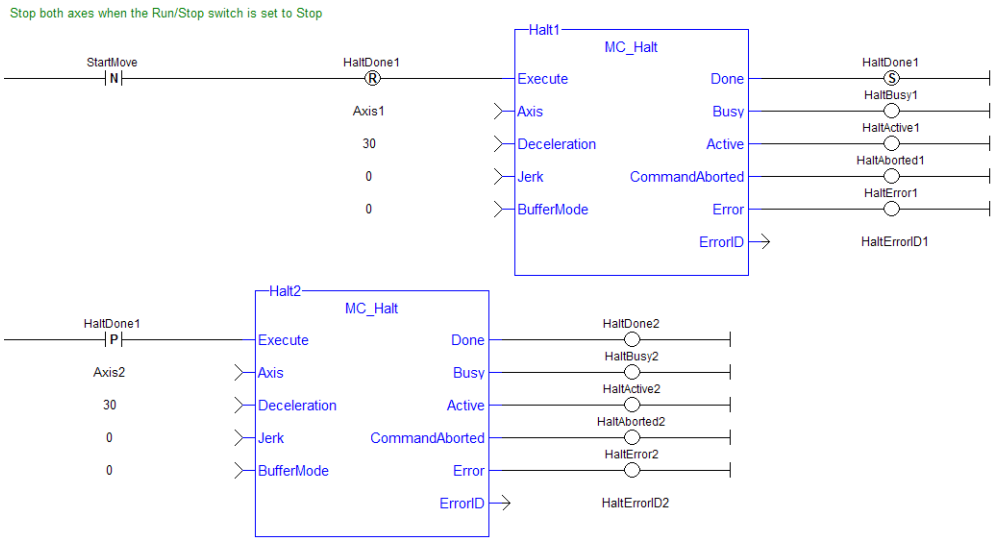


Figure 4-70: Time Diagram

4.2.11.2.4 FBD Language Example

Not available.

4.2.11.2.5 FFLD Language Example



4.2.11.2.6 IL Language Example

Not available.

4.2.11.2.7 ST Language Example

```
(* MC_Halt ST example *)
Inst_MC_Halt( HaltReq, Axis1,100.0, 100.0, 0 );
//Inst_MC_Halt is an instance of MC_halt function block
HaltComplete := Inst_MC_Halt.Done;              //store Done output into user
defined variable
```

See Also

- Call a Function Block
- Differences Between Functions and Function Blocks

4.2.11.3 MC_MoveAbsolute

PLCopen

Function Block - Performs a single-axis move to a specified endpoint position.

4.2.11.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to queue the move.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.

Input	Data Type	Range	Unit	Default	Description
Position	LREAL	See Description.	User units	No default	Endpoint position. • If Rollover Position is nonzero, this value must be in the range 0 <= Position < Rollover Position. • When not in Rollover mode, the input accepts a 64-bit floating point value. • When converted to feedback units, the range is [-2^{51}, 2^{51}-1].
Velocity	LREAL	No range	User unit/sec	No default	Velocity setpoint.
Acceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Acceleration rate. S-curve: Maximum acceleration. • If Acceleration is not valid, the ErrorID is set to 21.
Deceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Deceleration rate. S-curve: Unused.
Jerk	LREAL	No range	User unit/sec^3	No default	Trapezoidal: 0 (zero). S-curve: Constant jerk. • If Jerk is not valid, the ErrorID is set to 21.

<table>
<tr><th>Input</th><th>Data Type</th><th>Range</th><th>Unit</th><th>Default</th><th>Description</th></tr>
<tr><td>Direction</td><td>SINT</td><td>0, 4</td><td>N/A</td><td>No default</td><td>When Rollover Position is zero, a value of 0 must be specified.
When Rollover Position is nonzero, a value of 1, 2, 3, or 4 must be specified.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>No direction specification.</td></tr>
<tr><td>1</td><td>Positive direction.
The axis travels in the positive direction to the endpoint.</td></tr>
<tr><td>2</td><td>Shortest distance.
The axis travels in the direction that provides the shortest distance to the endpoint.</td></tr>
<tr><td>3</td><td>Negative direction.
The axis travels in the negative direction to the endpoint.</td></tr>
<tr><td>4</td><td>Last direction.
The axis travels to the endpoint in the same direction as its previous move.</td></tr>
</table>
If the Position input is the same as the axis's current position, then:
If Direction = 2 (shortest distance):
The axis does not move.
The Done output goes high indicating that</td></tr>
</table>

Input	Data Type	Range	Unit	Default	Description

					the move has been completed.
					• If Direction = **1, 3, or 4**:
					  • The axis travels in the specified direction, through one rollover cycle.
					  • The axis arrives back at the same position.

BufferMode	SINT	0, 5	N/A	No default	• 0 = Abort. • 1 = Buffer. • 2 = Blend to active. • 3 = Blend to next. • 4 = Blend to low velocity. • 5 = Blend to high velocity.

4.2.11.3.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates the move completed successfully. The Command Position has reached the endpoint.
Busy	BOOL	No range	N/A	High from the moment the Execute input goes high until the time the move is ended.
Active	BOOL	No range	N/A	Indicates this move is the Active move.
CommandAborted	BOOL	No range	N/A	Indicates the move was aborted.
Error	BOOL	No range	N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE.

4.2.11.3.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

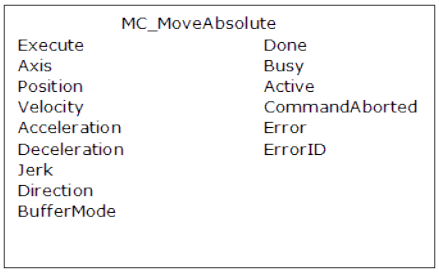


Figure 4-71: MC_MoveAbsolute

4.2.11.3.3.1 Time Diagram

These images show two examples of the combination of two absolute move Function Blocks (FBs):

- The left part of the diagram illustrates when the second FB is called **after** the first one.
  - If the first reaches the commanded position of 6000, and the velocity is 0 (zero), the output **Done** causes the second FB to move to the position 10000.
- The right part of the diagram illustrates when the second move FB starts the execution **while** the first FB is still executing.
  - The first motion is interrupted and aborted by the Test signal during the constant velocity of the first FB.
  - The second FB moves directly to the position 10000 although the position of 6000 is not yet reached.

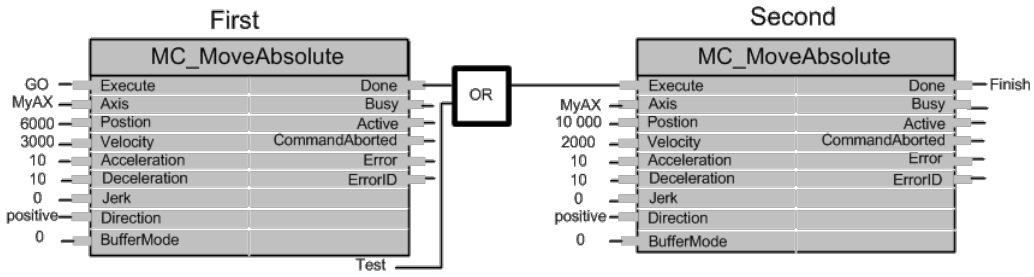


Figure 4-72: Time Diagrams: First and Second FBs

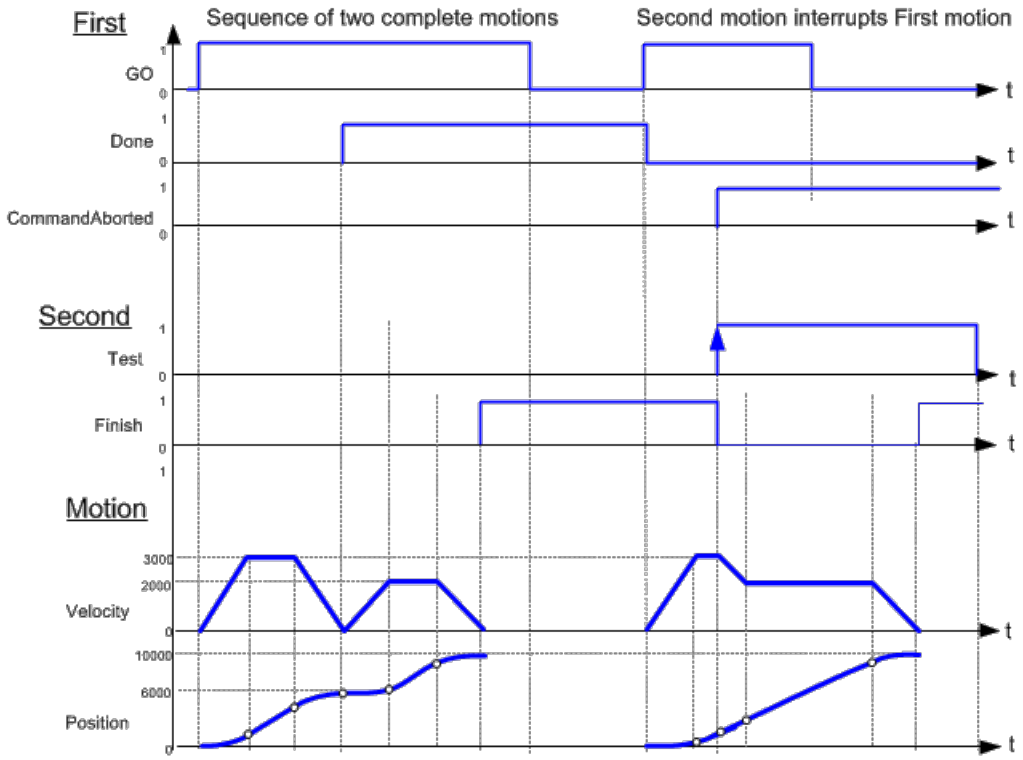
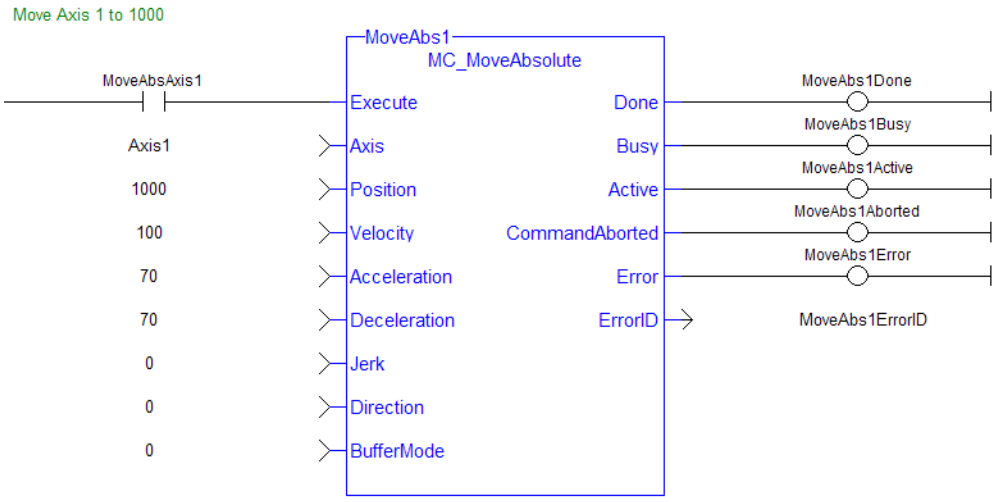


Figure 4-73: Time Diagram

4.2.11.3.4 FBD Language Example

Not available.

4.2.11.3.5 FFLD Language Example



4.2.11.3.6 IL Language Example

Not available.

4.2.11.3.7 ST Language Example

```
            (* MC_MoveAbsolute S
T example *)
Inst_MC_MoveAbsolute( MovAbsReq, Axis1, 1234.567, 100.0, 100.0, 100.0, 0, 0,
0 );  //instance of MC_MoveAbsolute
MovAbsDone := Inst_MC_MoveAbsolute.Done;  //store done output into user
defined variable
MovAbsBusy := Inst_MC_MoveAbsolute.Busy;
MovAbsActive := Inst_MC_MoveAbsolute.Active;
MovAbsAborted := Inst_MC_MoveAbsolute.CommandAborted;
MovAbsError := Inst_MC_MoveAbsolute.Error;
MovAbsErrID := Inst_MC_MoveAbsolute.ErrorID;
```

See Also

- Call a Function Block
- Differences between Functions and Function Blocks

4.2.11.4 MC_MoveAdditive

PLCopen

Function Block - Performs a single-axis move for a specified distance from the endpoint of the previous move.

4.2.11.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to queue the move.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
Distance	REAL	No range	User units	No default	Distance to add to the endpoint of the previous move.

Input	Data Type	Range	Unit	Default	Description
Velocity	LREAL	No range	User unit/sec	No default	Velocity setpoint.
Acceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Acceleration rate. S-curve: Maximum acceleration.
Deceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Deceleration rate. S-curve: Unused.
Jerk	LREAL	No range	User unit/sec^3	No default	Trapezoidal: 0 (zero). S-curve: Constant jerk.
BufferMode	SINT	0, 5	N/A	No default	• 0 = Abort. • 1 = Buffer. • 2 = Blend to active. • 3 = Blend to next. • 4 = Blend to low velocity. • 5 = Blend to high velocity.

4.2.11.4.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates the move completed successfully. The Command Position has reached the endpoint.
Busy	BOOL	No range	N/A	High from the moment the Execute input goes high until the time the move is ended.
Active	BOOL	No range	N/A	Indicates this move is the Active move.
CommandAborted	BOOL	No range	N/A	Indicates the move was aborted.
Error	BOOL	No range	N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE.

4.2.11.4.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- It is typically used with Abort specified at the BufferMode input.
  - If BufferMode is not Abort, this move is identical to an MC_MoveRelative.

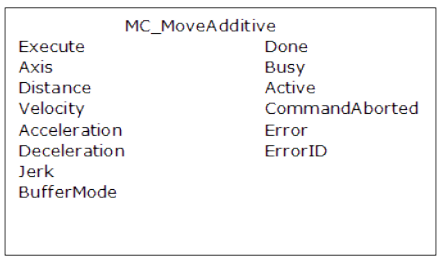


Figure 4-74: MC_MoveAdditive

4.2.11.4.3.1 Time Diagram

These images show two examples of the combination of two Function Blocks (FBs) while the axis is in Discrete Motion state:

- The left part of the diagram illustrates when the second FB is called **after** the first one.
  - If the first reaches the commanded distance 6000, and the velocity is 0 (zero), the output **Done** causes the second FB to move to the distance 10000.
- The right part of the diagram illustrates when the second move FB starts the execution **while** the first FB is still executing.
  - The first motion is interrupted and aborted by the Test signal during the constant velocity of the first FB.
  - The second FB:
    - Adds on the previous commanded position of 6000 the distance 4000.
    - Moves the axis to the resulting position of 10000.

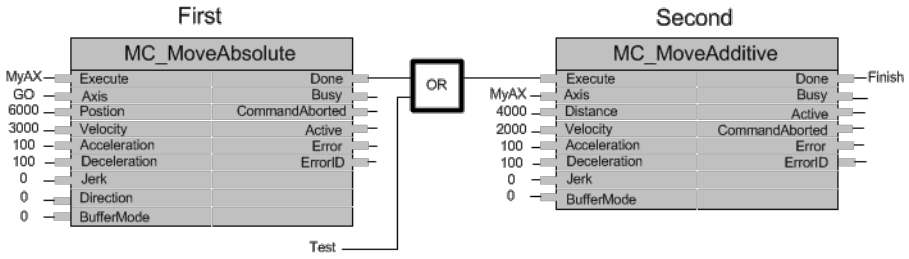


Figure 4-75: Time Diagrams: First and Second FBs

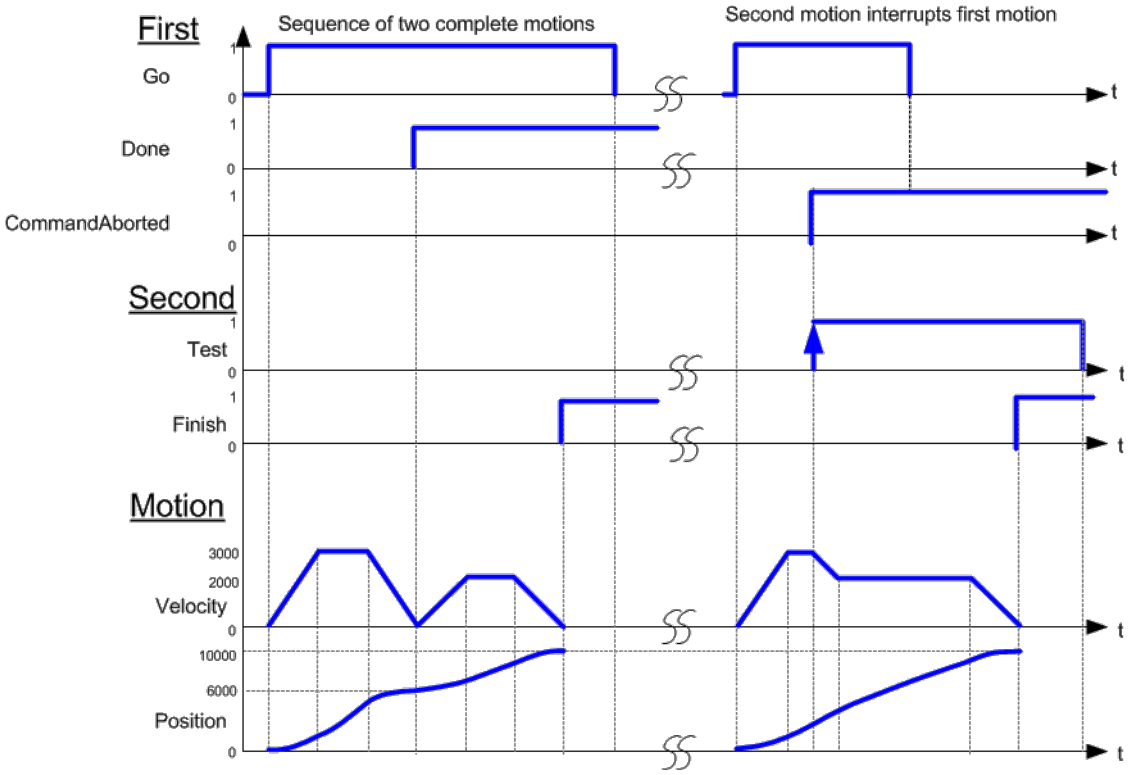
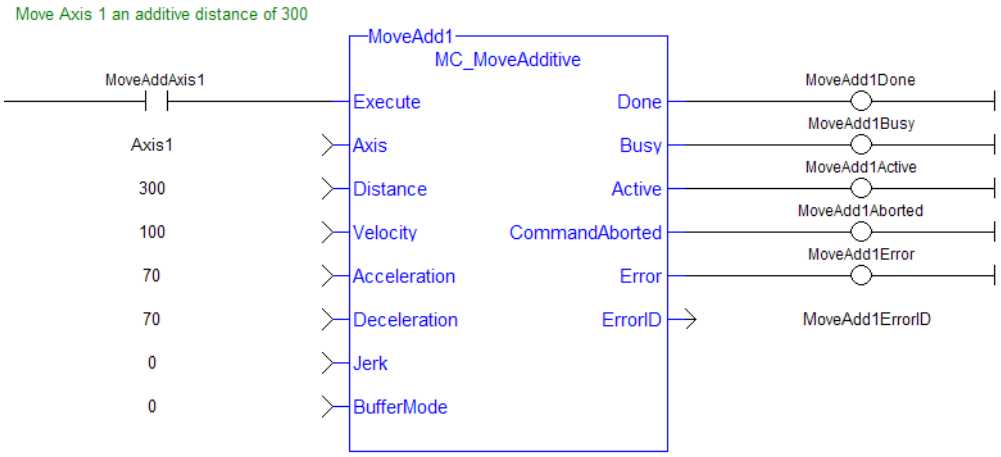


Figure 4-76: Time Diagram

4.2.11.4.4 FBD Language Example

Not available.

4.2.11.4.5 FFLD Language Example



4.2.11.4.6 IL Language Example

Not available.

4.2.11.4.7 ST Language Example

```
      (* MC_MoveAdditive ST example *)

Inst_MC_MoveAdditive( MovAddReq, Axis1, 123.456, 100.0, 100.0, 100.0, 0, 0 );
   //Inst_MC_MoveAdditive is an instance of MC_MoveAdditive function block
MovAddDone := Inst_MC_MoveAdditive.Done;
   //store Done output into user defined variable
```

See Also

- Call a Function Block
- Differences between Functions and Function Blocks

4.2.11.5 MC_MoveContVel

PLCopen ✓

Function Block - Performs a single-axis non-ending move at a specified velocity with the option of continually updating the ongoing motion with the current input parameters.

4.2.11.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	FALSE, TRUE	N/A	No default	On the rising edge, request to queue the move.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.

Input	Data Type	Range	Unit	Default	Description
Velocity	LREAL	All finite values.	User unit/sec	No default	The target axis velocity. • Negative values of velocity move the axis in the negative direction. • Positive values move the axis in the positive direction. • A value of 0 (zero) is valid. • 0 (zero) indicates a deceleration to zero velocity.
Acceleration	LREAL	Positive values.	User unit/sec 2	No default	Trapezoidal: Acceleration rate. S-curve: Maximum acceleration.
Deceleration	LREAL	Positive values.	User unit/sec 2	No default	Trapezoidal: Deceleration rate. S-curve: Unused.
Jerk	LREAL	No range	User unit/sec 3	No default	Trapezoidal: 0 (zero). S-curve: Constant jerk.

Input	Data Type	Range	Unit	Default	Description
ContinuousUpdate	BOOL	FALSE, TRUE	N/A	No default	Determines if the inputs of the function block are re-evaluated every cycle or if they are only evaluated on the rising edge of **Execute**. • If TRUE, when the function block is triggered (on the rising edge of **Execute**), the function block uses the current updated values of the input variables and apply it to the ongoing movement of the axis. • This continues as long as **ContinuousUpdate** stays TRUE. • The impact of **ContinuousUpdate** ends as soon as the function block is no longer busy (**Busy** output is FALSE) or **ContinuousUpdate** is set to FALSE.
BufferMode	SINT	See Description	N/A	No default	The specified buffer mode. • MC_BUFFER_MODE_ABORTING = 0 = Abort • MC_BUFFER_MODE_BUFFERED = 1 = Buffered • MC_BUFFER_MODE_BLENDING_PREVIOUS = 2 = Blending Previous • MC_BUFFER_MODE_BLENDING_NEXT = 3 = Blending Next • MC_BUFFER_MODE_BLENDING_LOW = 4 = Blending Low • MC_BUFFER_MODE_BLENDING_HIGH = 5 = Blending High

4.2.11.5.2 Outputs

Output	Data Type	Range	Unit	Description
InVelocity	BOOL	No range	N/A	Indicates the command velocity has reached the programmed velocity.
Busy	BOOL	No range	N/A	High from the moment the Execute input goes high until the time the move is ended.
Active	BOOL	No range	N/A	Indicates this move is the Active move.
CommandAborted	BOOL	No range	N/A	Indicates the move was aborted.
Error	BOOL	No range	N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE.

4.2.11.5.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

NOTE

This function block starts a motion-related action and stores data for calculations and error checking.
See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- After **MC_MoveContVel** execution begins (**Execute** input - low to high), follow up changes to input parameters immediately affect the ongoing motion, without requiring an additional low to high transition on the **Execute** input.
- This type of move can be terminated with the "MC_Halt" (→ p. 449) function block or by aborting it with another move.

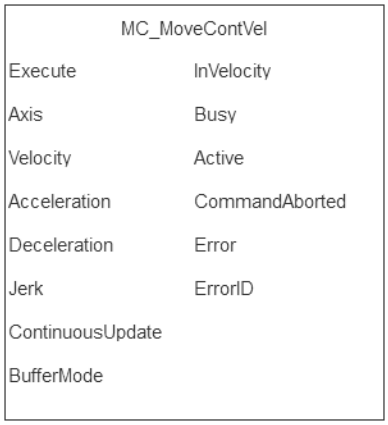


Figure 4-77: MC_MoveContVel

4.2.11.5.3.1 Time Diagram

This example shows the behavior of the combination of a "MC_Stop" (→ p. 423) function block with a MC_MoveContVel function block.

- A rotating axis is ramped down with FB2 "MC_Stop" (→ p. 423).
- The axis rejects motion commands as long as "MC_Stop" (→ p. 423) parameter **Execute** = TRUE.
- FB1 MC_MoveContVel reports an error indicating the busy "MC_Stop" (→ p. 423) command.

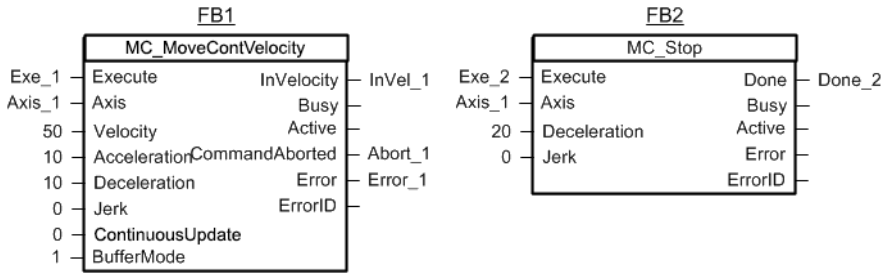


Figure 4-78: Time Diagrams: First and Second FBs

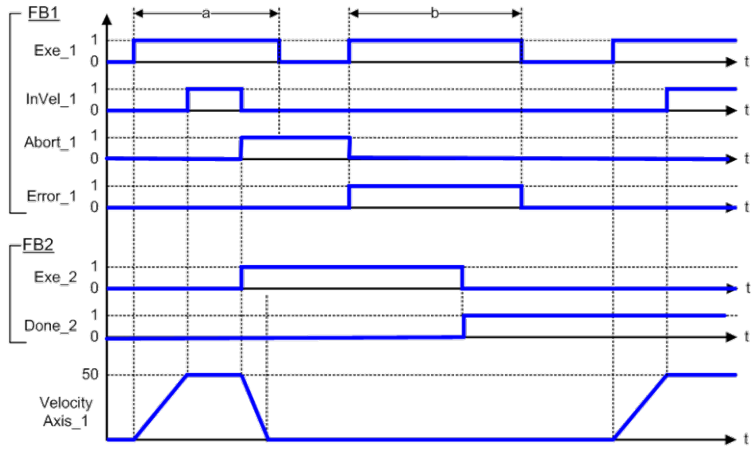
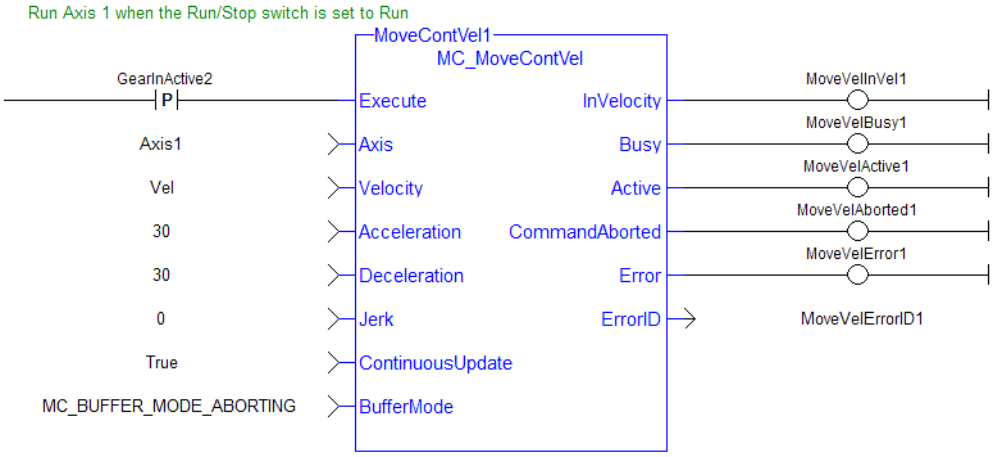


Figure 4-79: Time Diagram

4.2.11.5.4 FBD Language Example

Not available.

4.2.11.5.5 FFLD Language Example



4.2.11.5.6 IL Language Example

Not available.

4.2.11.5.7 ST Language Example

```
(* MC_MoveContVel ST example *)
Inst_MC_MoveContVel( MovVelReq , Axis1, Vel, 100.0, 100.0, 0, True, MC_
BUFFER_MODE_ABORTING );
```

See Also

- Call a Function Block
- Differences between Functions and Function Blocks
- "MC_MoveVelocity" (➔ p. 470)

4.2.11.6 MC_MoveRelative

PLCopen

Function Block - Performs a single-axis move for a specified distance to perform incremental motion.

4.2.11.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to queue the move.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
Distance	LREAL	No range	User units	No default	Distance.
Velocity	LREAL	No range	User unit/sec	No default	Velocity setpoint.
Acceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Acceleration rate. S-curve: Maximum acceleration.
Deceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Deceleration rate. S-curve: Unused.
Jerk	LREAL	No range	User unit/sec^3	No default	Trapezoidal: 0 (zero). S-curve: Constant jerk.
BufferMode	SINT	0, 5	N/A	No default	• 0 = Abort. • 1 = Buffer. • 2 = Blend to active. • 3 = Blend to next. • 4 = Blend to low velocity. • 5 = Blend to high velocity.

4.2.11.6.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates the move completed successfully. The Command Position has reached the endpoint.
Busy	BOOL	No range	N/A	High from the moment the Execute input goes high until the time the move is ended.
Active	BOOL	No range	N/A	Indicates this move is the Active move.
CommandAborted	BOOL	No range	N/A	Indicates the move was aborted.
Error	BOOL	No range	N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE.

4.2.11.6.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

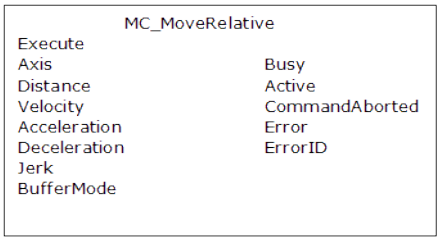


Figure 4-80: MC_MoveRelative

4.2.11.6.3.1 Time Diagram

These images show two examples of the combination of two relative move Function Blocks (FBs).

- The left part of the diagram illustrates when the second FB is called **after** the first one.
  - If the first reaches the commanded position of 6000, and the velocity is 0 (zero), the output **Done** causes the second FB to move to the distance 10000.
- The right part of the diagram illustrates when the second move FB starts the execution **while** the first FB is still executing.
  - The first motion is interrupted and aborted by the Test signal during the constant velocity of the first FB.
  - The second FB **adds on the actual position** of 3250 the distance 4000 and moves the axis to the resulting position of 7250.

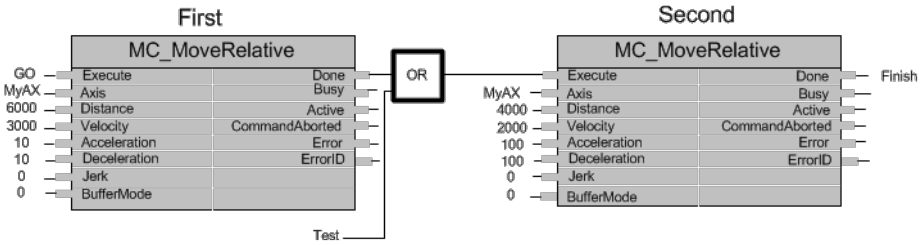


Figure 4-81: Time Diagrams: First and Second FBs

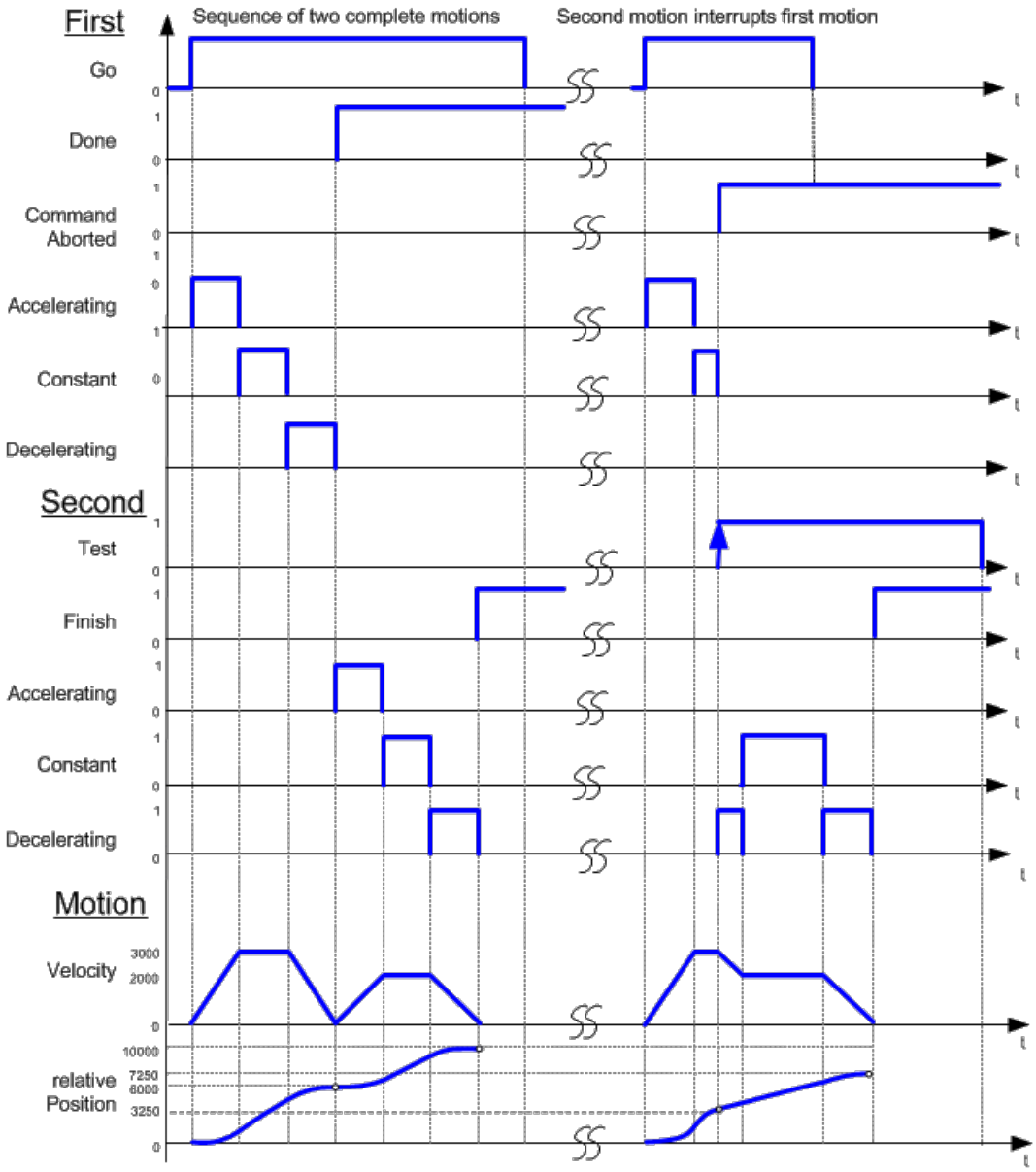
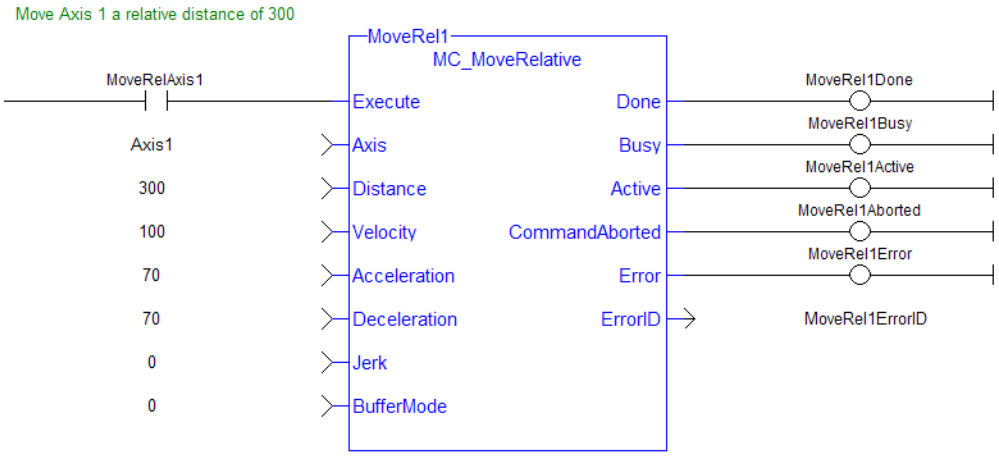


Figure 4-82: Time Diagram

4.2.11.6.4 FBD Language Example

Not available.

4.2.11.6.5 FFLD Language Example



4.2.11.6.6 IL Language Example

Not available.

4.2.11.6.7 ST Language Example

```
(* MC_MoveRelative ST example *)
Inst_MC_MoveRelative( MovRelReq, Axis1, 10.0, 200.0,150.0, 150.0, 0,0 );
MovRelDone := Inst_MC_MoveRelative.Done;  //store Done output into user
defined variable
```

See Also

- Call a Function Block
- Differences between Functions and Function Blocks

4.2.11.7 MC_MoveSuperimp

PLCopen

Function Block - Performs a single-axis move superimposed upon the currently executing move.

4.2.11.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to queue the move.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
Distance	LREAL	No range	User units	No default	Distance.
VelocityDiff	LREAL	No range	User unit/sec	No default	Velocity rate.
Acceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Acceleration rate. S-curve: Maximum acceleration.
Deceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Deceleration rate. S-curve: Unused.
Jerk	LREAL	No range	User unit/sec^3	No default	Trapezoidal: 0 (zero). S-curve: Constant jerk.

Input	Data Type	Range	Unit	Default	Description
BufferMode	SINT	0, 5	N/A	No default	• 0 = Abort. • 1 = Buffer. • 2 = Blend to active. • 3 = Blend to next. • 4 = Blend to low velocity. • 5 = Blend to high velocity.

4.2.11.7.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates the move completed successfully. The Command Position has reached the endpoint.
Busy	BOOL	No range	N/A	High from the moment the Execute input goes high until the time the move is ended.
Active	BOOL	No range	N/A	Indicates this move is the Active move.
CommandAborted	BOOL	No range	N/A	Indicates the move was aborted.
Error	BOOL	No range	N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE.

4.2.11.7.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

4.2.11.7.3.1 Usage

This function block:

- Provides a way to smoothly apply a shift in axis position while it is executing a move.
- Is commonly used with "MC_TouchProbe" (→ p. 428) for performing position corrections on the slave axis in a **Mark-to-Machine** registration application.

TIP

MC_MoveSuperimp performs a similar function to the SlaveOffset input in the "MC_CamIn" (→ p. 476) function block.
It has the additional features of setting the velocity, acceleration, deceleration, and jerk motion parameters.

- A superimposed move is executed similar to a "MC_MoveRelative" (→ p. 464) move using the specified **Distance**, **Velocity**, **Acceleration**, **Deceleration**, and **Jerk** values.
  - The interpolated command generated by a superimposed move is added to the command of the currently executing move.
  - Subsequent calls to MC_MoveSuperimp can abort or blend to an executing MC_MoveSuperimp move.

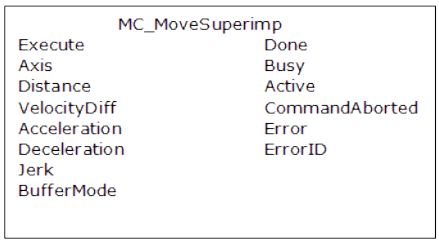


Figure 4-83: MC_MoveSuperimp

4.2.11.7.3.2 Time Diagram

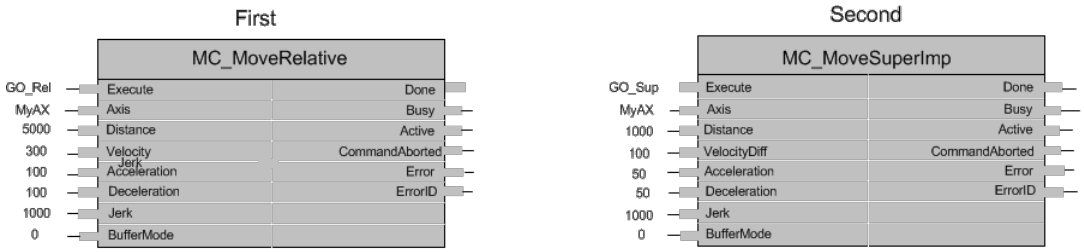


Figure 4-84: Time Diagrams: First and Second FBs

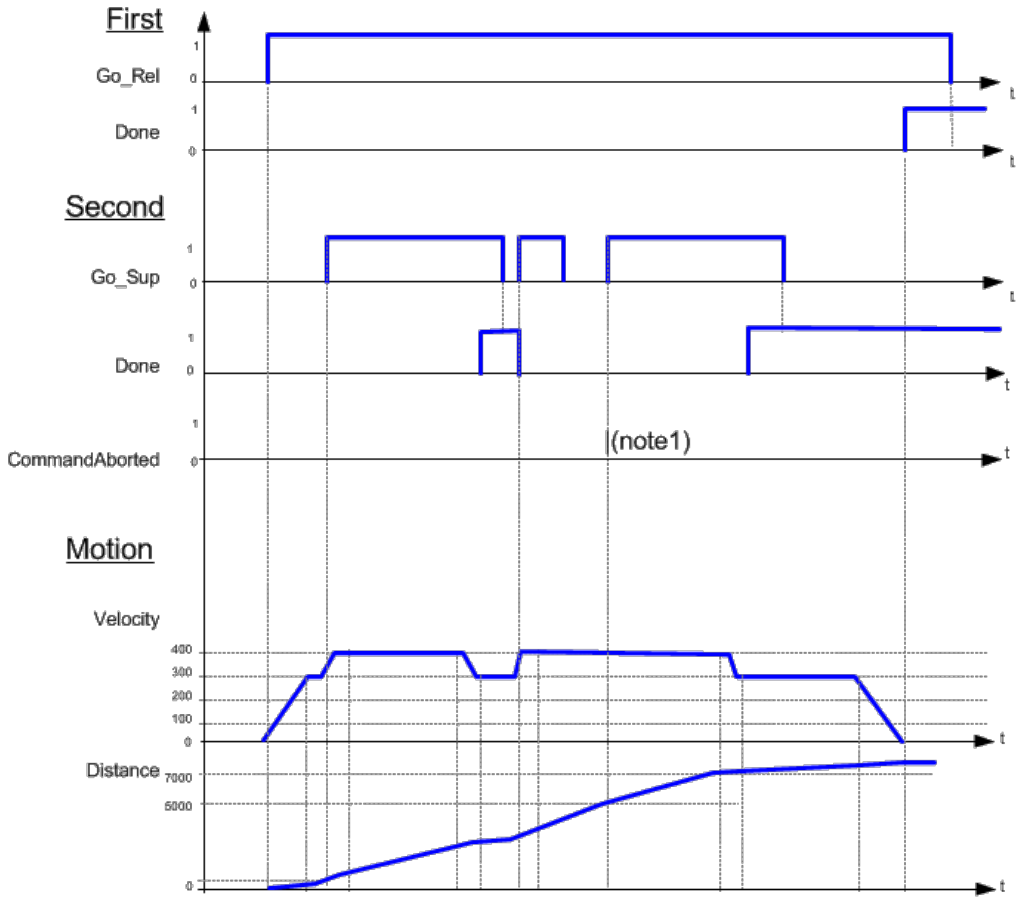


Figure 4-85: Time Diagram

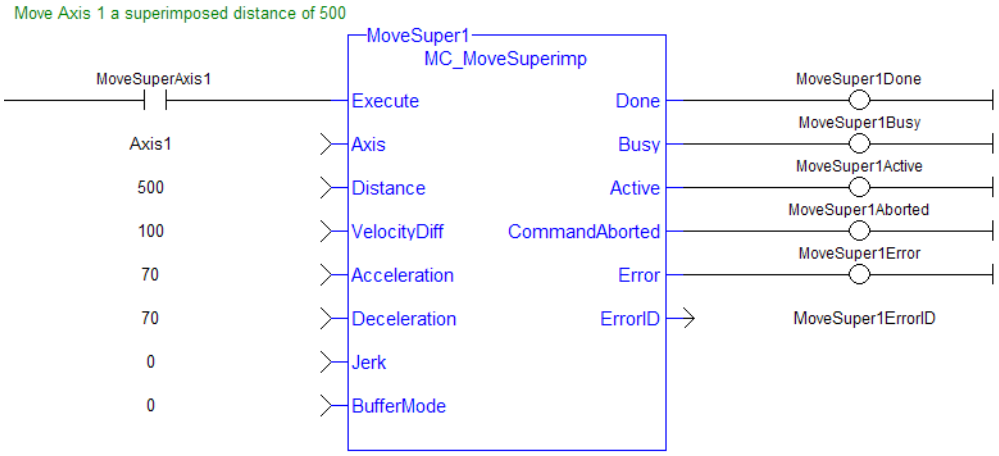
NOTE

1. The CommandAborted is not visible here because the new command works on the same instance.
2. The end position is between 7000 and 8000, depending on the timing of the aborting of the second command set for the MC_MoveSuperimposed.

4.2.11.7.4 FBD Language Example

Not available.

4.2.11.7.5 FFLD Language Example



4.2.11.7.6 IL Language Example

Not available.

4.2.11.7.7 ST Language Example

```
(* MC_MoveSuperimp ST example *)
Inst_MC_MoveSuperimp( MovSupReq, Axis1, 123.555, 10.0, 100.0, 100.0, 0, 0 );
MovSupDone := Inst_MC_MoveSuperimp.Done;  //store Done output into user
defined variable
```

See Also

- Call a Function Block
- Differences between Functions and Function Blocks

4.2.11.8 MC_MoveVelocity

PLCopen

Function Block - Performs a single-axis, non-ending move at a specified velocity.

4.2.11.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	FALSE, TRUE	N/A	No default	On the rising edge, request to queue the move.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
Speed	LREAL	Positive values.	User unit/sec	No default	The target axis speed. Direction is specified by the **Direction** input parameter.

Input	Data Type	Range	Unit	Default	Description
Acceleration	LREAL	Positive values.	User unit/sec^2	No default	Trapezoidal: Acceleration rate. S-curve: Maximum acceleration.
Deceleration	LREAL	Positive values.	User unit/sec^2	No default	Trapezoidal: Deceleration rate. S-curve: Unused.
Jerk	LREAL	No range	User unit/sec^3	No default	Trapezoidal: 0 (zero). S-curve: Constant jerk.
Direction	SINT	0, 1	N/A	No default	• A 0 or False value specifies that the axis should move in the positive direction. • A 1 or True value specifies that the axis should move in the negative direction.
BufferMode	SINT	• MC_BUFFER_MODE_ABORTING • MC_BUFFER_MODE_BUFFERED • MC_BUFFER_MODE_BLENDING_PREVIOUS • MC_BUFFER_MODE_BLENDING_NEXT • MC_BUFFER_MODE_BLENDING_LOW • MC_BUFFER_MODE_BLENDING_HIGH	N/A	No default	The specified buffer mode. See Buffer Modes for more information.

4.2.11.8.2 Outputs

Output	Data Type	Range	Unit	Description
InVelocity	BOOL	No range	N/A	Indicates the command velocity has reached the programmed velocity.
Busy	BOOL	No range	N/A	High from the moment the Execute input is one-shot to the time the move is ended.
Active	BOOL	No range	N/A	Indicates this move is the Active move.

Output	Data Type	Range	Unit	Description
CommandAborted	BOOL	No range	N/A	Indicates the move was aborted.
Error	BOOL	No range	N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE.

4.2.11.8.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

TIP

Consider using the "MC_MoveContVel" (→ p. 459) function block.
It is more flexible and allows for the continuous update of motion parameters.

- This type of move can be terminated with the MC_Halt function block or by aborting it with another move.

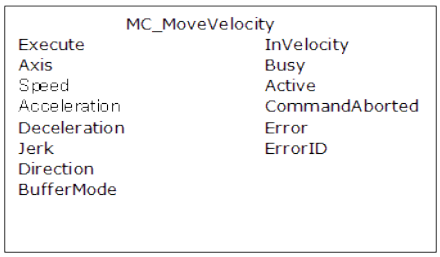


Figure 4-86: MC_MoveVelocity

4.2.11.8.3.1 Time Diagram

These images show two examples of the combination behavior of an MC_Stop FB with a MC_ MoveVelocity FB.

- A rotating axis is ramped down with FB2 MC_Stop.
- The axis rejects motion commands as long as MC_Stop parameter “Execute” = TRUE.
- FB1 MC_MoveVelocity reports an error indicating the busy MC_Stop command.

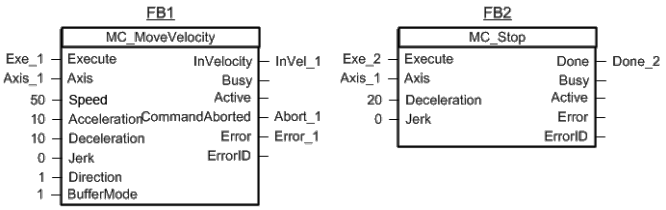


Figure 4-87: Time Diagrams: First and Second FBs

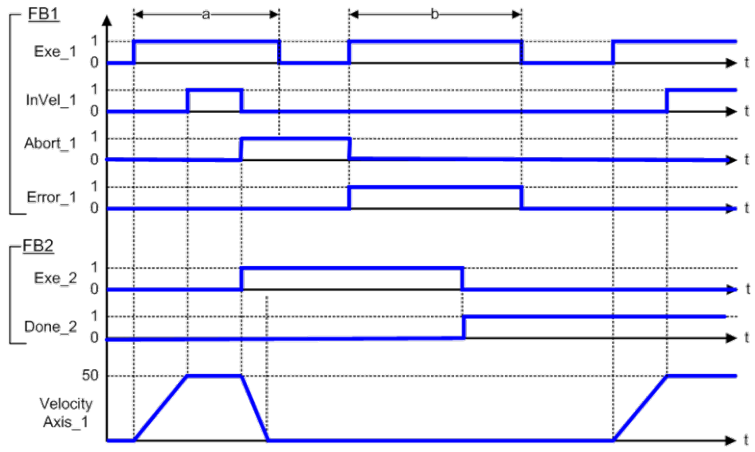
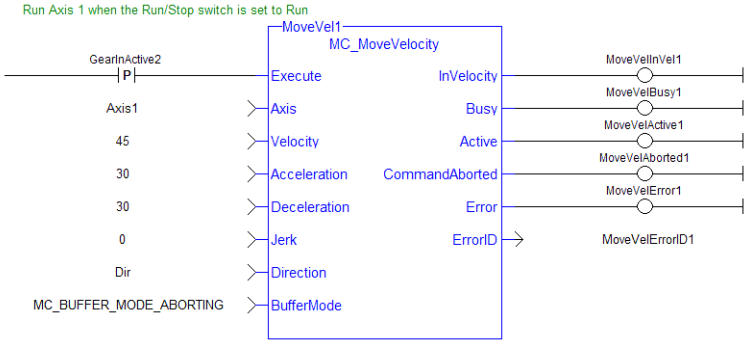


Figure 4-88: Time Diagram

4.2.11.8.4 FBD Language Example

Not available.

4.2.11.8.5 FFLD Language Example



4.2.11.8.6 IL Language Example

Not available.

4.2.11.8.7 ST Language Example

```
            (* MC_MoveVelocity ST example *)

Inst_MC_MoveVelocity( MovVelReq , Axis1, 200.0, 100.0, 100.0, 0, True, MC_
BUFFER_MODE_ABORTING );
```

See Also

- Call a Function Block
- Differences between Functions and Function Blocks
- "MC_MoveContVel" (➔ p. 459)

4.2.11.9 MC_RemSuperAxis

PLCopen

Function Block - Remove an axis from the axis's list of assigned, superimposed axes.

4.2.11.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	N/A	N/A	N/A	Enables execution.
Axis	AXIS_REF	AXIS_NUM1, 256	N/A	N/A	Axis whose list of assigned superimposed axes is updated.
SuperimposedAxis	UINT	N/A	N/A	N/A	Axis number of the superimposed axis that is removed from Axis's list of assigned superimposed axes. A value of 0 (zero) removes all superimposed axes from Axis's list.

4.2.11.9.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Execution successful.

4.2.11.9.3 Remarks

NOTE

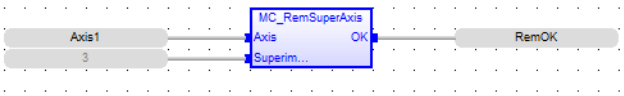
This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

This feature allows the application program to superimpose the moves of multiple axes ("Superimposed Axes") on top of the move of another axis ("Receiving Axis").

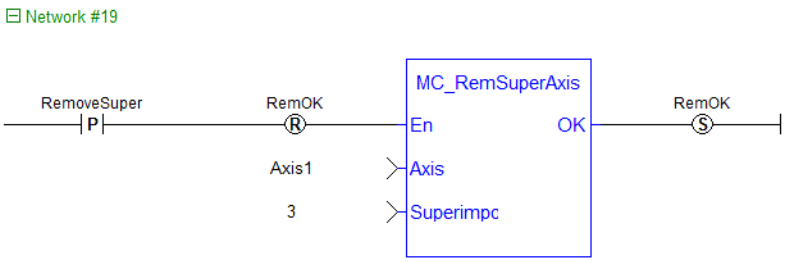
- This is performed internally by adding the command deltas of the Superimposed Axes to the command delta of the Receiving Axis.
- A maximum of four different Superimposed Axes can be superimposed upon a Receiving Axis.

- If the value at **SuperimposedAxis** is 0 (zero), all assigned superimposed axes are removed from Axis's list.
- The **OK** output goes high to indicate the function executed successfully.
- If the **OK** output **does not** go high, one of these errors is detected:
  - The axis is not a valid axis.
  - The axis is not a servo or virtual axis.

4.2.11.9.4 FBD Language Example



4.2.11.9.5 FFLD Language Example



4.2.11.9.6 IL Language Example

Not available.

4.2.11.9.7 ST Language Example

```
RemOK := MC_RemSuperAxis(Axis1,3);
```

See Also

"MC_AddSuperAxis" (→ p. 447)

4.2.11.10 MC_SetOverride

PLCopen

Function Block - Writes velocity and acceleration override factors.

4.2.11.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	Request to write the override factors.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
VelFactor	REAL	0.0, 2.0	N/A	No default	Velocity override factor.

4.2.11.10.2 Outputs

Output	Data Type	Range	Unit	Description
Enabled	BOOL			Indicates the override values have been written.
Busy	BOOL			Indicates this function block is executing.
Error	BOOL			Indicates an invalid input was specified.
ErrorID	INT			Indicates the error if **Error** output is TRUE.

4.2.11.10.3 Remarks

-
- A change in the velocity override factor takes effect immediately on the active move.
- The velocity override factor is applied to the programmed velocity (of a "MC_MoveAbsolute" (→ p. 451), "MC_MoveAdditive" (→ p. 456), "MC_MoveRelative" (→ p. 464), "MC_MoveSuperimp" (→ p. 467), or "MC_MoveVelocity" (→ p. 470) function block) to determine the command velocity:

```
command velocity = programmed velocity * VelFactor
```

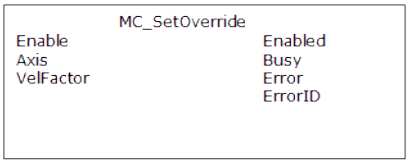
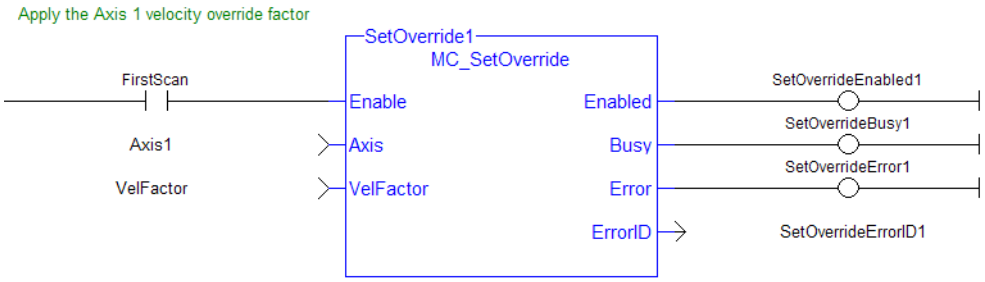


Figure 4-89: MC_SetOverride

4.2.11.10.4 FBD Language Example

Not available.

4.2.11.10.5 FFLD Language Example



4.2.11.10.6 IL Language Example

Not available.

4.2.11.10.7 ST Language Example

```
            (* MC_SetOverride ST example *)

VelFactor := 1.25 ;   //set the velocity factor to 1.25 (125%)
Inst_MC_SetOverride( TRUE , Axis1, VelFactor ); // Inst_MC_Setoverride is an
instance of MC_SetOverride
```

4.2.12 Profile

These functions provide commands for slave axes, such as cams and gearing.

Name	Description
MC_CamIn	Performs a slave axis move which follows the master axis based on the Cam Table specified by CamTableID.
MC_CamOut	Disengages the slave axis from a MC_CamIn move.
MC_ CamResumePos	Returns the slave axis position for resuming an "MC_CamIn" (→ p. 476) move.
MC_CamStartPos	Returns the slave axis position for starting an "MC_CamIn" (→ p. 476) move.
MC_ CamTblSelect	Defined to read and initialize the specified profile.
MC_GearIn	Performs a slave axis move which follows the master axis based on the ratio specified by RatioNumerator and RatioDenominator.
MC_GearInPos	Performs a slave axis move which follows the master axis based on the ratio specified by RatioNumerator and RatioDenominator.
MC_GearOut	Disengages the slave axis from a MC_GearIn or MC_GearInPos move.
MC_Phasing	Performs a master position phase shift for the slave axis.
MC_SyncSlaves	Allows the application to specify what slave axes are to be synchronized and which master they follow.

4.2.12.1 MC_CamIn

PLCopen

Function Block - Performs a slave axis move which follows the master axis based on the Cam Table specified by CamTableID.

4.2.12.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to queue the move.
Master	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
Slave	AXIS_REF	1, 256	N/A	No default	AXIS_REF.AXIS_NUM is the slave axis number.
MasterOffset	LREAL	No range	User units	No default	Master axis offset. This input is not used if the StartMode input is set to 1 for Resume Mode.
SlaveOffset	LREAL	No range	User units	No default	Slave axis offset. This input is not used if the StartMode input is set to 1 for Resume Mode.
MasterScaling	LREAL	No range	User units	No default	Master axis scale factor. • This input is not used if the StartMode input is set to 1 for Resume Mode. • Scaling must be a positive value greater than 0 (zero).
SlaveScaling	LREAL	No range	User units	No default	Slave axis scale factor. • This input is not used if the StartMode input is set to 1 for Resume Mode. • Scaling must be a positive value greater than 0 (zero).

Input	Data Type	Range	Unit	Default	Description
StartMode	INT	0, 1	N/A	No default	Starting mode of the cam profile. The function block uses values that were in effect during the most recently programmed MC_CamIn move for the slave axis. **0 = Start Mode** • Input indicates if the axis should start a MC_CamIn move as an initial cam start. **1 = Resume Mode** • Input indicates if the axis should resume the most recently programmed MC_CamIn move. • The inputs MasterOffset, SlaveOffset, MasterScaling, and SlaveScaling are not used.
CamTableID	INT	No range	N/A	No default	Profile ID number.
BufferMode	SINT	• MC_BUFFER_MODE_ABORTING • MC_BUFFER_MODE_BUFFERED	N/A	No default	The Buffer mode for CamIn block. • MC_BUFFER_MODE_ABORT is used to abort an existing camming, gearing, point-to-point, or velocities move. • MC_BUFFER_MODE_BUFFERED is only used when an endpoint is specified with the previous move. • This limits its use to when the previous move is a point-to-point move ("MC_MoveAbsolute" (→ p. 451) or "MC_MoveRelative" (→ p. 464)). • See Buffer Modes for more information.

4.2.12.1.2 Outputs

Output	Data Type	Range	Unit	Description
InSync	BOOL	0, 1	N/A	Indicates the slave axis is in sync with the profile.
Busy	BOOL	0, 1	N/A	Indicates this function block is executing.
Active	BOOL	0, 1	N/A	Indicates this move is the Active move.
CommandAborted	BOOL	0, 1	N/A	Indicates the move was aborted.
Error	BOOL	0, 1	N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
ErrorID	INT	No range	N/A	Indicates the error if the Error output is high.
EndOfProfile	BOOL	0, 1	N/A	Indicates the end of profile has been reached. • If the profile is periodic, this output is set to ON for one ladder scan. • If the profile is not periodic, the output remains ON while outside the range of the profile. • See "Periodic or Not Periodic" (→ p. 480) for more information.

4.2.12.1.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- This function block is used to either:
  - Initiate a new MC_CamIn move.
  - Resume a previously programmed MC_CamIn move.
- See "MC_CamStartPos" (→ p. 488) or "MC_CamResumePos" (→ p. 486) for more information about positioning the slave axis prior to calling MC_CamIn.

4.2.12.1.3.1 Abort Camming

NOTE

Ending camming is also possible with other single axis function blocks (e.g., "MC_MoveRelative" (→ p. 464) or "MC_MoveAbsolute" (→ p. 451)).

There are two options to stop camming after MC_CamIn has been called:

- "MC_CamOut" (→ p. 484) continues motion at the instantaneous final actual velocity of the slave axis when it is called.
  - Axis motion continues at that final actual velocity.
- "MC_Halt" (→ p. 449) (with buffer mode input = 0) decelerates axis motion to 0 (zero) speed and stop motion.
- The master / slave relationship between the two axes is ended when MC_CamOut or MC_Halt is called.

Figure 4-90: MC_CamIn

4.2.12.1.3.2 Usage

The slave axis immediately locks on to the Cam Table profile.

- The **Master Offset** is used to shift the profile along the master axis.
- The **Master Scaling** defines the range of the profile along the master axis.
- The **Slave Offset** is used to shift the profile along the Slave axis.
- The **Slave Scaling** defines the range of the profile along the slave axis.

4.2.12.1.3.3 Periodic or Not Periodic

- If the profile is periodic, when the end of profile is reached, the profile continues at the start of the profile.
  - The EndOfProfile output is ON for 1 ladder scan.
- If the profile is not periodic, when the end of profile is reached, the slave axis stops and remains at the end of the profile until the master axis returns to within the profile range as defined by MasterScaling.
  - The EndOfProfile output remains ON anytime the master axis is outside of the profile range.

4.2.12.1.3.4 Adjustments Computation is Done

When cam is first started, offsets are adjusted if necessary.

- If the master is not absolute, then master offset = master offset + starting position.
- If the slave is not absolute, then slave offset = slave offset + starting position.

At run-time:

- Master position for profile = master position - master offset.
  - Use master position for profile table to obtain slave profile position.
- Slave commanded position = slave profile position + slave offset.

4.2.12.1.3.5 Dynamically Change a Cam Profile

MC_CamIn can be used to dynamically change from one cam profile to another.

Care must be taken when doing this to avoid unexpected motion.

TIP

Tips to dynamically change cam profiles:

- Verify the first cam's last position and the replacement cam's first position are the same.
  - Offsets as set by"MC_CamTblSelect" (→ p. 490) affect actual cam position.
- Verify the first cam's last velocity and the replacement cam's first velocity do not cause any unexpected motion.
- Jumps can be eliminated by defining the present cam as **Cyclic** and defining the replacement cam as an **Absolute Master** and **Slave**, as set by the "MC_CamTblSelect" (→ p. 490) inputs.
  - This eliminates any possible small error accumulating when the cam is switched.

4.2.12.1.4 Examples

These examples use the "MC_CamIn" (→ p. 476) image showing the cam profile **MyProfile**:

"Example 1" (→ p. 481)

"Example 2" (→ p. 482)

"Example 3" (→ p. 482)

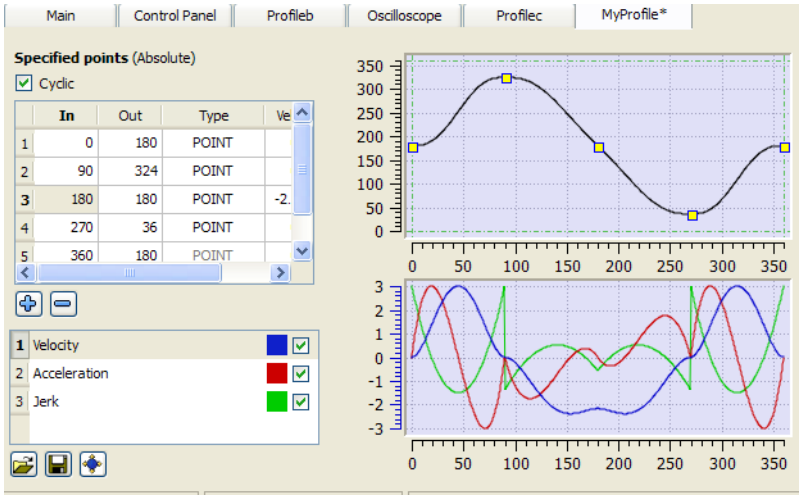
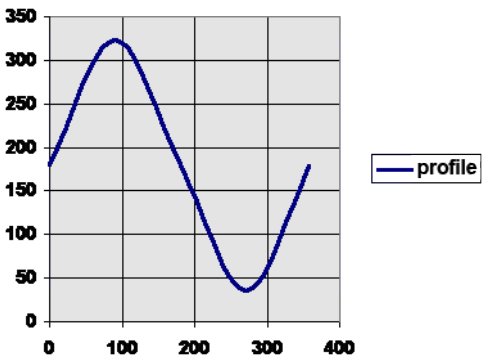


Figure 4-91: MC_CamIn Example

4.2.12.1.4.1 Example 1

Example 1- MC_CamIn Cam Profile

Profile	MyProfile
Cycle	NO
MasterAbsolute	YES
SlaveAbsolute	YES
MasterOffset	0.0
SlaveOffset	0.0
MasterScaling	360.0
SlaveScaling	360.0
Initial Master position	0.0
Initial Slave position	180.0



After `MC_CamTblSelect` and `MC_CamIn` are programmed with the parameters, the slave axis is locked on to the profile.

- Since both axes have zero offsets, the profile is not shifted in either axis.
- The initial condition of the master axis at position 0 (zero), yields a slave command position of 180.0.
- As the master axis moves positive, the slave position follows the profile.
- When the master position is at 90.0, the slave is commanded to 324.0.
- See curve below where in = 90, out = 324.

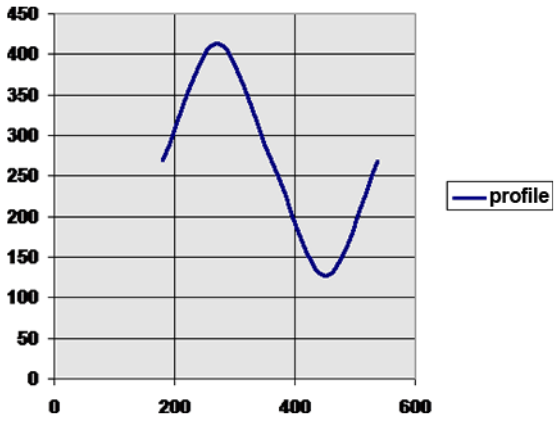
- The slave follows the profile as the master axis moves until the master axis reaches a position of 360.0.
- At this time, the slave is commanded to 180.0.

- If the master were to continue to move past 360.0, the slave commanded position would remain at 180.0 since the Cyclic input is false.
- If the master moves negative, and its position returns to less than 360.0, the slave follows the profile again.

4.2.12.1.4.2 Example 2

Example 2 - MC_CamIn Cam Profile

Profile	MyProfile
Cycle	YES
MasterAbsolute	NO
SlaveAbsolute	NO
MasterOffset	0.0
SlaveOffset	0.0
MasterScaling	360.0
SlaveScaling	360.0
Initial Master position	180.0
Initial Slave position	90.0



After `MC_CamTblSelect` and `MC_CamIn` are programmed with the parameters, the slave axis is locked on to the profile.

- Since both axes have zero offsets, the profile is not shifted in either axis.
- Neither the **MasterAbsolute** nor **SlaveAbsolute** input is on, so the profile is relative to the axes initial positions.
- Specifically, the initial condition of the master axis at position 180 would represent a master profile position of 0 (180-180).
- This yields a slave command position of 270 (180 + 90).

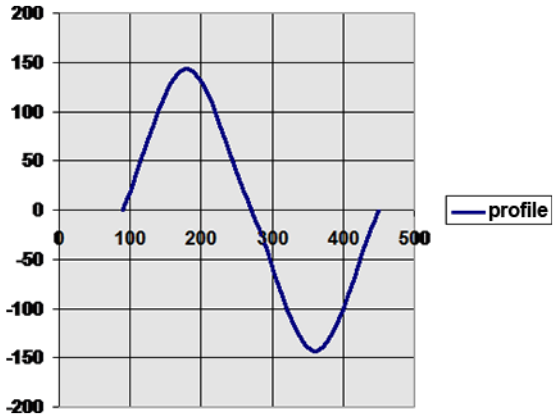
- As the master axis moves positive, the slave position follows the profile.
- When the master position is at 270, the slave is commanded to 414.0 (324 + 90).
- The slave follows the profile as the master axis moves until the master axis reaches a position of 540.
- At this time the slave is commanded to 270.0 (180 + 90).

- If the master continues to move past 540.0, the slave commanded position follows the profile from the beginning since the Cyclic input is TRUE.
- When the master reaches a position of 630, the slave is commanded to a position of 414.0 (324 + 90).

4.2.12.1.4.3 Example 3

Example 3 - MC_CamIn Cam Profile

Profile	MyProfile
Cycle	NO
MasterAbsolute	YES
SlaveAbsolute	YES
MasterOffset	90.0
SlaveOffset	-180
MasterScaling	360.0
SlaveScaling	360.0
Initial Master position	180.0
Initial Slave position	144.0



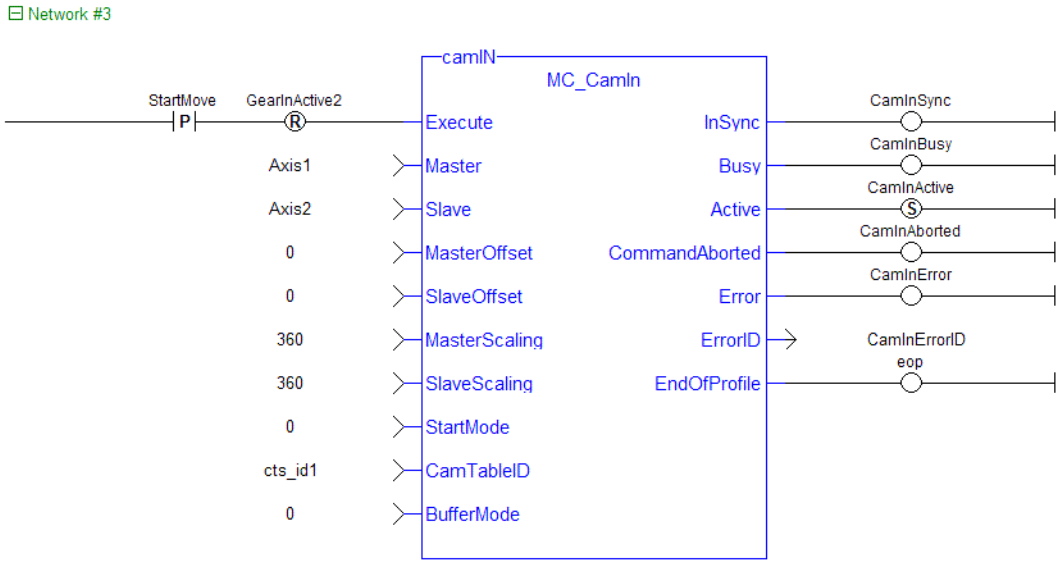
After `MC_CamTblSelect` and `MC_CamIn` are programmed with the parameters, the slave axis is locked on to the profile.

- Since the both axes **have offsets**, the profile **is shifted** along both axes.
- Specifically, the master axis is shifted 90, and the slave axis is shifted -180.
- Initially the master axis position of 180 yields a master position for the profile calculation of 90 (master position 180 - Master offset 90).
- This yields a slave command position of 144 (slave profile command 324 + slave offset (-180)).
- As the master axis moves positive, the slave position follows the profile.
- When the master axis position is at 270, the master position for profile calculation is 180 (270 - 90).
- This yields a slave command position of 0 (zero) (180 + (-180)).
- The slave follows the profile as the master axis moves until the master axis reaches a position of 450.
- The master axis position of 450 yields a master position for profile calculation of 360 (450 - 90).
- The slave command position is 0 (180 + (-180)).
- When the master reaches a position of 450, the slave commanded position remains at 0 (zero) since the Cyclic input is FALSE.

4.2.12.1.5 FBD Language Example

Not available.

4.2.12.1.6 FFLD Language Example



4.2.12.1.7 IL Language Example

Not available.

4.2.12.1.8 ST Language Example

```
(* MC_CamIn ST example *)  //Inst_MC_CamIn is an instance of MC_CamIn
Inst_MC_CamIn( CamStartBool, Axis1, Axis2, 0.0, 0.0, 360.0, 360.0, 0,
CamTableID, 0 );
```

See Also

- Call a Function Block
- Differences Between Functions and Function Blocks
- "MC_CamOut" (→ p. 484)
- "MC_CamResumePos" (→ p. 486)
- "MC_CamStartPos" (→ p. 488)
- "MC_CamTblSelect" (→ p. 490)

4.2.12.2 MC_CamOut

PLCopen

Function Block - Disengages the slave axis from a MC_CamIn move.

4.2.12.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to queue the move.
Slave	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
Acceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Acceleration rate. S-curve: Maximum acceleration.
Deceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Deceleration rate. S-curve: Unused.
Jerk	LREAL	No range	User unit/sec^3	No default	Trapezoidal: 0 (zero). S-curve: Constant jerk.

4.2.12.2.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	0, 1	N/A	Indicates the axis is disengaged from its master.
Busy	BOOL	0, 1	N/A	Indicates this function block is executing.
Active	BOOL	0, 1	N/A	Indicates this move is the Active move.
CommandAborted	BOOL	0, 1	N/A	Indicates the move was aborted.
Error	BOOL	0, 1	N/A	Indicates either: • An invalid input was specified. • No MC_CamIn move was active.
ErrorID	INT	No range	N/A	When the Error output is: • high, this output indicates the type of error. • low, this output is undefined.

4.2.12.2.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- If this function block is called and the active move is not a MC_CamIn move, this function block returns an error and the active move is not aborted.
- This function block:
  - Aborts the active MC_CamIn move.
  - Disengages the axis from its master.
  - Disengages the slave axis from a MC_CamIn move.
  - Commands the axis to continue at its current velocity.
    - The axis continues to run at this velocity until this MC_CamOut move is aborted.

TIP

The current velocity is calculated by taking the average of the actual velocity during the previous 16 cycles.

4.2.12.2.3.1 Cancel the Cam Motion

- An alternative method to cancel the cam motion is a single axis move ("MC_MoveAbsolute" (→ p. 451), "MC_MoveRelative" (→ p. 464), "MC_MoveAdditive" (→ p. 456), "MC_MoveVelocity" (→ p. 470), and "MC_Halt" (→ p. 449)) with the BufferMode input set to 0 (zero).
  - This cancels the "MC_CamIn" (→ p. 476) function and starts the new motion function on the slave axis.
- Many applications prefer calling MC_Halt instead of MC_CamOut because it does not send a velocity command to the slave axis.

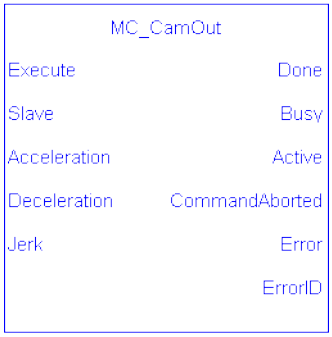
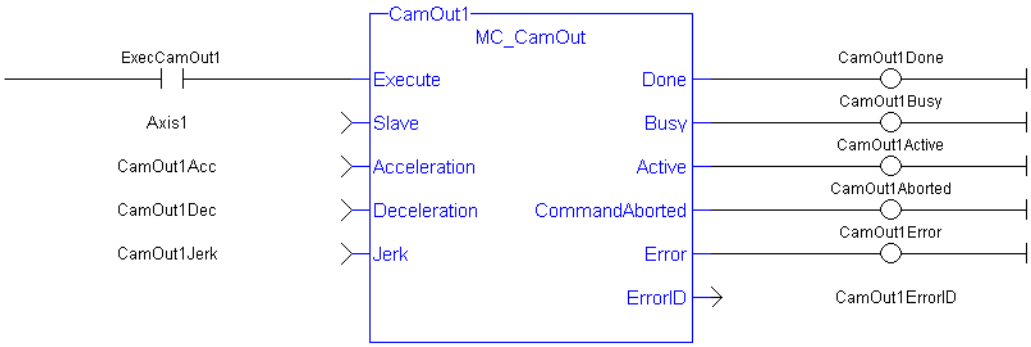


Figure 4-92: MC_CamOut

4.2.12.2.4 FBD Language Example

Not available.

4.2.12.2.5 FFLD Language Example



4.2.12.2.6 IL Language Example

Not available.

4.2.12.2.7 ST Language Example

```
(* MC_CamOut ST example *)
Inst_MC_CamOut(ExecCamOut1,Axis1,CamOut1Acc,CamOut1Dec,CamOut1Jerk);
//Inst_MC_CamOut is an instance of MC_CamOut
```

See Also

- Call a Function Block
- Differences between Functions and Function Blocks
- "MC_CamIn" (→ p. 476)
- "MC_CamTblSelect" (→ p. 490)

4.2.12.3 MC_CamResumePos

PLCopen

Function Block - Returns the slave axis position for resuming an "MC_CamIn" (→ p. 476) move.

4.2.12.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	N/A	N/A	No default	Enables execution.
Master	AXIS_REF	AXIS_NUM - 1, 256	N/A	No default	Master axis. This must be the same as the Master Axis specified for the most recently executed MC_CamIn function block.
Slave	AXIS_REF	AXIS_NUM - 1, 256	N/A	No default	Slave axis. This must be the same as the Slave Axis specified for the most recently executed MC_CamIn function block.
CamTableID	INT	0, 255	N/A	No default	Profile ID number. • This value is generated by "MC_CamTblSelect" (→ p. 490). • This must be the same as the CamTableID specified for the most recently executed MC_CamIn function block.

4.2.12.3.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	TRUE = the function block has successfully calculated the slave position. The slave position is available at the SlavePos output.
Error	BOOL	No range	N/A	TRUE = an invalid input was specified or an error occurred in the calculations. The value at the SlavePos output is undefined.

Output	Data Type	Range	Unit	Description
ErrorID	INT	No range	N/A	Indicates the error if the **Error** output is set to TRUE.
SlavePos	LREAL	No range	User units	If the Done output is TRUE, this output returns the: • position for the slave axis given the profile. • current master axis position. • previously programmed master and slave offsets and scaling.

4.2.12.3.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- Returns the slave axis position for the most recently executed "MC_CamIn" (→ p. 476) profile, based on the current position of the master axis.
  - This slave axis position can be used to command the slave axis to return to the proper location prior to resuming a MC_CamIn function.
  - When calculating the slave axis position, MC_CamResumePos utilizes the master offset, slave offset, master scaling, and slave scaling of the most recently executed MC_CamIn function block for the slave axis.

- The typical application of MC_CamResumePos is to aid in returning a slave axis back to its profile position after an event (e.g., E-stop) caused the slave axis to go off path.
- See Resuming Camming After an E-Stop for instructions.

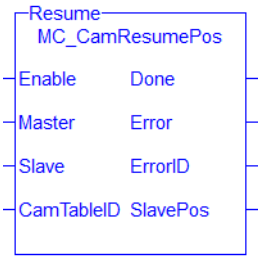
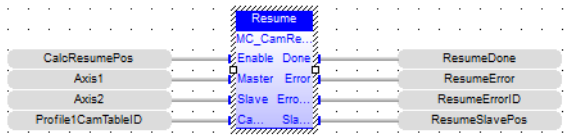
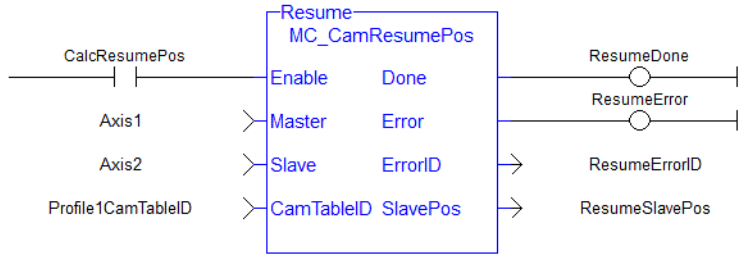


Figure 4-93: MC_CamResumePos

4.2.12.3.4 FBD Language Example



4.2.12.3.5 FFLD Language Example



4.2.12.3.6 IL Language Example

```
            CAL Inst_MC_CamResumePos( TRUE, Axis1, Axis2, Profile1CamTable
ID)
```

4.2.12.3.7 ST Language Example

```
Inst_MC_CamResumePos( TRUE, Axis1, Axis2, Profile1CamTableID);
```

See Also

- "MC_CamIn" (→ p. 476)
- "MC_CamStartPos" (→ p. 488)

4.2.12.4 MC_CamStartPos

PLCopen ✓

Function Block - Returns the slave axis position for starting an "MC_CamIn" (→ p. 476) move.

4.2.12.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	N/A	N/A	No default	Enables execution.
Master	AXIS_REF	AXIS_NUM - 1, 256	N/A	No default	Master axis.
Slave	AXIS_REF	AXIS_NUM - 1, 256	N/A	No default	Slave axis.
MasterOffset	LREAL	No range	User units	No default	Master axis offset. This input is not used if the StartMode input is set to 1 for Resume Mode.
SlaveOffset	LREAL	No range	User units	No default	Slave axis offset. This input is not used if the StartMode input is set to 1 for Resume Mode.
MasterScaling	LREAL	No range	User units	No default	Master axis scale factor. Scaling must be a positive value greater than 0 (zero).
SlaveScaling	LREAL	No range	User units	No default	Slave axis scale factor. Scaling must be a positive value greater than 0 (zero).
CamTableID	INT	0, 255	N/A	No default	Profile ID number. • This value is generated by "MC_CamTblSelect" (→ p. 490).

4.2.12.4.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	TRUE = the function block has successfully calculated the slave position. The slave position is available at the SlavePos output.
Error	BOOL	No range	N/A	TRUE = an invalid input was specified or an error occurred in the calculations. The value at the SlavePos output is undefined.
ErrorID	INT	No range	N/A	Indicates the error if the **Error** output is set to TRUE.
SlavePos	LREAL	No range	User units	If the Done output is TRUE, this output returns the: • position for the slave axis given the profile. • current master axis position.

4.2.12.4.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- This function block returns the slave axis position for the specified profile, based on the current position of the master axis.
  - This slave axis position can be used to command the slave axis to move to the proper location prior to commanding a "MC_CamIn" (→ p. 476) move with StartMode = 0 (Start mode).
- The typical application of MC_CamStartPos is to aid in positioning a slave axis to its starting position for a MC_CamIn move with a slave absolute profile.
- See Positining an Axis Before Starting Camming for instructions.

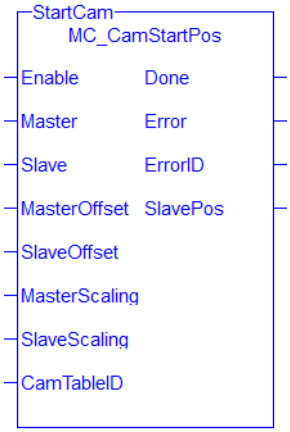
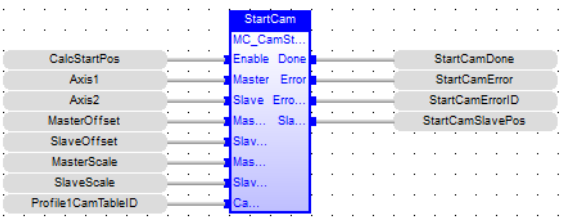
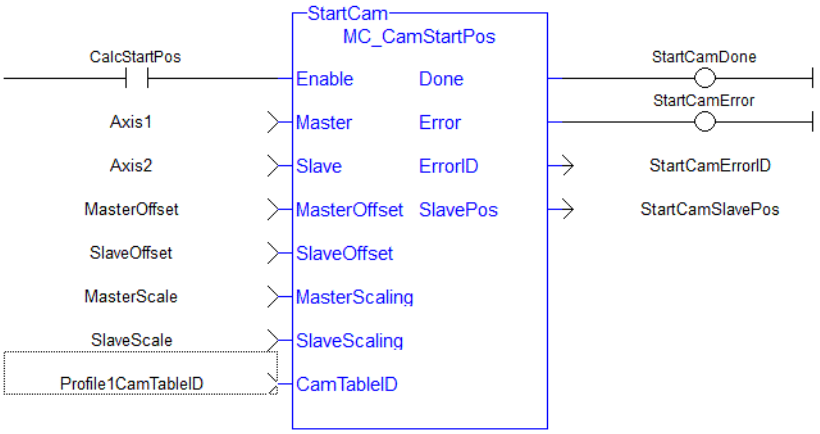


Figure 4-94: MC_CamStartPos

4.2.12.4.4 FBD Language Example



4.2.12.4.5 FFLD Language Example



4.2.12.4.6 IL Language Example

```
            CAL Inst_MC_CamStartPos( TRUE, Axis1, Axis2, MasterOffset,
SlaveOffset, MasterScale, SlaveScale, Profile1CamTable ID)
```

4.2.12.4.7 ST Language Example

```
Inst_MC_CamStartPos( TRUE, Axis1, Axis2, MasterOffset, SlaveOffset,
MasterScale, SlaveScale, Profile1CamTableID);
```

4.2.12.5 MC_CamTblSelect

PLCopen

Function Block - Defined to read and initialize the specified profile.

4.2.12.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	Requests to queue the slave gear ratio move.
CamTable	STRING	No range	N/A	No default	Profile name as defined in the CAM Profile Properties dialog.
Periodic	BOOL	0, 1	N/A	No default	Selects if the profile is periodic. See the Usage for more information.
MasterAbsolute	BOOL	0, 1	N/A	No default	Selects if master profile is absolute or relative. See the Usage for more information.
SlaveAbsolute	BOOL	0, 1	N/A	No default	Selects if slave profile is absolute or relative. See the Usage for more information.

4.2.12.5.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	0, 1	N/A	Indicates whether this function block has completed **without** error.
Busy	BOOL	0, 1	N/A	Indicates this function block is executing.
Error	BOOL	0, 1	N/A	Indicates an invalid input was specified.
ErrorID	INT	No range	N/A	Indicates the error if the Error output is high.
CamTableID	INT	0, 255	N/A	Indicates the ID number of the profile to be used with MC_CamIn.

4.2.12.5.3 Remarks

- Returns an ID to be used with "MC_CamIn" (→ p. 476).
- See Main for information about how this function block is used in the Hole punch project.

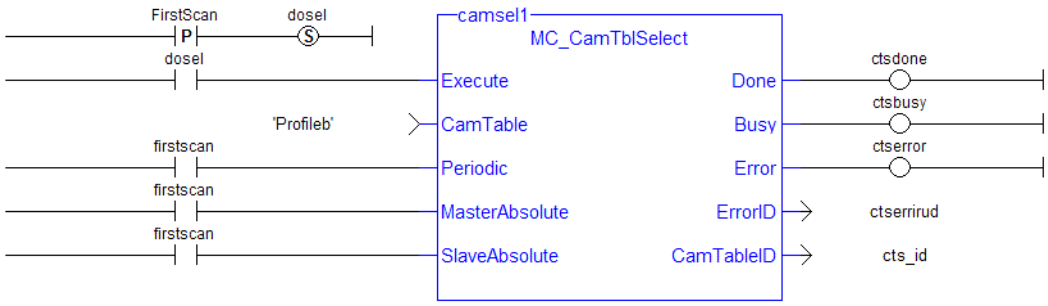
4.2.12.5.3.1 Usage

- Each positive transition of the **Enable** input creates a unique Cam ID and store the profile information in a table.
  - The number of unique Cam IDs is limited to 256.
  - If the application attempts to create more than 256 Cam IDs:
    - The **Error** output is TRUE.
    - The **ErrorID** output is `22` (`Too Many Profiles`).
  - It is only necessary to call MC_CamTblSelect once for each Profile/Periodic/MasterAbsolute/SlaveAbsolute configuration to be used.
- The **Periodic** input selects if the profile is to repeat each cycle.
  - If the profile is not periodic and the master axis moves beyond the profile range, the slave stops at the end of the profile.
- If the master axis moves back into the profile range, the slave resumes following the profile.
- If the **MasterAbsolute** input is ON, the profile is in reference to the Master axis position.
  - If the MasterAbsolute input is OFF, the profile is in reference to the Master axis position at the time the MC_CamIn function block is executed.
- The **SlaveAbsolute** input selects if the slave positions are in reference to the Slave axis position or the Slave axis position at the time the MC_CamIn function block is executed.
  - If the **SlaveAbsolute** input is set to TRUE, the axis jumps back to the starting position.
  - If the **SlaveAbsolute** input is set to FALSE, the axis no longer jumps back.
    - Instead, as the profile repeats, the slave moves relative to the start of each period.

4.2.12.5.4 FBD Language Example

Not available.

4.2.12.5.5 FFLD Language Example



4.2.12.5.6 IL Language Example

Not available.

4.2.12.5.7 ST Language Example

```
(* MC_CamTblSelect ST example *) //call this function block every scan until
"Done"
Inst_MC_CamTblSelect(DoSelect, 'Profileb', TRUE, TRUE, TRUE ); //Inst_MC_
CamTblSelect is instance of MC_CamTblSelect
CamSelDone := Inst_MC_CamTblSelect.Done;  //store Done output to user defined
variable
IF CamSelDone = TRUE THEN//when function block is "done" store
CamTableID := Inst_MC_CamTblSelect.CamTableID;  //CamTableID in user defined
variable
END_IF;
```

See Also

- Call a Function Block
- Differences Between Functions and Function Blocks
- "MC_CamIn" (→ p. 476)
- "MC_CamOut" (→ p. 484)

4.2.12.6 MC_GearIn

PLCopen

Function Block - Performs a slave axis move which follows the master axis based on the ratio specified by RatioNumerator and RatioDenominator.

4.2.12.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to queue the slave gear ratio move.
Master	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
Slave	AXIS_REF	1, 256	N/A	No default	AXIS_REF.AXIS_NUM is the slave axis number.
RatioNumerator	DINT	-2147483648 to 2147483647	N/A	No default	Numerator of master/slave ratio.
RatioDenominator	DINT	-2147483648 to 2147483647	N/A	No default	Denominator of master/slave ratio.
Acceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Acceleration rate. S-curve: Maximum acceleration.
Deceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Deceleration rate. S-curve: Unused.

Input	Data Type	Range	Unit	Default	Description
Jerk	LREAL	No range	User unit/sec^3	No default	Trapezoidal: 0 (zero). S-curve: Constant jerk.
BufferMode	SINT	0, 1	N/A	No default	• 0 = Abort. • 1 = Buffer.

4.2.12.6.2 Outputs

Output	Data Type	Range	Unit	Description
InGear	BOOL	No range	N/A	Indicated the slave axis is locked on to the master axis.
Busy	BOOL	No range	N/A	High from the moment the Execute input goes high until the time the move is ended.
Active	BOOL	No range	N/A	Indicates this move is the Active move.
CommandAborted	BOOL	No range	N/A	Indicates the move was aborted.
Error	BOOL	No range	N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
ErrorID	INT	No range	N/A	Indicates the error if the **Error** output is set to TRUE.

4.2.12.6.3 Remarks

```
SlaveCommandPosition = MasterActualPosition * RatioNumerator /
RatioDenominator
```

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- When this command is executed, the slave axis accelerates or decelerates (using the Acceleration, Deceleration, and Jerk) to the target velocity determined by the master axis velocity and the ratio.
- When the slave axis reaches a velocity within the In Gear bandwidth around the target velocity, it locks on to the master, and the InGear output goes high.
  - See Axis Parameters for more information.

- When the slave is locked to the master, the slave motion is no longer affected by the acceleration, deceleration, and jerk inputs.
  - Example: If the In Gear bandwidth is set to 0.1 User unit/sec, the InGear output turns on if the slave velocity is within +/- 0.1 User unit/sec of the target velocity.
  - The slave axis continues to follow the master axis until this move is aborted.

4.2.12.6.3.1 Aborting Gearing

Gearing functions can generate large accelerations while following the master.

- If the aborting function block has small, non-zero Jerk, or small acceleration values, it can take a long time for an accelerating axis to reach the target velocity, or position of the aborting function block.
  - If the Jerk and/or acceleration of the aborting function cannot be increased to suitable values, it may be desirable to either:
    - Abort the gearing function with an "MC_GearOut" (→ p. 503) with higher accelerations and/or Jerk values (or zero jerk value).
    - Execute the next MC motion function block (e.g., "MC_Halt" (→ p. 449)).

4.2.12.6.3.2 Time to Reach the Target Velocity

While following the master, gearing functions can generate large accelerations.

- If the gearing function is aborted while the axis is currently accelerating, and the aborting function block has small non-zero Jerk or small acceleration values, it can take a long time to reach the target velocity, or position of the aborting function block.
  - If the Jerk and/or acceleration of the aborting function cannot be increased to suitable values, it may be desirable to either:
    - Abort the gearing function with an "MC_GearOut" (→ p. 503) with higher accelerations and/or Jerk values (or zero jerk value).
    - Execute the next MC motion function block (e.g., "MC_Halt" (→ p. 449)).

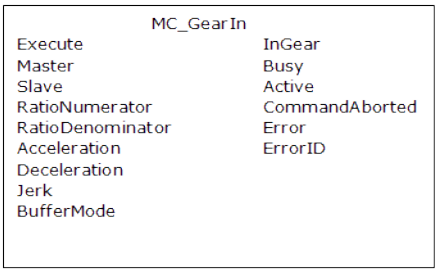
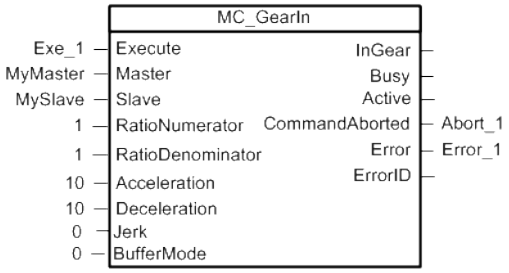


Figure 4-95: MC_GearIn

4.2.12.6.3.3 Time Diagram



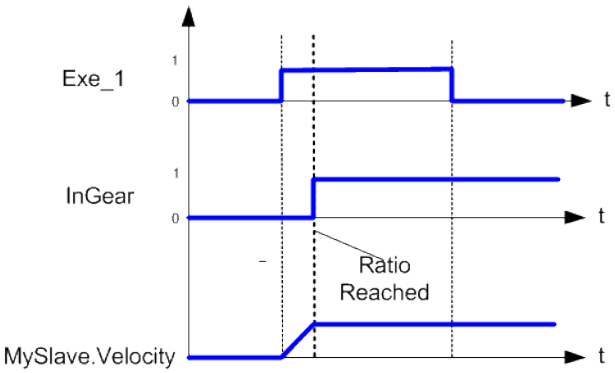
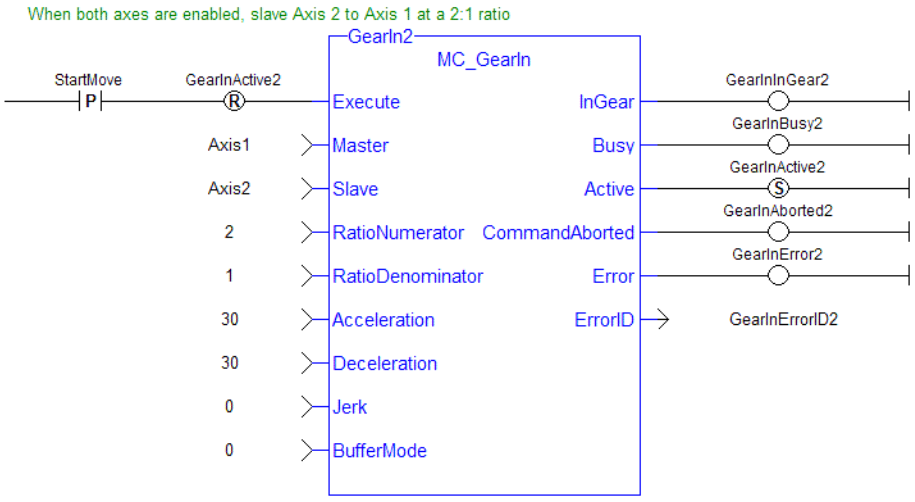


Figure 4-96: Time Diagram

4.2.12.6.4 FBD Language Example

Not available.

4.2.12.6.5 FFLD Language Example



4.2.12.6.6 IL Language Example

Not available.

4.2.12.6.7 ST Language Example

```
(* MC_GearIn ST example *)
   Inst_MC_GearIn( GearInReq, Axis1, Axis2, 2, 1, 150.0, 150.0, 0, 0 );
//Inst_MC_GearIn is an instance of MC_GearIn
```

4.2.12.7 MC_GearInPos

PLCopen

Function Block - Performs a slave axis move which follows the master axis based on the ratio specified by RatioNumerator and RatioDenominator.

4.2.12.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to queue the slave gear ratio move.
Master	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
Slave	AXIS_REF	1, 256	N/A	No default	AXIS_REF.AXIS_NUM is the slave axis number.
RatioNumerator	DINT	-2147483648 to 2147483647	N/A	No default	Numerator of master/slave ratio. The + or - sign indicates the direction for the slave axis.

Input	Data Type	Range	Unit	Default	Description

RatioDenominator	DINT	-2147483648 to 2147483647	N/A	No default	Denominator of master/slave ratio.
					• The + or - sign indicates the direction for the master axis.
					• The direction determines the sync trigger comparison direction for the slave.
					• For a master moving in the:
					  • positive direction, use a positive RatioDenominator.
					  • negative direction, use a negative RatioDenominator.

MasterSyncPosition	LREAL	-1.7E308 to 1.7E308	N/A	No default	Master axis sync position. 14 to 15 significant digits of accuracy.
SlaveSyncPosition	LREAL	-1.7E308 to 1.7E308	N/A	No default	Slave axis sync position. 14 to 15 significant digits of accuracy.

Input	Data Type	Range	Unit	Default	Description

SyncMode	INT	0, 1	N/A	No default	SyncMode determines the allowed conditions for synchronization.
					0 = Normal Synchronization
					• Prior to executing the MC_GearInPos function block, the Master axis position must be before the MasterSyncPosition by a distance greater than the MasterStartDistance.
					  • The Slave axis position must be before the SlaveSyncPosition.
					• Axes that have a non-zero rollover, the MC_GearInPos function block always assumes the axes meet these conditions by assuming the sync point is in the next occurrence of the sync position.
					  • MasterStartDistance must be positive and greater than the distance the master axis is currently moving per axis update.
					  • If the master start distance and the slave axis distance from the SlaveSyncPosition are sufficiently large enough, the slave axis ramps to the sync position.
					  • If not sufficiently large enough, acceleration of the slave axis may be excessive.
					1= Immediate Synchronization Allowed
					• This mode is only allowed if both the

Input	Data Type	Range	Unit	Default	Description

					master and slave axes have rollover = 0 (zero).
					• If the conditions of SyncMode = 0 (zero) are not met, Synchronization is allowed even though the axis positions may be beyond their respective Sync Positions.
					  • The MasterStartDistance may be 0 (zero).
					  • If the MasterStartDistance is 0 (zero), the Slave axis synchronizes with the master the instant the master axis crosses the MasterSyncPosition.
					• If either the master or slave axis are beyond their respective sync start positions, the slave axis immediately synchronizes to the master axis.
					• If the master start distance and the slave axis distance from the SlaveSyncPosition are sufficiently large enough, the slave axis ramps to the sync position.
					• If not sufficiently large enough or immediate synchronization occurs, slave axis acceleration may be excessive.
					  • Excessive slave acceleration may occur if the master axis velocity is large or the master and slave axes have disproportionally

Input	Data Type	Range	Unit	Default	Description
					different distances to their respective sync positions. • If the slave axis is ahead of the master axis at the time of synchronization, the slave axis moves backwards.
MasterStartDistance	LREAL	-1.7E308 to 1.7E308	User units	No default	When the master axis reaches this distance before MasterSyncPosition, the slave axis begins its lock-on process. • 14 to 15 significant digits of accuracy. • The **MasterStartDistance** * (**RatioNumerator** /**RatioDenominator**) should be greater than (or equal to) the slave sync distance. • The slave sync distance is defined as the distance between the slave position when MC_GearInPos executes and the **SlaveSyncPosition**. • If the MasterStartDistance is too short, the MC_GearInPos may have excessive acceleration and a warning log message is generated.
BufferMode	SINT	1	N/A	No default	• 1 = Buffer. • See Buffer Modes for more information.

4.2.12.7.2 Outputs

Output	Data Type	Range	Unit	Description
StartSync	BOOL	No range	N/A	Indicates the master axis has reached the MasterStartDistance from the MasterSyncPosition and the lock-on process has begun.

Output	Data Type	Range	Unit	Description
InSync	BOOL	No range	N/A	Indicated the slave axis is locked on to the master axis.
Busy	BOOL	No range	N/A	High from the moment the Execute input goes high until the time the move is ended.
Active	BOOL	No range	N/A	Indicates this move is the Active move.
CommandAborted	BOOL	No range	N/A	Indicates the move was aborted. If the abort arises because the inputs cause inconsistent motion, then this FB: • Performs no motion. • Sets an error flag. • Set the ErrorID to 13.
Error	BOOL	No range	N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
ErrorID	INT	No range	N/A	Indicates the error if the **Error** output is set to TRUE.

4.2.12.7.3 Remarks

```
SlaveCommandPosition = MasterActualPosition * RatioNumerator /
RatioDenominator
```

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- This function block allows the application to specify sync positions for the master and slave axes.
  - It is the point in which the master and slave axes become engaged in synchronous motion.
- When the master axis reaches the MasterStartDistance from the MasterSyncPosition, the slave axis begins to accelerate to the target velocity determined by the master axis velocity and the ratio.
  - The slave axis arrives at the target velocity and the SlaveSyncPosition at the same time the master axis arrives at the MasterSyncPosition.
  - At that time, the slave is locked on to the master and follows the master at the ratio specified.
  - The slave axis continues to follow the master axis until this move is aborted.

4.2.12.7.3.1 Aborting Gearing

Gearing functions can generate large accelerations while following the master.

- If the aborting function block has small, non-zero Jerk, or small acceleration values, it can take a long time for an accelerating axis to reach the target velocity, or position of the aborting function block.
  - If the Jerk and/or acceleration of the aborting function cannot be increased to suitable values, it may be desirable to either:
    - Abort the gearing function with an "MC_GearOut" (→ p. 503) with higher accelerations and/or Jerk values (or zero jerk value).
    - Execute the next MC motion function block (e.g., "MC_Halt" (→ p. 449)).

4.2.12.7.3.2 Time to Reach the Target Velocity

While following the master, gearing functions can generate large accelerations.

- If the gearing function is aborted while the axis is currently accelerating, and the aborting function block has small non-zero Jerk or small acceleration values, it can take a long time to reach the target velocity, or position of the aborting function block.

- If the Jerk and/or acceleration of the aborting function cannot be increased to suitable values, it may be desirable to either:
  - Abort the gearing function with an "MC_GearOut" (→ p. 503) with higher accelerations and/or Jerk values (or zero jerk value).
  - Execute the next MC motion function block (e.g., "MC_Halt" (→ p. 449)).

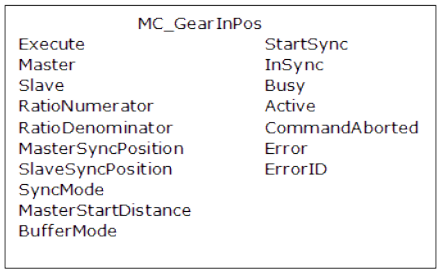


Figure 4-97: MC_GearInPos

4.2.12.7.3.3 Time Diagram

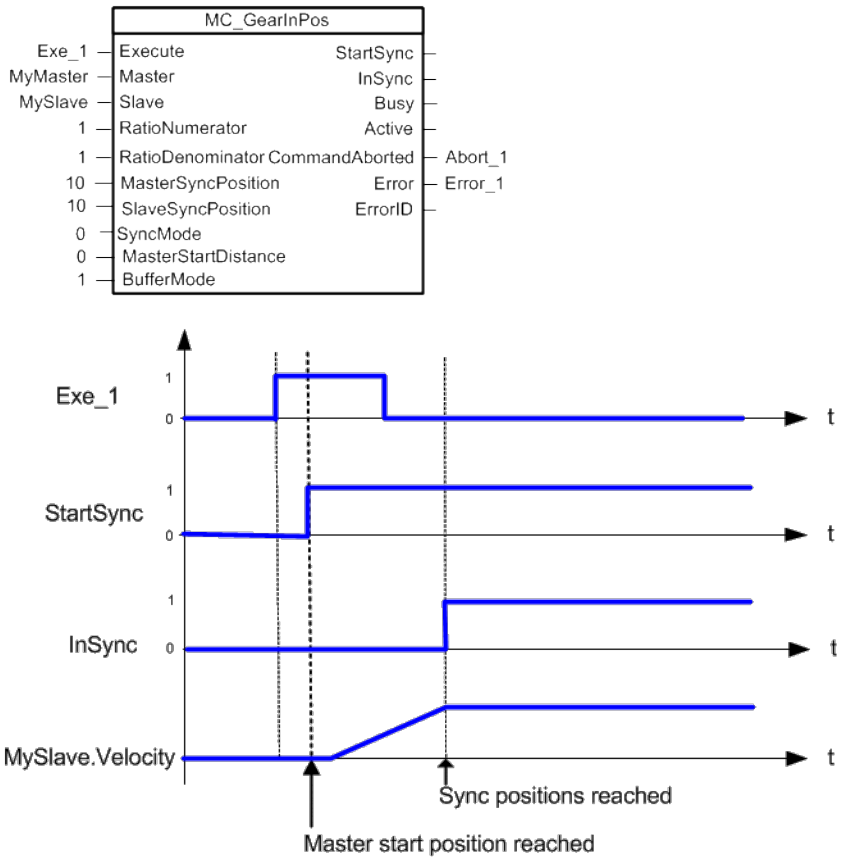
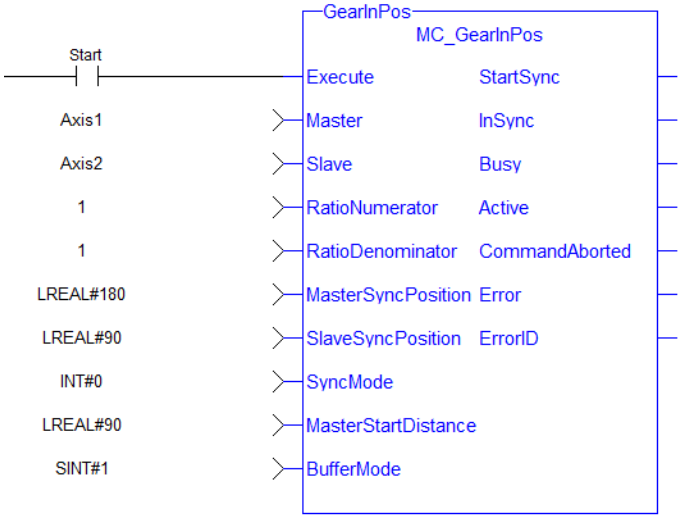
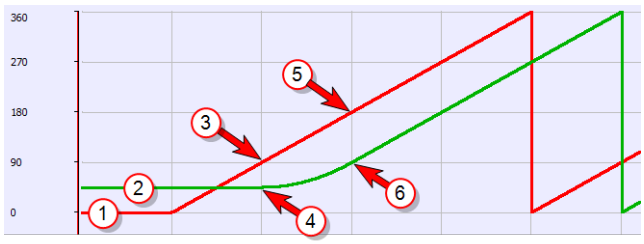


Figure 4-98: Time Diagram

4.2.12.7.3.4 Example

- Master and Slave are rotary axes with rollovers at 360 degrees.
- The Master initial position is 0 degrees and the slave initial position is 45 degrees.
- The GearInPos FB commands the slave to accelerate up to the geared ratio (1:1) during the master start distance (90 degrees) and be synchronized with the master at the master and slave sync positions.



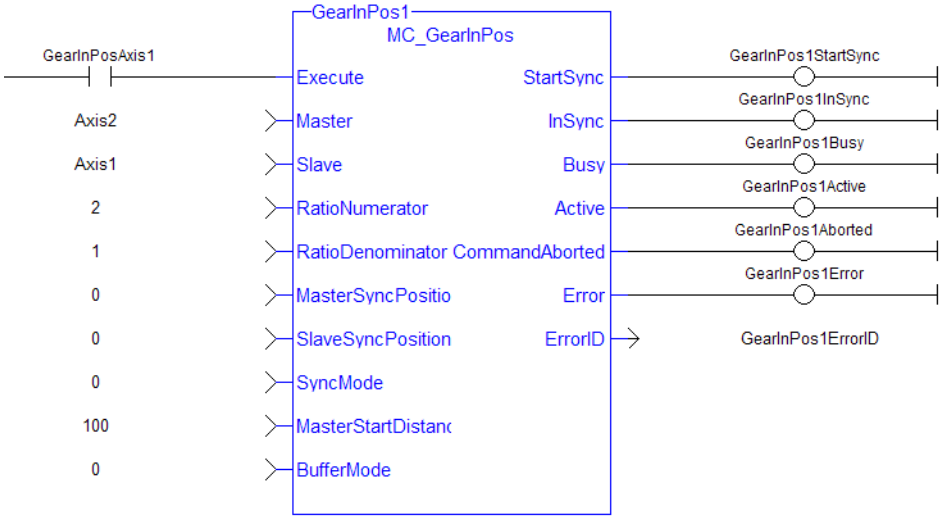


1. Master Axis
2. Slave Axis
3. Master Start Distance
4. Slave starts to accelerate
5. Master Sync Position
6. Slave Sync Position

4.2.12.7.4 FBD Language Example

Not available.

4.2.12.7.5 FFLD Language Example



4.2.12.7.6 IL Language Example

Not available.

4.2.12.7.7 ST Language Example

```
            (* MC_GearInPos ST example *)

Inst_MC_GearInPos( GearInPosReq, Axis1, Axis2, 2, 1, 0, 0, 0, 100.0, 1 );
//Inst_MC_GearInPos is instance of MC_GearInPos
GearInPosSync:= Inst_MC_GearInPos.InSync;
//store InSync output into user defined variable
```

4.2.12.8 MC_GearOut

PLCopen ✓

Function Block - Disengages the slave axis from a MC_GearIn or MC_GearInPos move.

4.2.12.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to disengage the slave axis from a "MC_GearIn" (→ p. 492) or "MC_GearInPos" (→ p. 495) move.
Slave	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
Acceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Acceleration rate. S-curve: Maximum acceleration.
Deceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Deceleration rate. S-curve: Unused.
Jerk	LREAL	No range	User unit/sec^3	No default	Trapezoidal: 0 (zero). S-curve: Constant jerk.

4.2.12.8.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates the axis is disengaged from its master.
Busy	BOOL	No range	N/A	Indicates the function is executing.
Active	BOOL	No range	N/A	Indicates this move is the Active move.
CommandAborted	BOOL	No range	N/A	Indicates the move was aborted.
Error	BOOL	No range	N/A	Indicates either: • An invalid input was specified. • No MC_GearIn or MC_GearInPos move is active.
ErrorID	INT	No range	N/A	Indicates the error if the **Error** output is set to TRUE.

4.2.12.8.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

This function block:

- Aborts the active MC_GearIn or MC_GearInPos move.
- Disengages the axis from its master.
- Commands the axis to continue at its current velocity.

TIP

The current velocity is calculated by taking the average of the actual velocity during the previous 16 cycles.

- The control continues to command the axis to move at this velocity until this MC_GearOut move is aborted.
- The Acceleration, Deceleration, and Jerk input parameters are applied if this command velocity is modified by the MC_SetOverride function block.
- If this function block is called, and the active move is not a MC_GearIn or MC_GearInPos move, this function block returns an error and the active move is not aborted.

NOTE

The MC_GearOut is done when the slave axis is disengaged from the master axis.
Once done, the MC_GearOut remains busy and active until it is aborted by a different motion function block.
This is different behavior than most other motion function blocks.
The MC_GearOut function block represents an exception to the exclusivity rule as the **Done** and **Active** outputs may be true at the same time.

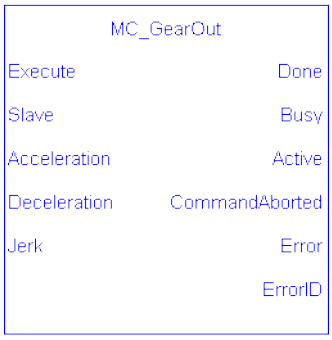
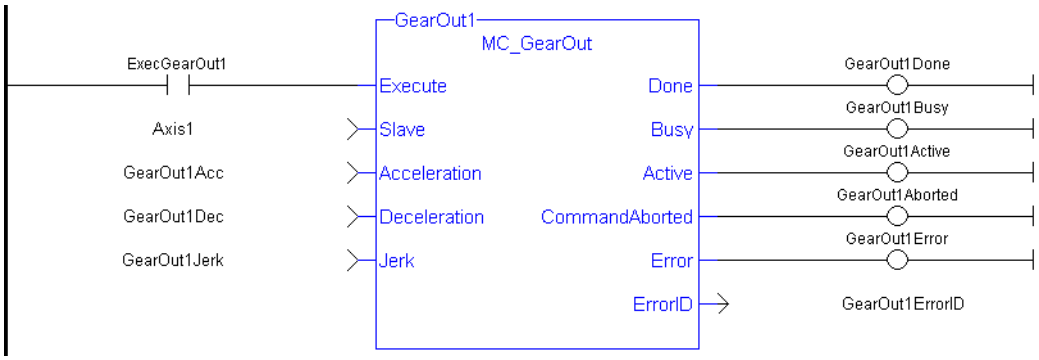


Figure 4-99: MC_GearOut

4.2.12.8.4 FBD Language Example

Not available.

4.2.12.8.5 FFLD Language Example



4.2.12.8.6 IL Language Example

Not available.

4.2.12.8.7 ST Language Example

```
            (* MC_GearOut ST example *)

Inst_MC_GearOut(ExecGearOut1,Axis1,GearOut1Acc,GearOut1Dec,GearOut1Jerk);
//Inst_MC_GearOut is instance of MC_GearOut
```

See Also

- Call a Function Block
- Differences Between Functions and Function Blocks

4.2.12.9 MC_Phasing

PLCopen ✓

Function Block - Performs a master position phase shift for the slave axis.

4.2.12.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to queue the phase shift.
Master	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
Slave	AXIS_REF	1, 256	N/A	No default	AXIS_REF.AXIS_NUM is the slave axis number.
PhaseShift	LREAL	No range	User units	No default	Amount of phase shift.
Velocity	LREAL	No range	User unit/sec	No default	Velocity setpoint.
Acceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Acceleration rate. S-curve: Maximum acceleration.
Deceleration	LREAL	No range	User unit/sec^2	No default	Trapezoidal: Deceleration rate. S-curve: Unused.
Jerk	LREAL	No range	User unit/sec^3	No default	Trapezoidal: 0 (zero). S-curve: Constant jerk.
BufferMode	SINT	0, 5	N/A	No default	• 0 = Abort. • 1 = Buffer. • 2 = Blend to active. • 3 = Blend to next. • 4 = Blend to low velocity. • 5 = Blend to high velocity.

4.2.12.9.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates the phase shift has been completely applied.
Busy	BOOL	No range	N/A	High from the moment the Execute input is one-shot to the time the move is ended.
Active	BOOL	No range	N/A	Indicates this phase shift is the active phase shift.
CommandAborted	BOOL	No range	N/A	Indicates the move was aborted.

Output	Data Type	Range	Unit	Description
Error	BOOL	No range	N/A	Indicates either: • An invalid input was specified. • The move was terminated due to an error.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE.

4.2.12.9.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

This function block:

- Provides a way to smoothly apply a master offset instead of writing values directly to the Master Offset Parameter 1002.
- Is commonly used with "MC_TouchProbe" (→ p. 428) for performing position corrections on the slave axis in a **Mark-to-Mark** registration application.

TIP

MC_Phasing performs a similar function to adjusting the MasterOffset input in the "MC_CamIn" (→ p. 476) function block.
It has the additional features of setting the velocity, acceleration, deceleration, and jerk motion parameters.

- The distance entered at the **PhaseShift** input is iterated into the Slave axis's Master Offset.
  - This distance is iterated like a "MC_MoveRelative" (→ p. 464) move using the specified **Velocity**, **Acceleration**, **Deceleration**, and **Jerk** values.
- The difference is the interpolated command delta is not commanded to the axis but is added to the Slave axis's Master Offset.
  - This shifts the Master axis's position as viewed by the Slave axis, causing a change in the Slave axis's physical position.
  - This only affects the Slave axis if it is executing a slave move.
- Subsequent calls to MC_Phasing can abort or blend to an executing MC_Phasing command.

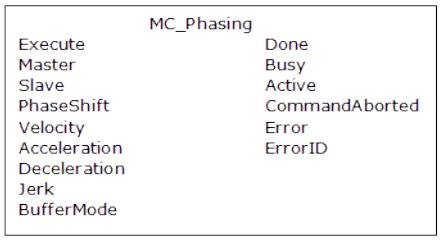
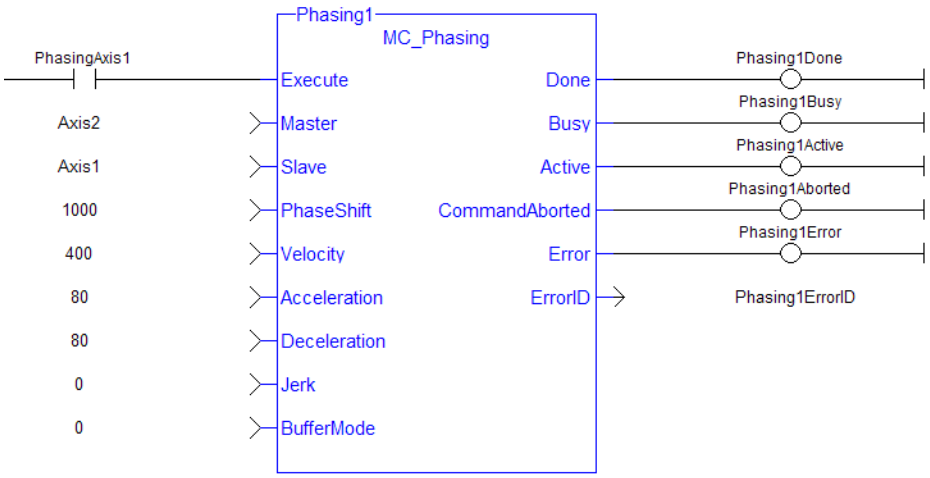


Figure 4-100: MC_Phasing

4.2.12.9.4 FBD Language Example

Not available.

4.2.12.9.5 FFLD Language Example



4.2.12.9.6 IL Language Example

Not available.

4.2.12.9.7 ST Language Example

```
(* MC_Phasing ST example *) //Inst_MC_Phasing is an instance of MC_Phasing
function block
Inst_MC_Phasing(PhasingAxis1, Axis2, Axis1, 1000.0,100.0, 200.0, 200.0, 0, 0
);
```

4.2.12.10 MC_SyncSlaves

PLCopen

Function Block - Allows the application to specify what slave axes are to be synchronized and which master they follow.

4.2.12.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to queue the synchronization of a list of slave axes.
Master	AXIS_REF	1, 256	N/A	No default	Master axis identifier.
SlaveCount	AXIS_REF	1, 256	N/A	No default	The number of slave axes listed in the SlaveList array input that are to be synchronized. • This number must not be greater than the declared size of the SlaveList array. • If this number is 0 (zero), the list of synchronized slaves for the specified Master axis is cleared. • The AXIS_NUM element of the AXIS_REF structure must be in the range.
SlaveList	UINT	1, 32	N/A	No default	The list of slave axes that are to be synchronized. • Each element of this array contains a unique axis number. • The axis number must not be the same as the Master axis number.

4.2.12.10.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			Indicates the synchronized slave assignments were completed without error.
Error	BOOL			Indicates an invalid input was specified.
ErrorID	INT			Indicates the error if the **Error** output is set to TRUE.

4.2.12.10.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- After this function block is executed successfully, all the slave axes specified at the **SlaveList** input start their slave moves (i.e., **MC_CamIn**, **MC_GearIn**, etc.) on the same servo interrupt for a synchronized slave start.
- When a slave move is commanded for one of the slave axes listed, the slave move is queued but the motion is held off until all of the listed slaves have queued their slave moves.

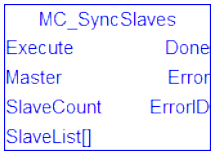


Figure 4-101: MC_SyncSlaves

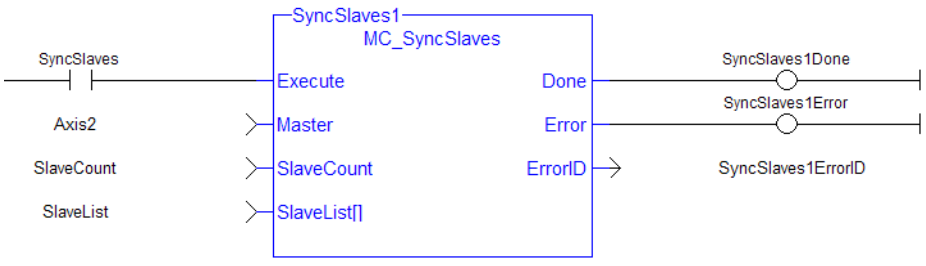
4.2.12.10.3.1 Usage

- Call MC_SyncSlaves to specify the slave axes to synchronize.
- Call each slave move (e.g., **MC_GearIn**) for each slave axis.
  - The motion is held off until all the slave moves have been queued.
- After all the slave moves have been queued, the interpolation for all the slave axes begin on the same servo interrupt, providing a synchronized start.
- The master axis can be in motion prior to this sequence or the master can be commanded after all the slave moves are queued.

4.2.12.10.4 FBD Language Example

Not available.

4.2.12.10.5 FFLD Language Example



4.2.12.10.6 IL Language Example

Not available.

4.2.12.10.7 ST Language Example

```
(* MC_SyncSlaves ST example *)
// Inst_MC_SyncSlaves is an instance of MC_SyncSlaves function block
Inst_MC_SyncSlaves( SyncSlaves, Axis1, SlaveCount, SlaveList );
```

See Also

- "MC_CamIn" (→ p. 476)
- "MC_GearIn" (→ p. 492)
- "MC_GearInPos" (→ p. 495)

4.2.13 Reference

These functions provide commands for reference points.

Name	Description
MC_Reference	Used to execute a fast home to a switch.
MC_SetPos	Changes the present actual position of the axis (as reported by "MC_ReadActPos" (→ p. 436)) to the position specified by the **Position** and **Mode** inputs.
MC_SetPosition	Deprecated - See "MC_SetPos" (→ p. 514).

4.2.13.1 MC_Reference

PLCopen

Function Block - Used to execute a fast home to a switch.

4.2.13.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to queue the move and arms reference trigger events.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.

Input	Data Type	Range	Unit	Default	Description
TriggerInput	TRIGGER_REF	See Description.	N/A	No default	TRIGGER_REF structure defines the trigger. **Direction**INT Range = 1, 2. • 1 = Rising edge. • 2 = Falling edge. **InputID** INT Range = 0, 1 • 0 = Touch Probe 1 / Capture Engine 0 • 1 = Touch Probe 2 / Capture Engine 1 **Trigid**INT • Must be 0 (zero). **NOTE** **TrigMode** INT (TriggerInput.TrigMode) is not presently supported by this function.
Velocity	LREAL	No range	User unit/sec	No default	Commanded velocity for the reference move.
Acceleration	LREAL	no r	User unit/sec 2	No default	Commanded acceleration for the reference move.
Deceleration	LREAL	no r	User unit/sec 2	No default	Commanded deceleration for the reference move.
Jerk	LREAL	No range	User unit/sec 3	No default	Commanded jerk for the reference move. If 0 (zero), the trapezoidal acc/dec is used.
Direction	SINT	0, 1	N/A	No default	Commanded direction of the reference.
Position	LREAL	No range	User units	No default	The position the axis is reset to when at the machine reference location.

Input	Data Type	Range	Unit	Default	Description
Option	SINT	0, 15	N/A	No default	Option identifier for Resolvers/Modulo reference. • 0 = Use latched position for reference. • 1 = Use resolver position of nearest null for reference 2 pole resolver. • 2 = Use resolver position of nearest null for reference 4 pole resolver. • 3 = Use resolver position of nearest null for reference 6 pole resolver. • 4 = Use resolver position of nearest null for reference 8 pole resolver. • 5 = Use resolver position of nearest null for reference 10 pole resolver. • ... • 15 = Use resolver position of nearest null for reference 30 pole resolver.

4.2.13.1.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			Indicates the reference move and position adjustment is complete.
Busy	BOOL			Indicates this function block is executing.
Active	BOOL			Indicates this move is the Active move.
CommandAborted	BOOL			Indicates the move was aborted.
Error	BOOL			Indicates either: • An invalid input was specified. • The move was terminated due to an error.
ErrorID	INT			Indicates the error if **Error** output is high.

4.2.13.1.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

NOTE

At this time, position capture is not available for PLCopen axes assigned to the secondary feedback input (digitizing axes).
MC_Reference cannot be used to home digitizing axes.

- If the application selects to reference to the index mark of an encoder, or the null of a resolver (which is typical), the new position value is assigned to the position of the index of the encoder (or the null of the resolver) and not the position of the switch.
- The **ECATWriteSDO** function block is used to setup the trigger event and any desired preconditions.
- This function block utilizes the Position Capture Mode of the AKD.

TIP

There are a differences between how an AKD and an AKD2G are used with this function block.
- "AKD Support With MC_TouchProbe" (→ p. 432)
- "AKD2G Support With MC_TouchProbe" (→ p. 433)

4.2.13.1.3.1 Usage

TIP

Once the **MC_Reference** block is queued, but before it is completed, the cycle can be aborted by either:

- Using a "MC_Halt" (→ p. 449) or "MC_Stop" (→ p. 423) function block.
- By queuing a new motion function block with the Abort selected for buffer mode.

These are steps for homing a PLCopen axis using the MC_Reference function block.

Not all of the steps are necessary depending on the configuration and the homing cycle design.

1. Verify the Axis is not on Reference switch.
   - If a switch is used in the homing cycle for the event or precondition to the event, verify the axis is not already tripping the switches that trigger the event and precondition.
   - If it is, move the axis off the switches.
2. Configure the AKD capture engine.
   - Configuration of the AKD capture engine is performed by writing drive CAN objects via SDO.
   - Use the **ECATWriteSdo** function for this action.
     - The AKD Capture mode must be set to **POSITION CAPTURE**.
   - See AKD Capture Engine Configuration for information about the available configurations.
   - See PLCopen Homing Methods for example AKD capture engine configurations and references.
3. Call the **MC_Reference** function to initiate optional homing motion and to arm the AKD capture engine.
   - **MC_Reference** selects the trigger edge (rising or falling edge) and arms the capture.
   - Then, it optionally moves the axis to the reference location as directed by inputs to this function.
   - When the AKD indicates the capture event has occurred, the coordinate system is shifted so the reference position input to this function block is set to the reference location and the reference motion is stopped.
4. Wait for the completion of the **MC_Reference** function block.
   - The application is notified by the completion, abort, or error of the homing by the **MC_Reference** function block.
5. When the **MC_Reference** function block is completed, the axis can be moved to the home position with a **MC_MoveAbsolute** function block.

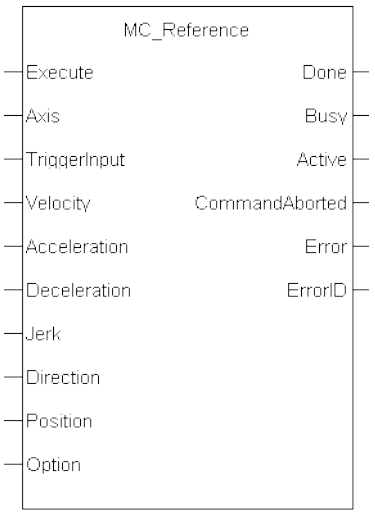
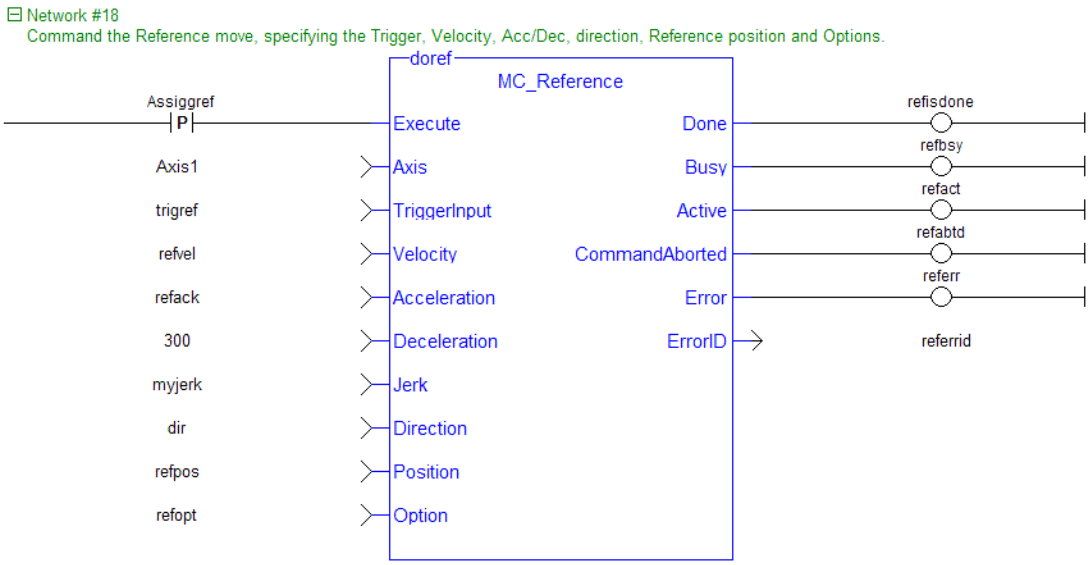


Figure 4-102: MC_Reference

4.2.13.1.4 FBD Language Example

Not available.

4.2.13.1.5 FFLD Language Example



4.2.13.1.6 IL Language Example

Not available.

4.2.13.1.6.1 ST Language Example

```
(* MC_Reference ST example *)
TriggerInput.InputID := 0;    //configure the reference InputID
TriggerInput.DIRECTION := 1;  //configure the reference direction
Inst_MC_Reference( RefReq, Axis1, TriggerInput, 20.0, 100.0, 100.0, 100.0, 0,
0.0, 0 );
```

See Also

- "ECATWriteSDO" (→ p. 61)
- "MC_MoveAbsolute" (→ p. 451)

4.2.13.2 MC_SetPos

PLCopen

Function Block - Changes the present actual position of the axis (as reported by "MC_ReadActPos" (→ p. 436)) to the position specified by the **Position** and **Mode** inputs.

4.2.13.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to queue the change to the axis position.
Axis	AXIS_REF	1, 256	N/A	No default	Name of a declared instance of the AXIS_REF library function.
Position	LREAL	No range	N/A	No default	**Absolute Mode**: New Axis Position to replace the present position. **Relative Mode**: Position offset to apply to present position. Typically used with multi-turn absolute position feedback devices.
Mode	BOOL	No range	N/A	No default	LOW = Position input is an absolute position. HIGH = Position input is a relative position

4.2.13.2.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			Indicates the reference move and position adjustment is complete.
Busy	BOOL			Indicates this function block is executing.
Error	BOOL			Indicates either: • An invalid input was specified. • The move was terminated due to an error.
ErrorID	INT			Indicates the error if **Error** output is high.

4.2.13.2.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

-
- If a motor is associated with the axis, it does not move when MC_SetPos is executed.
  - MC_SetPos does not cause any motion.
  - It applies an offset to the command and actual positions.
- MC_SetPos sets the accumulated Superimposed distance value for the input axis to 0 (zero).
  - See Axis Positions Data for more information.
- This function block replaces **MC_SetPosition**.

NOTE

This function block starts a motion-related action and stores data for calculations and error checking.
See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

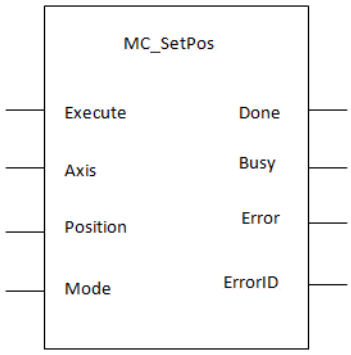
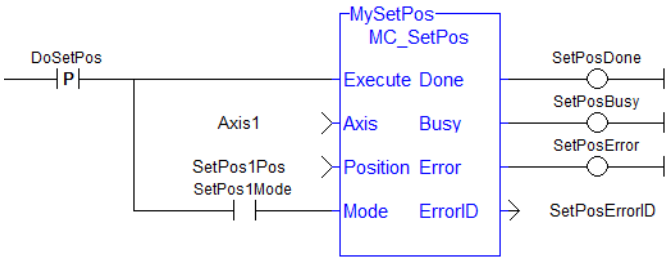


Figure 4-103: MC_SetPos

4.2.13.2.4 FBD Language Example

Not available.

4.2.13.2.5 FFLD Language Example

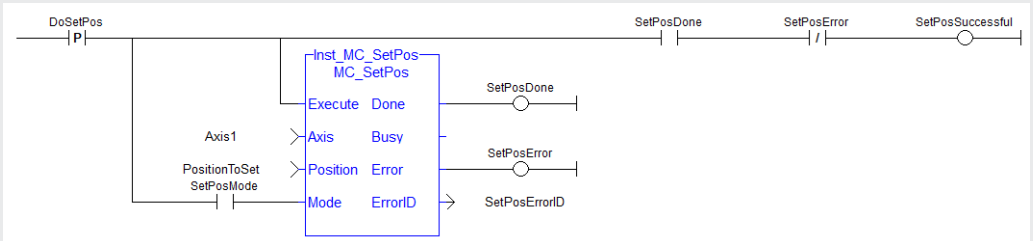


TIP

This function block finishes immediately.
Because of this, the **Done** output does not get set to FALSE in a second call to the same MC_SetPos instance unless there is an error.

If the application needs to look for a state change to determine if a particular call to MC_SetPos was successful, then use `AND` for the rising edge of the **Execute** input, the **Done** output, and the inverse of the **Error** output.

This FFLD Language example shows how this can be done.
In the example, the SetPosSuccessful variable is set to `TRUE` for one cycle upon a successful call to the MC_SetPos instance.



4.2.13.2.6 IL Language Example

Not available.

4.2.13.2.7 ST Language Example

```
            (* MC_SetPos ST example *)

Inst_MC_SetPos ( Axis1 , 0, 0 );
//Inst_MC_SetPos is an instance of MC_SetPos function
```

```
(* MC_SetPos absolute mode example: Set position value to zero. *)
Inst_MC_SetPos ( Axis1 , 0, 0 );
//Inst_MC_SetPos is an instance of MC_SetPos function

(* MC_SetPos relative mode example: Increase position value by 1000. *)
Inst_MC_SetPos ( Axis1 , 1000, 1 );
//Inst_MC_SetPos is an instance of MC_SetPos function
```

4.2.13.3 MC_SetPosition

Function - Deprecated by the "MC_SetPos" (→ p. 514) function block.

4.2.14 Registration

These function blocks allow for Mark-to-Mark or Mark-to-Machine registration.

See Registration for techniques on setting up and using the registration function blocks.

Name	Description
MC_MachRegist	Enables Mark-to-Machine registration.
MC_MarkRegist	Enables Mark-to-Mark registration.
MC_StopRegist	Turns off registration for the specified axis and disarms the specified fast input.

4.2.14.1 MC_MachRegist

PLCopen

Function Block - Enables Mark-to-Machine registration.

4.2.14.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	On the rising edge, request to enable registration.
Axis	AXIS_REF	The range of .AXIS_NUM is [1, 256]	N/A	N/A	Axis whose position is used to determine a good mark.

Input	Data Type	Range	Unit	Default	Description
TriggerInput	TRIGGER_REF	No range	N/A	No default	Structure specifying the fast input. The structure elements are: **Direction**INT Range = 1, 5. • 1 = Rising edge. • 2 = Falling edge. • 3 = N/A • 4 = Toggle between both, falling edge first. • 5 = Toggle between both, rising edge first. **InputID** INT Range = 0, 1 • 0 = Touch Probe 1 / Capture Engine 0 • 1 = Touch Probe 2 / Capture Engine 1 **TrigID**INT Range = 0, 256. • The axis number of the fast input. • 0 (zero) indicates this trigger axis is to be the same as the Axis input. **NOTE** **TrigMode** INT (TriggerInput.TrigMode) is not presently supported by this function.
Distance	LREAL	• When converted to feedback units, the range is [-251, 251-1]. • This value must have the same sign as **Ignore**.	User units	N/A	• This is the expected distance between good marks. • Along with **Tolerance** and **Ignore**, this value is used to determine if the mark detected by the fast input is a good mark.

Input	Data Type	Range	Unit	Default	Description
Tolerance	LREAL	When converted to feedback units, the range is [0 , 2^{51}-1].	User units	N/A	This value specifies the distance, plus or minus, about **Distance** to determine if the mark detected by the fast input is a good mark.
Ignore	LREAL	• When converted to feedback units, the range is [-251, 251-1]. • This value must have the same sign as **Distance**.	User units	N/A	This value specifies the distance after the previous good mark in which any detected marks are ignored.
Target	LREAL	When converted to feedback units, the range is: • [-251,251-1] if **PosAxis'** rollover value is zero. • [0, PosAxis' Rollover Value] if **PosAxis'** rollover value is non-zero (i.e., ≥ 0 < PosAxis' Rollover Value).	User units	N/A	• This is the target position. • This position is compared to the actual position captured by the fast input to determine the amount of registration compensation to apply.

Input	Data Type	Range	Unit	Default	Description
Position	LREAL	When converted to feedback units, the range is: • [-251,251-1] if **PosAxis'** rollover value is zero. • [0, PosAxis' Rollover Value] if **PosAxis'** rollover value is non-zero (i.e., ≥ 0 < PosAxis' Rollover Value).	User units	N/A	• The position the axis is set to when a good registration mark occurs. • If the **Inhibit Reference on Good Mark** option is specified for the Option argument, this argument is not used. • The position of the axis is not changed when a registration mark is encountered.
PosAxis	AXIS_REF	The range of .AXIS_NUM is [1, 256]	N/A	N/A	The position of this axis at the time the fast input occurs is compared to the Target position. This comparison determines the amount of registration compensation to apply.
CompAxis	AXIS_REF	The range of .AXIS_NUM is [1, 256]	N/A	N/A	The calculated registration compensation is applied to this axis.
PosTolerance	LREAL	When converted to feedback units, the range is [-2^{51}, 2^{51}-1].	User units	N/A	This value specifies the distance, plus or minus, about the Target position to determine if the position is accepted and the compensation value is calculated and applied.

Input	Data Type	Range	Unit	Default	Description
RatioNumerator	DINT	When converted to feedback units, the range is [1, 4294967295].	User units	N/A	• This value is typically the number of User Units of **CompAxis** motion for one product cycle. • This value is used with **RatioDenominator** to create a conversion factor for calculating the compensation value when **PosAxis** and **CompAxis** are different axes.
RatioDenominator	DINT	When converted to feedback units, the range is [1, 4294967295].	User units	N/A	• This value is typically the number of User Units of **PosAxis** motion for one product cycle. • This value is used with **RatioNumerator** to create a conversion factor for calculating the compensation value when **PosAxis** and **CompAxis** are different axes.
Options	UINT	See the "MC_MachRegist Options Table" (→ p. 523).	N/A	N/A	• Each bit enables / disables an option. • The "MC_MachRegist Options Table" (→ p. 523) defines the bits. • Any bits not defined are reserved. • The third bit, 0004H, must be 0 (zero).

4.2.14.1.2 Outputs

Output	Data Type	Range	Unit	Description
RegistOn	BOOL	No range	N/A	Indicates registration is activated.

Output	Data Type	Range	Unit	Description
Aborted	BOOL	No range	N/A	Indicates registration has been terminated by "MC_StopRegist" (→ p. 531).
Error	BOOL	No range	N/A	Indicates an invalid input was specified or the registration was terminated due to an error.
ErrorID	INT	No range	N/A	• Indicates the error if **Error** output is TRUE. • See the table in PLCopen Function Block ErrorIDs.

4.2.14.1.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking.
See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

TIP

Is this the right function block to use?
See Deciding Which Function Blocks to Use for Registration and Registration Application Guide.

- It is used on any servo or digitizing axis and with any move type.
- It is used most frequently in master/slave applications.

Used with ...	Effect
Non-slave moves	Resets the axis position when a good mark is captured by the fast input.
Slave moves	Applies a compensation offset to correct for the difference between the target position and the measured position. This provides the ability to compensate for product or process inconsistencies, provides a system that remains synchronized with no accumulated error, and maintains repeatable accuracy throughout the process.

4.2.14.1.3.1 Transition and Registration

- A positive transition of the **En** input starts registration.
  - The application may change the registration parameters while registration is active by changing the input values and causing another positive transition of the **En** input.
  - The function block then reads and applies the new values.
- The axis number at the **Axis** input indicates the axis whose position, at the fast input, is used to determine if the mark is a good mark.

4.2.14.1.3.2 Distance, Tolerance, and Ignore Inputs

- The **Distance**, **Tolerance**, and **Ignore** inputs are used to determine whether or not the registration mark is good.
  - For a mark to be recognized as good, it must be outside of the **Ignore** distance and the correct **Distance** from the previous mark +/- the **Tolerance** window.
  - A mark is bad if it occurs outside of the “good tolerance band” and is not ignored.
  - Both good marks and bad marks are recognized as marks, ignored marks are not recognized.
  - If all marks are to be recognized as good marks, enter 0 (zero) at both **Distance** and **Tolerance**.

- The **Distance** value defines the distance between good marks.
  - In Clear Lane and Product registration the **Distance** input value is typically the same as the **Target** input value.
  - In **Print** registration, the **Distance** is typically not the same as **Target**.

- The **Tolerance** value is the distance, plus and minus, about **Distance**.
  - Marks detected in this window are good marks and registration occurs.
  - Marks detected outside this window and outside the **Ignore** band, are bad marks and registration does not occur.
  - This window should be large enough to allow for the worst case error in the distance between the previous mark and the current mark.

- The **Ignore** value defines the distance from the previous mark where all marks detected by the fast input are ignored.
  - This is crucial when registering products that do not have Clear Lane registration marks.

4.2.14.1.3.3 Target Input

- The **Target** input is the expected target position that is used to calculate how much registration compensation is to be applied when a registration mark is considered good.
  - When a good mark is detected, the position of the **PosAxis** is compared to the **Target** position to calculate a correction.
  - The registration correction is only applied with master/slave move types.

4.2.14.1.3.4 Position and Option Inputs

4.2.14.2 Position Inputs

- The **Position** input is the position value the registration Axis position is reset to when a good registration mark is detected.
  - When a good mark occurs, the position of the **PosAxis** is compared to the **Target** position.
    - This comparison is used to calculate the amount of registration compensation to apply to the **CompAxis**.
      - Registration compensation is applied to the axis specified at the **CompAxis** input under these conditions:
      - If **CompAxis** is a master axis, the compensation is applied to the master offsets of all its slaves.
        - This shifts the master's position as seen by its slaves.
      - If **CompAxis** is executing a slave move (e.g., "MC_GearIn" (→ p. 492) or "MC_CamIn" (→ p. 476)), the compensation is applied directly to the axis.
    - The **PosTolerance** input is the distance, plus and minus, about the **Target** position used to determine if compensation is applied.
      - When a good mark occurs, the position of the **PosAxis** axis is checked to see if it lies in the window defined by **PosTolerance**.
        - If it is in the window, compensation is applied.
        - If it is outside the window, compensation is not applied even though a good mark was found.
    - If **PosAxis** and **CompAxis** are different axes, the **RatioNumerator** and **RatioDenominator** inputs define the conversion factor for calculating the compensation value.
      - This is needed because the amount of error between actual and target positions is determined by the **PosAxis's** position and the compensation is applied to the **CompAxis**.
      - The **RatioNumerator** should typically be the number of User Units of **CompAxis** motion for one registration cycle.
      - The **RatioDenominator** should typically be the number of User Units of **PosAxis** motion for one registration cycle.
      - If **PosAxis** and **CompAxis** are the same, **RatioNumerator** and **RatioDenominator** should be the same value, thus resulting in a 1:1 ratio.

4.2.14.3 Option Inputs

- The **Option** input defines various modes of operation for registration.
  - The first bit, 0001H, selects Absolute or Resetting.
    - This refers to the way the second mark and all subsequent marks are determined to be good marks.
    - With both registration schemes, the very first detected mark is the starting point.
    - With **Resetting** registration, when the next mark is detected, the position of that mark becomes the starting point for the next good mark detection calculation and so on.
      - The starting point is reset with each good or bad mark.
        - This allows the product to re-synchronize, if necessary, due to process issues (e.g., product shift) etc.
      - **Absolute** registration determines all good marks based on the very first mark.
        - The position of the second and each subsequent mark is compared to an integer multiple of **Distance** from the very first mark.
        - This method insures the product always registers to a known fixed distance.
    - The third bit, 0004H, must be 0 (zero).
      - Mark-to-machine registration requires time-based capture.

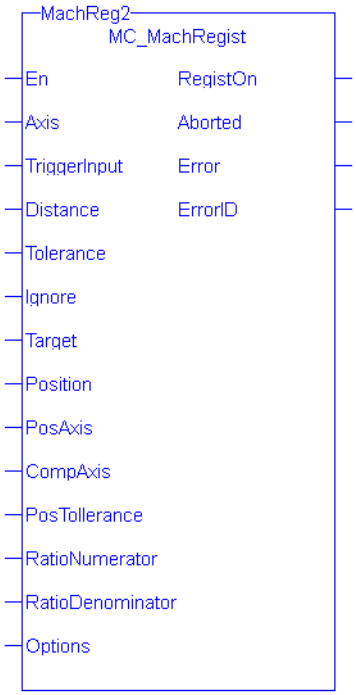


Figure 4-104: MC_MachRegist

4.2.14.3.0.1 MC_MachRegist Options Table

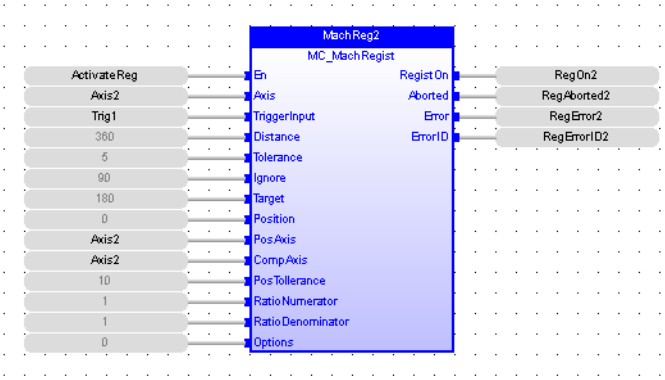
Hexadecimal	Decimal	Option	Description
0001 H	1	Absolute/Resetting	0 = Resetting, 1 = Absolute
0002 H	2	Reserved	0
0004 H	4	Time/position based capture	0 = time based capture, 1 = position based capture
0008 H	8	Inhibit Reference on Good Mark	0 = Perform reference, 1 = inhibit reference When this bit is set, the Position function block argument is unused and the axis position is not changed when a registration mark is encountered.

Hexadecimal	Decimal	Option	Description
0010H	16	Inhibit Master Compensation	0 = Perform Master Compensation, 1 = Inhibit Master Compensation
0020H	32	Inhibit Slave Compensation	0 = Perform Slave Compensation, 1 = Inhibit Slave Compensation.

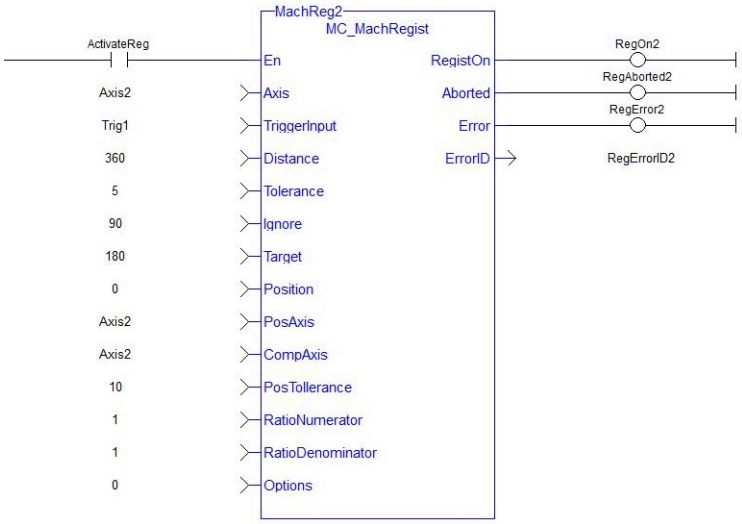
TIP

To use Capture Engine 1, modify the input PDOs that are used and add the Latch Position 1 parameter.

4.2.14.3.1 FBD Language Example



4.2.14.3.2 FFLD Language Example



4.2.14.3.3 IL Language Example

```
CAL Inst_MC_MachRegist( ActivateReg, Axist2, Trig1, 360, 5, 90, 180, 0,
Axis2, Axis2, 5, 1, 1, 0 )
```

4.2.14.3.4 ST Language Example

```
Inst_MC_MachRegist( ActivateReg, Axist2, Trig1, 360, 5, 90, 180, 0, Axis2,
Axis2, 10, 1, 1, 0 );
```

See Also

- "MC_ReadParam" (→ p. 441)
- "MC_StopRegist" (→ p. 531)
- "MC_WriteParam" (→ p. 445)

4.2.14.4 MC_MarkRegist

PLCopen

Function Block - Enables Mark-to-Mark registration.

4.2.14.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	On the rising edge, request to enable registration.
Axis	AXIS_REF	The range of .AXIS_NUM is [1, 256]	N/A	N/A	Axis to apply registration to.
TriggerInput	TRIGGER_REF	No range	N/A	No default	Structure specifying the fast input. The structure elements are: **Direction**INT Range = 1, 5. • 1 = Rising edge. • 2 = Falling edge. • 3 = N/A • 4 = Toggle between both, falling edge first. • 5 = Toggle between both, rising edge first. **InputID** INT Range = 0, 1 • 0 = Touch Probe 1 / Capture Engine 0 • 1 = Touch Probe 2 / Capture Engine 1 **TrigID**INT Range = 0, 256. • The axis number of the fast input. • 0 (zero) indicates this trigger axis is to be the same as the Axis input. **NOTE** **TrigMode** INT (TriggerInput.TrigMode) is not presently supported by this function.

Input	Data Type	Range	Unit	Default	Description
Distance	LREAL	When converted to feedback units, the range is [-2^{51}, 2^{51}-1].	User units	N/A	• This value must have the same sign as **Ignore**. • This is the expected distance between good marks. • Along with **Tolerance** and **Ignore**, this value is used to determine if the mark detected by the fast input is a good mark.
Tolerance	LREAL	When converted to feedback units, the range is [0 , 2^{51}-1].	User units	N/A	This value specifies the distance, plus or minus, about **Distance** to determine if the mark detected by the fast input is a good mark.
Ignore	LREAL	• When converted to feedback units, the range is [-2^{51}, 2^{51}-1].	User units	N/A	• This value must have the same sign as **Distance**. • This value specifies the distance after the previous good mark in which any detected marks are ignored.
Target	LREAL	When converted to feedback units, the range is [-2^{51}, 2^{51}-1].	User units	N/A	• This value must have the same sign as **Distance**. • This is the target distance between good marks. • This distance is compared to the actual distance measured by the fast input to determine the amount of registration compensation to apply.

Input	Data Type	Range	Unit	Default	Description
Position	LREAL	When converted to feedback units, the range is: • [-251,251-1] if **PosAxis'** rollover value is zero. • [0, PosAxis' Rollover Value] if **PosAxis'** rollover value is non-zero (i.e., ≥ 0 < PosAxis' Rollover Value).	User units	N/A	• The position the axis is set to when a good registration mark occurs. • If the **Inhibit Reference on Good Mark** option is specified for the Option argument, this argument is not used. • The position of the axis is not changed when a registration mark is encountered.
Options	UINT	See the "MC_MachRegist Options Table" (→ p. 529).	N/A	N/A	• Each bit enables / disables an option. • The "MC_MachRegist Options Table" (→ p. 529) defines the bits. • Any bits not defined are reserved. • The third bit, 0004H, must be 0 (zero).

4.2.14.4.2 Outputs

Output	Data Type	Range	Unit	Description
RegistOn	BOOL	No range	N/A	Indicates registration is activated.
Aborted	BOOL	No range	N/A	Indicates registration has been terminated by "MC_StopRegist" (→ p. 531).
Error	BOOL	No range	N/A	Indicates an invalid input was specified or the registration was terminated due to an error.
ErrorID	INT	No range	N/A	• Indicates the error if **Error** output is TRUE. • See the table in PLCopen Function Block ErrorIDs.

4.2.14.4.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

TIP

Is this the right function block to use?
See Deciding Which Function Blocks to Use for Registration and Registration Application Guide.

- It is used on any servo or digitizing axis and with any move type.
- It is used most frequently in master/slave applications.

Used with ...	Effect
Non-slave moves	Resets the axis position when a good mark is captured by the fast input.
Slave moves	In addition to resetting the axis position, applies a compensation offset to correct for the difference between the target mark-to-mark distance and the measured mark-to-mark distance. This provides the ability to compensate for product or process inconsistencies, provides a system that remains synchronized with no accumulated error, and maintains repeatable accuracy throughout the process.

4.2.14.4.3.1 Transition and Registration

- A positive transition of the **En** input starts registration.
  - The application may change the registration parameters while registration is active by changing the input values and causing another positive transition of the **En** input.
  - The function block then reads and applies the new values.

- The axis number at the **Axis** input identifies the axis of registration.
  - If **Axis** is a master axis for another axis's slave move, Master Registration is activated.
  - Master Registration calculates a compensation that is added to the master offset of its slaves.
    - This offset shifts the position of the master axis as seen by its slaves.
    - The compensation is not applied to the master axis, but to its slaves.

  - If **Axis** is a slave axis, Slave Registration is activated.
    - Slave Registration calculates a compensation that is added to the slave offset of the axis.
    - This compensation value is applied directly to the slave axis.

4.2.14.4.3.2 Distance, Tolerance, and Ignore Inputs

- The **Distance**, **Tolerance**, and **Ignore** inputs are used to determine whether or not the registration mark is good.
  - For a mark to be recognized as good, it must be outside of the **Ignore** distance and the correct **Distance** from the previous mark +/- the **Tolerance** window.
  - A mark is bad if it occurs outside of the "good tolerance band" and is not ignored.
  - Both good marks and bad marks are recognized as marks, ignored marks are not recognized.
  - If all marks are to be recognized as good marks, enter 0 (zero) at both **Distance** and **Tolerance**.

- The **Distance** value defines the distance between good marks.
  - In Clear Lane and Product registration the **Distance** input value is typically the same as the **Target** input value.
  - In **Print** registration, the **Distance** is typically not the same as **Target**.

- The **Tolerance** value is the distance, plus and minus, about **Distance**.
  - Marks detected in this window are good marks and registration occurs.
  - Marks detected outside this window and outside the **Ignore** band, are bad marks and registration does not occur.
  - This window should be large enough to allow for the worst case error in the distance between the previous mark and the current mark.

- The **Ignore** value defines the distance from the previous mark where all marks detected by the fast input are ignored.
  - This is crucial when registering products that do not have Clear Lane registration marks.

4.2.14.4.3.3 Target Input

- The **Target** input is the expected distance between good registration marks.
  - It is used to calculate how much registration compensation is applied when a registration mark is considered good.
  - In many applications, this is equivalent to the product length or the cycle length.
  - When a good mark is detected, the actual distance between the good mark and the previous mark is determined and compared to the **Target** distance to calculate a correction.
  - The registration correction is only applied with master/slave move types and always affects the slave axis.

4.2.14.4.3.4 Position and Option Inputs

- The **Position** input is the position value the registration Axis position is reset to when a good registration mark is detected.
- The **Option** input defines various modes of operation for registration.
  - The first bit, 0001H, selects Absolute or Resetting.
  - This refers to the way the second mark and all subsequent marks are determined to be good marks.
  - With both registration schemes, the very first detected mark is the starting point.
  - With **Resetting** registration, when the next mark is detected, the position of that mark becomes the starting point for the next good mark detection calculation and so on.
  - The starting point is reset with each good or bad mark.
    - This allows the product to re-synchronize, if necessary, due to process issues (e.g., product shift) etc.

  - **Absolute** registration determines all good marks based on the very first mark.
    - The position of the second and each subsequent mark is compared to an integer multiple of **Distance** from the very first mark.
    - This method insures the product always registers to a known fixed distance.

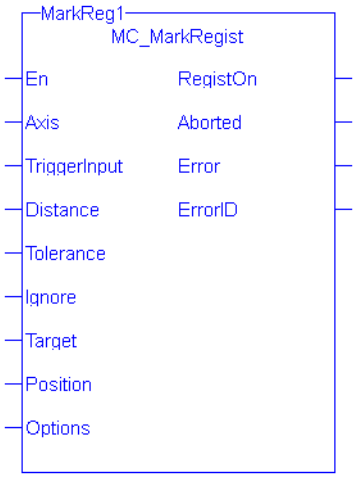


Figure 4-105: MC_MarkRegist

4.2.14.4.4 MC_MachRegist Options Table

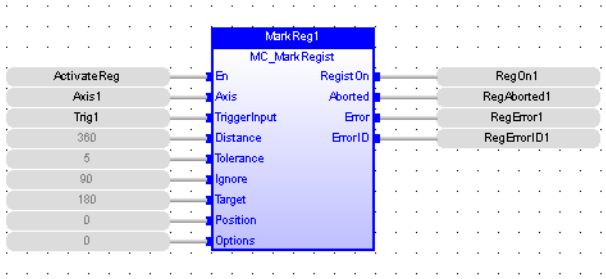
Hexadecimal	Decimal	Option	Description
0001 H	1	Absolute/Resetting	0 = Resetting, 1 = Absolute
0002 H	2	Reserved	0
0004 H	4	Time/position based capture	0 = time based capture, 1 = position based capture

Hexadecimal	Decimal	Option	Description
0008 H	8	Inhibit Reference on Good Mark	0 = Perform reference, 1 = inhibit reference When this bit is set, the Position function block argument is unused and the axis position is not changed when a registration mark is encountered.
0010H	16	Inhibit Master Compensation	0 = Perform Master Compensation, 1 = Inhibit Master Compensation
0020H	32	Inhibit Slave Compensation	0 = Perform Slave Compensation, 1 = Inhibit Slave Compensation.

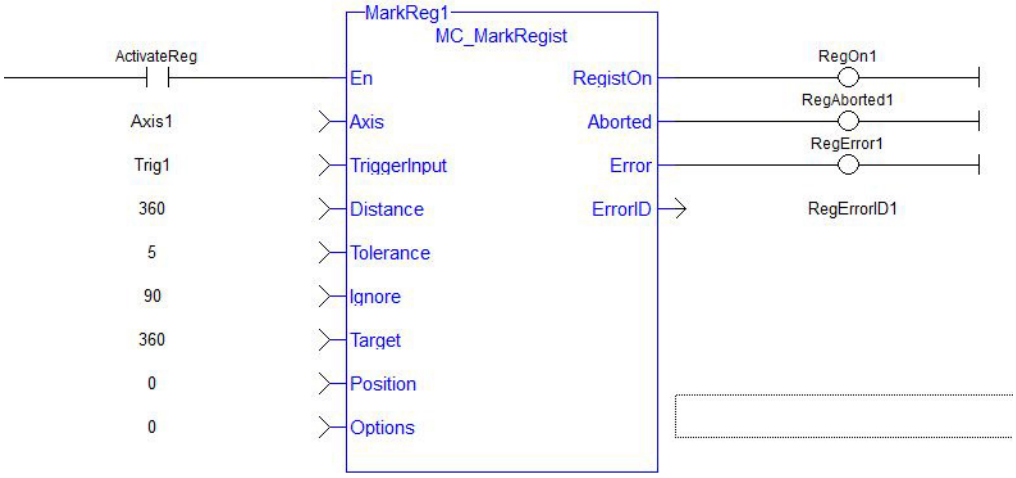
TIP

To use Capture Engine 1, modify the input PDOs that are used and add the Latch Position 1 parameter.

4.2.14.4.5 FBD Language Example



4.2.14.4.6 FFLD Language Example



4.2.14.4.7 IL Language Example

```
CAL Inst_MC_MarkRegist( ActivateReg, Axist1, Trig1, 360, 5, 90, 360, 0 0 )
```

4.2.14.4.8 ST Language Example

```
Inst_MC_MarkRegist( ActivateReg, Axist1, Trig1, 360, 5, 90, 360, 0, 0 );
```

See Also

- "MC_ReadParam" (→ p. 441)
- "MC_Stop" (→ p. 423)
- "MC_WriteParam" (→ p. 445)

4.2.14.5 MC_StopRegist

PLCopen

Function Block - Turns off registration for the specified axis and disarms the specified fast input.

4.2.14.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Enables execution.
Axis	AXIS_REF	1, 256	N/A	N/A	Axis registration to turn off.
TriggerInput	TRIGGER_REF	See Description.	N/A	No default	Structure specifying the fast input. (to disarm) The structure elements are: **Direction**INT Range = 1, 5. • 1 = Rising edge. • 2 = Falling edge. • 3 = N/A • 4 = Toggle between both, falling edge first. • 5 = Toggle between both, rising edge first. **InputID** INT Range = 0, 1 • 0 = Touch Probe 1 / Capture Engine 0 • 1 = Touch Probe 2 / Capture Engine 1 **Trigid**INT Range = 0, 256. • Axis number of the fast input. • 0 (zero) indicates this trigger axis is to be the same as the Axis input. . **NOTE** **TrigMode** INT (TriggerInput.TrigMode) is not presently supported by this function.

4.2.14.5.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL			Execution successful.

4.2.14.5.3 Remarks

NOTE

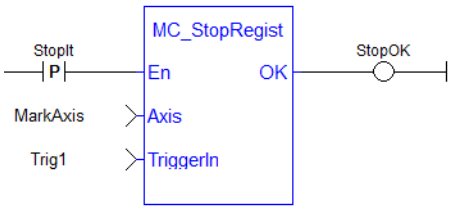
This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

TIP

To use Capture Engine 1, modify the input PDOs that are used and add the Latch Position 1 parameter.

4.2.14.5.4 FBD Language Example

4.2.14.5.5 FFLD Language Example



4.2.14.5.6 IL Language Example

Not available.

4.2.14.5.7 ST Language Example

```
StopOK := MC_StopRegist( Axis1, Trig1);
```

See Also

- "MC_MachRegist" (→ p. 516)
- "MC_MarkRegist" (→ p. 525)

4.3 Motion / Common

- "Coordinated Motion" (→ p. 532)
- "Info" (→ p. 532)
- "Motion" (→ p. 532)
- "Profiles" (→ p. 533)

4.3.1 Coordinated Motion

"Motion / Common - Coordinated Motion" (→ p. 557)

4.3.2 Info

Name	Description
MC_ErrorDescription	Converts the PLCopen error IDs into message strings.

4.3.3 Motion

Name	Description
MLMotionCycleTime	Returns the Motion Base Cycle time in seconds.
MLMotionInit	Initializes the motion library.
MLMotionRstErr	Clears motion engine errors, motion bus driver errors, and EtherCAT network errors.
MLMotionStart	Starts the motion engine, motion bus driver, clears the EtherCAT diagnostic registers of all nodes, and initializes EtherCAT network to operational mode.
MLMotionStatus	Returns the status of the motion engine 0 (zero).
MLMotionStop	Stops the motion bus driver, motion engine, and EtherCAT network operation, resulting in the EtherCAT network transitioning to the Init state.

4.3.4 Profiles

Name	Description
MLProfileBuild	Creates a CAM Profile from application data that can be executed by a CAM block in Pipe Network or PLCopen.
MLProfileCreate	Creates a new CAM Profile Object for use in a PLC Program or Pipe Network CAM block.
MLProfileInit	Initializes a previously created CAM Profile object for use in a PLC Program or Pipe Network CAM block.
MLProfileRelease	Removes a Profile so the ProfileID can be used by a different or new Profile.

4.3.5 Motion / Common - Info

Name	Description
MC_ErrorDescription	Converts the PLCopen error IDs into message strings.

4.3.5.1 MC_ErrorDescription

PLCopen ✓ Pipe Network ✓

Function - Converts the PLCopen error IDs into message strings.

4.3.5.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	If TRUE, this function converts the ErrorID to a string message.
ErrorID	INT	0, 69	N/A	No default	Error ID generated from a PLCopen function block. See the table in PLCopen Function Block ErrorIDs.

4.3.5.1.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	0, 1	N/A	If TRUE, the command completed successfully.
Description	STRING	No range	N/A	String error description.

4.3.5.1.3 Remarks

- Return a text description corresponding to a motion control error ID code.
- The message strings can be used for display or logging.

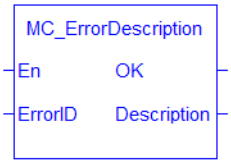
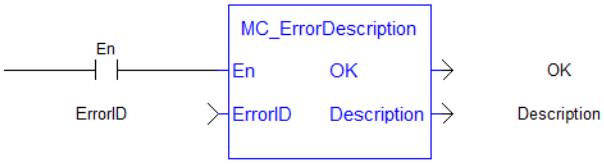


Figure 4-106: MC_ErrorDescription

4.3.5.1.4 FBD Language Example

4.3.5.1.5 FFLD Language Example



4.3.5.1.6 IL Language Example

Not available.

4.3.5.1.7 ST Language Example

```
Description:= MC_ErrorDescription(ErrorID);
```

4.3.6 Motion / Common - Profiles

Name	Description
MLProfileBuild	Creates a CAM Profile from application data that can be executed by a CAM block in Pipe Network or PLCopen.
MLProfileCreate	Creates a new CAM Profile Object for use in a PLC Program or Pipe Network CAM block.
MLProfileInit	Initializes a previously created CAM Profile object for use in a PLC Program or Pipe Network CAM block.
MLProfileRelease	Removes a Profile so the ProfileID can be used by a different or new Profile.

4.3.6.1 MLProfileBuild

PLCopen ✓ Pipe Network ✓

Function - Creates a CAM Profile from application data that can be executed by a CAM block in Pipe Network or PLCopen.

4.3.6.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	No range	N/A	No default	Enables execution. Starts on the rising edge.
Cam_Props	CamProps_ref	No range	N/A	No default	Structures containing the CAM Profile properties.
Cam_Data	CamData_ref	N=3 to 20,000 elements	N/A	3 elements minimum size	Array of structures containing the CAM Profile data.
CamDataCount	UINT	3 - 20,000	Elements	3	Number of elements in the Cam_Data array to be used.
ProfileID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of an initialized CAM Profile.
Cyclic	BOOL	0, 1 TRUE, FALSE	N/A	No default	• FALSE: One time through the profile. • TRUE: Repeating profile.
Options	UINT	• CAM_PROFILE_OPTION_DEFAULT: Allows the use of point and line segments. • CAM_PROFILE_OPTION_PARABOLIC: Allows the use of line and parabolic segments. **NOTE** The DEFAULT and PARABOLIC options cannot be combined. A CAM Profile can only use point and line segments, or parabolic and line segments. Point and parabolic segments cannot both be used in the same profile.	N/A	No default	Describes the combinations of segments that can be used to build a cam profile.

4.3.6.1.2

4.3.6.1.3 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	0, 1 TRUE, FALSE	N/A	Indication of whether or not the profile was successfully compiled and built.
Busy	BOOL	0, 1 TRUE, FALSE	N/A	Indication that the function block is executing. • TRUE if executing. • FALSE if not executing.
Err	BOOL	0, 1 TRUE, FALSE	N/A	Indication that the function did not execute correctly. ErrorID output is valid and indicate the reason.
ErrorID	INT	No range	N/A	Indicates the reason for the failure. See the "Error Codes" (→ p. 538) table.
ErrorElem	UINT	No range	N/A	The array element number of the CAM data where an error is detected.

4.3.6.1.4 Remarks

- This function block takes input as CAM data and CAM Profile properties from application data memory and compile the input data to a form the controller can use to calculate cam positions.
  - See Profile Editor's Table for information.
- The input CAM data and CAM Profile properties are similar to the CAM data entered in the IDE's CAM Editor and the runtime's CAM Profile Properties dialog.

MLProfileBuild internally performs two functions:

1. Compile the CAM data (like the cam editor performs in the IDE).
2. Puts the compiled profile into the profile object so it can be used by other Profile Function Blocks. Provides similar functionality to "MLProfileInit" (→ p. 544).

NOTE

Prior to using **MLProfileBuild**, you must call "MLProfileCreate" (→ p. 542) to create the profile object. The `ID` output of **MLProfileCreate** is used as the `ProfileID` input to **MLProfileBuild**. **MLProfileCreate** must be performed in the application **before** the "MLMotionStart" (→ p. 552) command is executed.

- **MLProfileBuild** compiles the CAM Profile data specified at the `CamData` input and writes the resulting profile to the CAM Profile object specified at input `ProfileID`.
- The created profile can be used as an input to PLCopen CAM Function Blocks (e.g., MC_CamTblSelect, MC_CamIn, MC_CamOut), or any Pipe Network CAM Profile Function/Function Blocks.
- When the operation is complete, the `Done` output is high.
  - If an error is encountered, the `Error` output is high and the `ErrorID` output returns an error code.
  - If the Error can be attributed to a specific profile element in the `CamData` array, `ErrorElem` attempts to indicate the element in error.

4.3.6.1.4.1 CamProps_Ref Structure

See Master/Input offset for more information about the parameters which transform the CAM Profile.

CamProps_Ref Structure

The CAM properties structure (CamProps_Ref) contains these data members:

Parameter	Type	Description
InputScale	LREAL	The input amplitude or master axis multiplier applied to the CAM Profile.
OutputScale	LREAL	The output amplitude or slave axis multiplier applied to the CAM Profile.
InputOffset	LREAL	input offset or master axis shift applied to the CAM Profile.
OutputOffset	LREAL	The output offset or slave axis shift applied to the CAM Profile.

4.3.6.1.4.2 CamData_Ref Structure

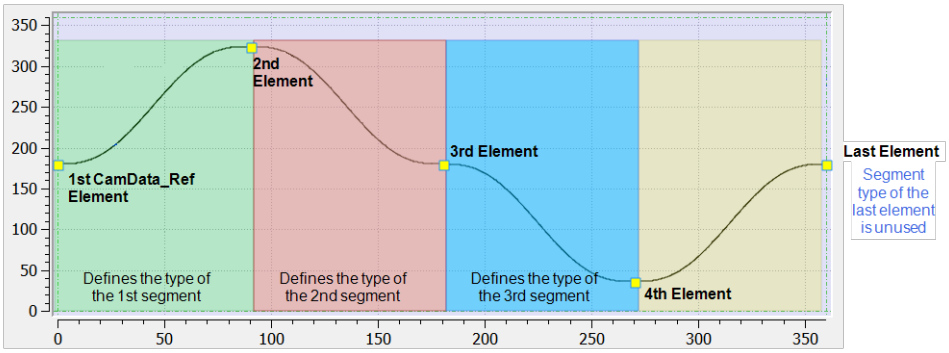
The **Cam_Data** function block input is an array of **CamData_Ref** structures.

CamData_Ref Structure

Each element of the structure contains these data members:

Parameter	Type	Description
Accel	LREAL	**CAM_SEGMENT_TYPE_POINT** • **Accel** represents the cam acceleration at the master position specified by **MasterIn**. • Units: `(slave position user units) / (master position user units)` **CAM_SEGMENT_TYPE_LINE and CAM_SEGMENT_TYPE_PARABOLIC** • **Accel** is ignored.
MasterIn	LREAL	Master position (in the unit range [0 - InputScale]).
SegType	UINT	Defines the segment type for the segment following the master positions defined by **MasterIn**. 1. CAM_SEGMENT_TYPE_POINT = Point 5th order polynomial). 2. CAM_SEGMENT_TYPE_LINE = Line (constant velocity segment). 3. CAM_SEGMENT_TYPE_PARABOLIC = Parabolic (constant acceleration). See Cam Profile Segment Overview for information on the segment types.
SlaveOut	LREAL	Slave position (in the unit range [0 - OutputScale]).
Vel	LREAL	CAM velocity at the master position specified by **MasterIn**. Units: `(slave position user units) / (master position user units)`

- The type of the Nth CAM segment is defined by the Nth Cam_Data element.
- The CAM is constructed with one less segment than the Cam_Data elements.
- The last element's SegType is not used.
- See Profile Editor's Table for more information.



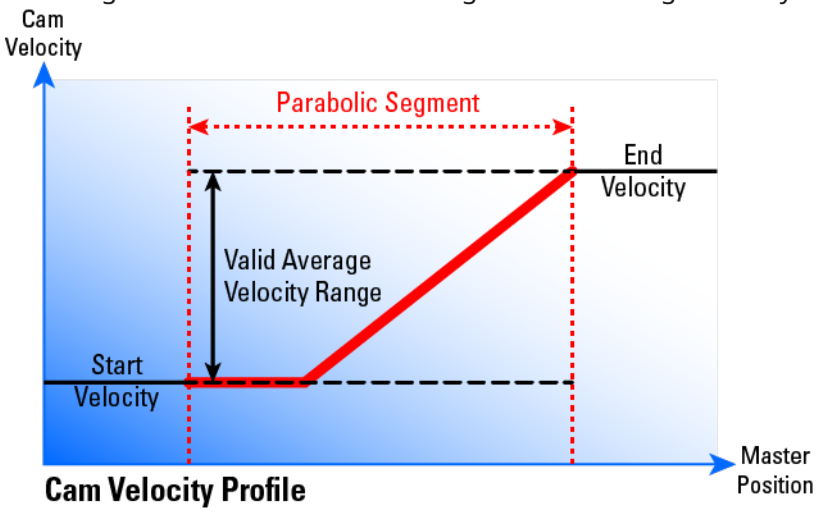
4.3.6.1.4.3 Error Codes

NOTE

If **Cyclic** is TRUE and the Vel/Accel of the first and last elements do not match, **MLProfileBuild** automatically copies the first element's Vel/Accel to the last element's.
A log warning message is posted indicating that this change has occurred.

Error Codes

ErrorID	Description
100	`Cam_Data` array does not have `CamDataCount` elements. The `Cam_Data` array is not large enough to hold the specified number of elements.
101	Invalid master or slave scale. The scale cannot be less than zero.
102	Element master or slave position is outside the range defined by `Cam_Props`.
103	A segment type was specified that is not supported by the value of the `Options` argument.
104	Master position of an element is too close to the master position of a previous element. • `0.0000125 * InputScale` is the minimum distance allowed. • See "CamProps_Ref Structure" (➔ p. 536) for more information. • Each element is compared to all previous elements. • If the master distance between any two elements in the list are too close, this error is generated.
106	Invalid ProfileID. This can occur if the ProfileID: • Does not exist. • Has not been created yet. • The ProfileID is not a profile.
107	`CamDataCount` exceeds maximum array size of 20,000.
108	Profile cannot be released because it is in use by the motion engine or currently selected by an active CAM block.
109	Attempting to build a profile already containing elements. Profile needs to be released first using MLProfileRelease.
110	The controller is running low on memory and could not allocate memory to hold the cam table data.
111	`CamDataCount` is not large enough. The minimum allowed value is 3.

ErrorID	Description
112	CAM_PROFILE_OPTION_PARABOLIC • Elements are not sorted in increasing order by master position. • After the first element, each element must have its master position be greater than the master position of the previous element.
113	CAM_SEGMENT_TYPE_PARABOLIC • The average velocity for the segment is outside of range defined by start and end element velocities. • This diagram illustrates the valid range for the average velocity:  Cam Velocity Profile
200	First element's `MasterIn` value not equal to zero.
201	Last element's `MasterIn` value does not equal value of X-amplitude.
202	Cannot modify the first element in the cam element table. `SlaveOut` value is outside the output range specified by `Cam_Props`.
203	Cannot modify the last element in the cam element table. `SlaveOut` value is outside the output range specified by `Cam_Props`.

4.3.6.1.4.4 Example: Use MLProfileBuild

Example: Use MLProfileBuild

This example has these procedure steps:

- "1. Create a Profile" (→ p. 539)
- "2. Define the Profile Data" (→ p. 540)
- "3. Define the Profile Properties" (→ p. 540)
- "4. Call the MLProfileBuild Function Block" (→ p. 540)
- "5. Use the New Profile" (→ p. 540)

4.3.6.1.4.5 1. Create a Profile

IMPORTANT

Prior to using **MLProfileBuild**, you must first create a profile.
This must be done prior to **MLMotionStart**.

```
            // Allocate space for a profile that will be built later

profileID := MLProfileCreate('ProfileName');
```

4.3.6.1.4.6 2. Define the Profile Data

1. Create an array of CamData_Ref structures in the data dictionary.
2. Enter each of the elements into that newly created structure.
   In this example, `ProfileData` is the name of the CamData_Ref structure.

```
            // Define the profile data
ProfileData[0].MasterIn := 0.0;
ProfileData[0].SlaveOut := 180.0;
ProfileData[0].SegType := CAM_SEGMENT_TYPE_POINT;
ProfileData[0].Velocity := 0.0;
ProfileData[0].Acceleration := 0.0;

ProfileData[1].MasterIn := 180.0;
ProfileData[1].SlaveOut := 324.0;
ProfileData[1].SegType := CAM_SEGMENT_TYPE_POINT;
ProfileData[1].Velocity := 0.5;
ProfileData[1].Acceleration := -0.025;

ProfileData[2].MasterIn := 360.0;
ProfileData[2].SlaveOut := 240.0;
ProfileData[2].SegType := CAM_SEGMENT_TYPE_POINT;
ProfileData[2].Velocity := 0.0;
ProfileData[2].Acceleration := 0.0;
```

4.3.6.1.4.7 3. Define the Profile Properties

1. Create a CamProps_Ref structure in the data dictionary.
2. Enter each of the properties into the newly created structure.
   In this example, `ProfileProps` is the name of the CamProps_Ref structure.

```
            // Define the profile properties
ProfileProps.InputScale := 360.0;  // Must be Positive!
ProfileProps.OutputScale := 360.0; // Must be Positive!
ProfileProps.InputOffset := 0.0;
ProfileProps.OutputOffset := 0.0;
```

4.3.6.1.4.8 4. Call the MLProfileBuild Function Block

Use the IEC language of choice.

As part of this call it is recommended that you validate the `Done` and `Error` output before proceeding.

```
            // Build the profile
Inst_MLProfileBuild( TRUE, ProfileProps, ProfileData, 3, ProfileID, TRUE,
CAM_PROFILE_OPTION_DEFAULT);
```

4.3.6.1.4.9 5. Use the New Profile

After verifying that **MLProfileBuild** is `Done` and there are no errors, you can use the newly generated profile.

The next step depends on the motion engine in use.

- PLCopen: call "MC_CamTblSelect" (→ p. 490).
- Pipe Network: call either "MLCamInit" (→ p. 280) or "MLCamSwitch" (→ p. 281).

IMPORTANT

Pipe Network: To correctly set the CAM scales and offsets (defined by the Cam_Props argument) "MLPrfWriteIScale" (→ p. 289), "MLPrfWriteOScale" (→ p. 291), "MLPrfWriteIOffset" (→ p. 287) and "MLPrfWriteOOffset" (→ p. 290) must be called before calling "MLCamSwitch" (→ p. 281).

```
            // Switch Pipe Network Profile
MLPrfWriteIScale(profileID, ProfileProps.InputScale);
MLPrfWriteOScale(profileID, ProfileProps.OutputScale);
MLPrfWriteIOffset(profileID, ProfileProps.InputOffset);
MLPrfWriteOOffset(profileID, ProfileProps.OutputOffset);
MLCamSwitch(PipeNetwork.CAM, profileID);
```

4.3.6.1.4.10 Example: Building a Parabolic Cam Profile

To build a parabolic CAM Profile, your CAM data elements must use `CAM_SEGMENT_TYPE_PARABOLIC` or `CAM_SEGMENT_TYPE_LINE` when defining the cam data array.

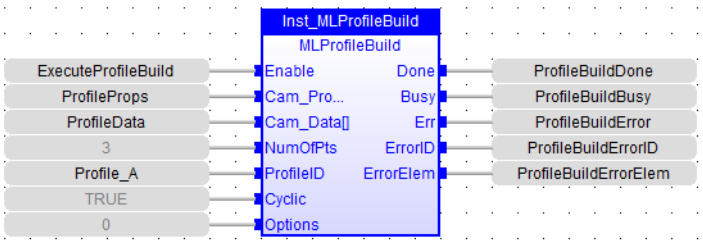
Example: Building a Parabolic Cam Profile

```
            // Define the profile data
ProfileData[0].MasterIn     := 0.0;
ProfileData[0].SlaveOut     := 0.0;
ProfileData[0].SegType      := CAM_SEGMENT_TYPE_PARABOLIC;
ProfileData[0].Velocity     := 0.0;
ProfileData[0].Acceleration := 0.5;

ProfileData[1].MasterIn     := 50.0;
ProfileData[1].SlaveOut     := 150.0;
ProfileData[1].SegType      := CAM_SEGMENT_TYPE_LINE;
ProfileData[1].Velocity     := 5.0;
ProfileData[1].Acceleration := 0.0;            // Not used

ProfileData[2].MasterIn     := 55.0;
ProfileData[2].SlaveOut     := 175.0;
ProfileData[2].SegType      := CAM_SEGMENT_TYPE_PARABOLIC;
ProfileData[2].Velocity     := 5.0;
ProfileData[2].Acceleration := 0.0;            // No limit to the acceleration rate
of the segment.

ProfileData[3].MasterIn     := 105.0;
ProfileData[3].SlaveOut     := 250.0;
ProfileData[3].SegType      := CAM_SEGMENT_TYPE_PARABOLIC;
ProfileData[3].Velocity     := 0.0;
ProfileData[3].Acceleration := 0.5;

ProfileData[4].MasterIn     := 225.0;
ProfileData[4].SlaveOut     := 125.0;
ProfileData[4].SegType      := CAM_SEGMENT_TYPE_PARABOLIC;
ProfileData[4].Velocity     := -10.0;
ProfileData[4].Acceleration := 0.5;

ProfileData[5].MasterIn     := 360.0;
ProfileData[5].SlaveOut     := 0.0;
ProfileData[5].SegType      := CAM_SEGMENT_TYPE_PARABOLIC;     // Not used
ProfileData[5].Velocity     := 0.0;
ProfileData[5].Acceleration := 0.0;            // Not used
```

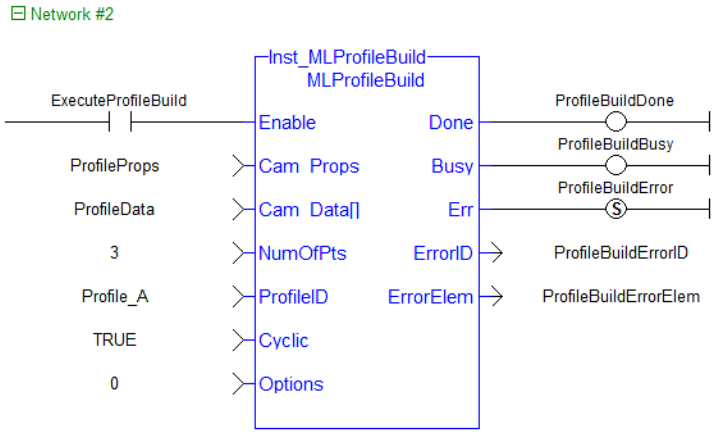
- When calling the MLProfileBuild function block, verify `CAM_PROFILE_OPTION_PARABOLIC` is specified for the Option argument: in the IEC language of choice.
- As part of this call, it is recommended that you validate the **Done** and **Error** output before proceeding.

```
// Build the profile
Inst_MLProfileBuild( TRUE, ProfileProps, ProfileData, 3, ProfileID, TRUE,
CAM_PROFILE_OPTION_PARABOLIC);
```

4.3.6.1.5 FBD Language Example



4.3.6.1.6 FFLD Language Example



4.3.6.1.7 IL Language Example

Not available.

4.3.6.1.8 ST Language Example

```
            // Build the profile
Inst_MLProfileBuild( TRUE, ProfileProps, ProfileData, 3, ProfileID, TRUE,
CAM_PROFILE_OPTION_DEFAULT);
```

See Also

- Cam Profile Segment Overview
- "MC_CamIn" (→ p. 476)
- "MC_CamOut" (→ p. 484)
- MC_CamTblSelect
- "MLCamInit" (→ p. 280)
- "MLCamSwitch" (→ p. 281)
- "MLProfileCreate" (→ p. 542)
- "MLProfileInit" (→ p. 544)
- "MLProfileRelease" (→ p. 545)

4.3.6.2 MLProfileCreate

PLCopen ✓ Pipe Network ✓

Function - Creates a new CAM Profile Object for use in a PLC Program or Pipe Network CAM block.

4.3.6.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Name	STRING	No range	N/A	No default	Name of initialized CAM Profile

4.3.6.2.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Indicates the profile has been created.
ID	DINT	No range	N/A	Returns the ID number of the created CAM Profile. • If MLProfileCreate(...) fails, the ID is zero (NULL). • A CAM ProfileID = 0 (zero) is not valid for cam profile functions.

4.3.6.2.3 Remarks

This function block is automatically called if a Profile is created in the Project Explorer with user-defined settings entered in the CAM Profile Properties screen.

- Profiles are created and initiated separately.
- The shape is modified with the CAM Editor.
  - With the Editor, profiles can be changed graphically or by manually changing values in a numeric table relating input and output values with specific slopes.
  - The Cam Editor software tool provides the capability to visualize, analyze, edit, and smooth profiles.
- Profile switching can be done on the fly without losing synchronization and without dead time.
  - Offsets and ratios of CAM Profiles can be changed on the fly.

NOTE

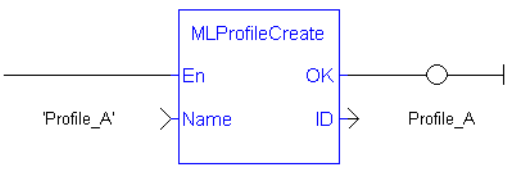
Profile objects are normally created in the Project Explorer.
You do not have to add **MLCamInit** function blocks to their programs.
Right click the Profiles folder under PLC->Motion and click Add to create a new profile.
Parameters are entered directly in a window and the code is automatically added to the current project.

TIP

This function should be called after "MLMotionInit" (→ p. 549) is called and before "MLMotionStart" (→ p. 552) is called.

4.3.6.2.4 FBD Language Example

4.3.6.2.5 FFLD Language Example



4.3.6.2.6 IL Language Example

Not available.

4.3.6.2.7 ST Language Example

```
//Create a new Profile
Profile_A := MLProfileCreate( 'Profile_A' );
```

See Also

- "MLProfileInit" (→ p. 544)
- "MLCamInit" (→ p. 280)

4.3.6.3 MLProfileInit

PLCopen ✓ Pipe Network ✓

Function - Initializes a previously created CAM Profile object for use in a PLC Program or Pipe Network CAM block.

4.3.6.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ProfileID	DINT	-2147483648 to 2147483647	N/A	No default	ID number of a created CAM Profile.
FileName	STRING	No range	N/A	No default	Filename used to save Profile on the computer's hard disk.
InputScale	LREAL	Positive	N/A	No default	The input amplitude or X-axis multiplier applied to the CAM Profile.
OutputScale	LREAL	No range	N/A	No default	The output amplitude or Y-axis multiplier applied to the CAM Profile.
InputOffset	LREAL	No range	N/A	No default	The input offset or X-axis shift applied to the CAM Profile.
OutputOffset	LREAL	No range	N/A	No default	The output offset or Y-axis shift applied to the CAM Profile.

4.3.6.3.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL			Returns TRUE if a new CAM Profile is initialized.

4.3.6.3.3 Remarks

This function block is automatically called if a Profile is created in the Project Explorer with user-defined settings entered in the CAM Profile Properties screen.

- Profiles are created and initiated separately.
- The shape is modified with the CAM Editor.
  - With the Editor, profiles can be changed graphically or by manually changing values in a numeric table relating input and output values with specific slopes.
  - The Cam Editor software tool provides the capability to visualize, analyze, edit, and smooth profiles.
- Profile switching can be done on the fly without losing synchronization and without dead time.
  - Offsets and ratios of CAM Profiles can be changed on the fly.

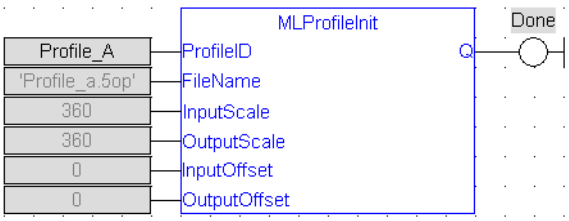
NOTE

Profile objects are normally created in the Project Explorer.
You do not have to add **MLCamInit** function blocks to their programs.
Right click the Profiles folder under PLC->Motion and click Add to create a new profile.
Parameters are entered directly in a window and the code is automatically added to the current project.

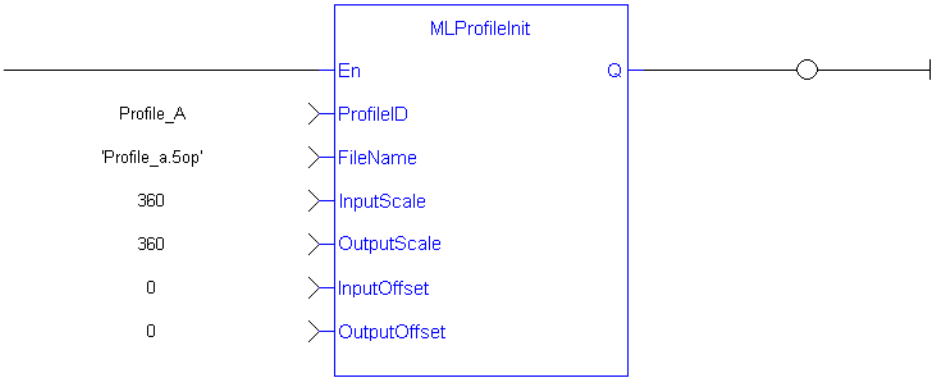
IMPORTANT

Loading a Profile Editor-generated profile into a ProfileID released by **MLProfileRelease** should be done with care.
The **MLProfileInit ()** function call can take in excess of 4 milliseconds to execute.
Application execution is suspended during this time until the function call is completed.

4.3.6.3.4 FBD Language Example



4.3.6.3.5 FFLD Language Example



4.3.6.3.6 IL Language Example

Not available.

4.3.6.3.7 ST Language Example

```
//Initialize a previously created CAM Profile
MLProfileCreate( Profile_A , 'Profile_A.5op' , 360, 360, 0, 0 );
```

See Also

- "MLProfileCreate" (➔ p. 542)
- "MLCamInit" (➔ p. 280)

4.3.6.4 MLProfileRelease

PLCopen ✓ Pipe Network ✓

Function - Removes a Profile so the ProfileID can be used by a different or new Profile.

4.3.6.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	0	Enables execution. • Successful completion results in a profile ID no longer assigned to a specific profile and can be reused for a different/new Profile. • Prior to reusing this Profile ID, it must be re-initialized by either an **MLProfileInit** function call or by calling **MLProfileBuild**.
ProfileID	DINT	1, 256	N/A	0	Specify a Profile ID that has been created by MLProfileCreate. • This is the profile ID released so it can be reused for a different or new Profile. • This Profile ID must not be in use by a motion engine.

4.3.6.4.2

4.3.6.4.3 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	0, 1	N/A	If high, Successful completion. The Profile can now be reused.
Err	BOOL	0, 1	N/A	If high, the function block did not complete successfully. Reason is given in ErrorID.
ErrorID	INT	No range	N/A	Indicates the reason for the failure. See the "Error Codes" (→ p. 547) table.

4.3.6.4.4 Remarks

-
- An application program is limited to 256 ProfileIDs.
- Once the existing ProfileID definition has been successfully released, the ProfileID can be used by either "MLProfileInit" (→ p. 544) or "MLProfileBuild" (→ p. 534) to create a new Profile.
- The ProfileID selected by the input parameter must not be in-use by a motion engine.

4.3.6.4.4.1 In-use Definitions

- Pipe Network – Must not be currently selected for use by an active CAM block in an active pipe.
  - Pipe has been activated by "MLCamSwitch" (→ p. 281).
- PLCopen – Selected for use by "MC_CamIn" (→ p. 476) and has an active move.

There are a number of ways to change an in-use profile to one that is not in-use (deactivated):

- Pipe Network – Perform a "MLCamSwitch" (→ p. 281) on an active Pipe to a different Profile or deactivate the pipe.
- PLCopen – Whenever the active profile move is halted or aborted, the profile is no longer in use.
  - "MC_CamOut" (→ p. 484) is one way of aborting the profile move.
  - Any PLCopen motion command that aborts a profile move deactivates a profile.

NOTE

Any profile ID created by "MC_CamTblSelect" (→ p. 490) from the specified ProfileID is destroyed and must be recreated when this FB is completed.
All derived profile ID's created by the **MC_CamTblSelect** FB **must not** be in use by the PLCopen motion engine for this function to succeed.

IMPORTANT

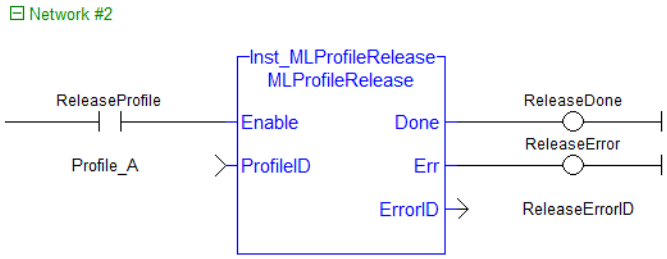
Loading a Profile Editor-generated profile into a ProfileID released by **MLProfileRelease** should be done with care.
The **MLProfileInit ()** function call can take in excess of 4 milliseconds to execute.
Application execution is suspended during this time until the function call is completed.

4.3.6.4.4.2 Error Codes

ErrorID	Description
106	Invalid ProfileID. This can occur if the ProfileID: • Does not exist. • Has not been created yet. • The ProfileID is not a profile.
108	Profile cannot be released because it is in use by the motion engine or currently selected by an active CAM block.

4.3.6.4.5 FBD Language Example

4.3.6.4.6 FFLD Language Example



4.3.6.4.7 IL Language Example

Not available.

4.3.6.4.8 ST Language Example

```
//Release a Cam Profile
Inst_MLProfileRelease( Profile_A , 'Profile_A.5op');

If Inst_MLProfileRelease.Done THEN
   // Do Something
ELSIF Inst_MLProfileRelease.Err THEN
   // Handle Error
END_IF;
```

See Also

- "MC_CamIn" (→ p. 476)
- "MC_CamOut" (→ p. 484)
- "MC_CamTblSelect" (→ p. 490)
- "MLCamInit" (→ p. 280)
- "MLProfileBuild" (→ p. 534)

- "MLProfileCreate" (→ p. 542)
- "MLProfileInit" (→ p. 544)

4.3.7 Motion / Common - Motion

Name	Description
MLMotionCycleTime	Returns the Motion Base Cycle time in seconds.
MLMotionInit	Initializes the motion library.
MLMotionRstErr	Clears motion engine errors, motion bus driver errors, and EtherCAT network errors.
MLMotionStart	Starts the motion engine, motion bus driver, clears the EtherCAT diagnostic registers of all nodes, and initializes EtherCAT network to operational mode.
MLMotionStatus	Returns the status of the motion engine 0 (zero).
MLMotionStop	Stops the motion bus driver, motion engine, and EtherCAT network operation, resulting in the EtherCAT network transitioning to the Init state.

4.3.7.1 State Machine

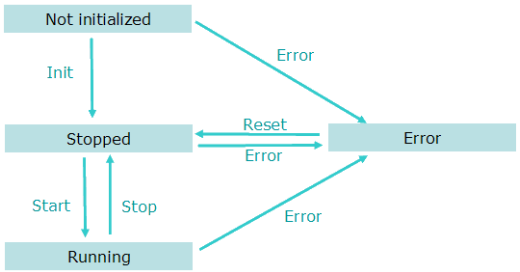


Figure 4-107: Motion State Machine

The Motion State Machine is driven by the IEC 61131-3 application with the help of the "Motion / Common - Motion" (→ p. 548) function blocks.

Each arrow represents a transition from one State to another one.

4.3.7.2 MLMotionCycleTime

PLCopen ✓ Pipe Network ✓

Function - Returns the Motion Base Cycle time in seconds.

4.3.7.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL				Enable execution of the function.

4.3.7.2.2 Outputs

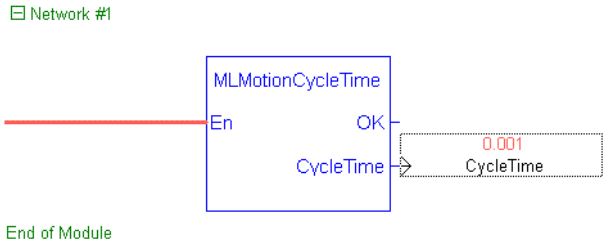
Output	Data Type	Range	Unit	Description
OK	BOOL			If high, successful completion.
CycleTime	LREAL			PLC cycle period in seconds.

4.3.7.2.3 Remarks

None

4.3.7.2.4 FBD Language Example

4.3.7.2.5 FFLD Language Example



4.3.7.2.6 IL Language Example

Not available.

4.3.7.2.7 ST Language Example

```
//Read EtherCAT cycle rate in seconds
CycleTime := MLMotionCycleTime();
```

4.3.7.3 MLMotionInit

PLCopen ✓ Pipe Network ✓

Function - Initializes the motion library.

4.3.7.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	TRUE, FALSE	N/A	FALSE	Enable function.

BasePeriod	LREAL	Minimum: 125	Microseconds	1000	Duration of one motion cycle.
					• With the EtherCAT Motion Bus:
					  • The EtherCAT cycle time overrides the **BasePeriod** argument (the cycle time is defined in the EtherCAT Master Settings tab).
					  • The EtherCAT cycle time becomes the base cycle time for the Motion Engine.
					• Without the EtherCAT Motion Bus:
					  • The **BasePeriod** argument establishes the base cycle time (in microseconds) for the Motion Engine when running simulations

4.3.7.3.2 Outputs

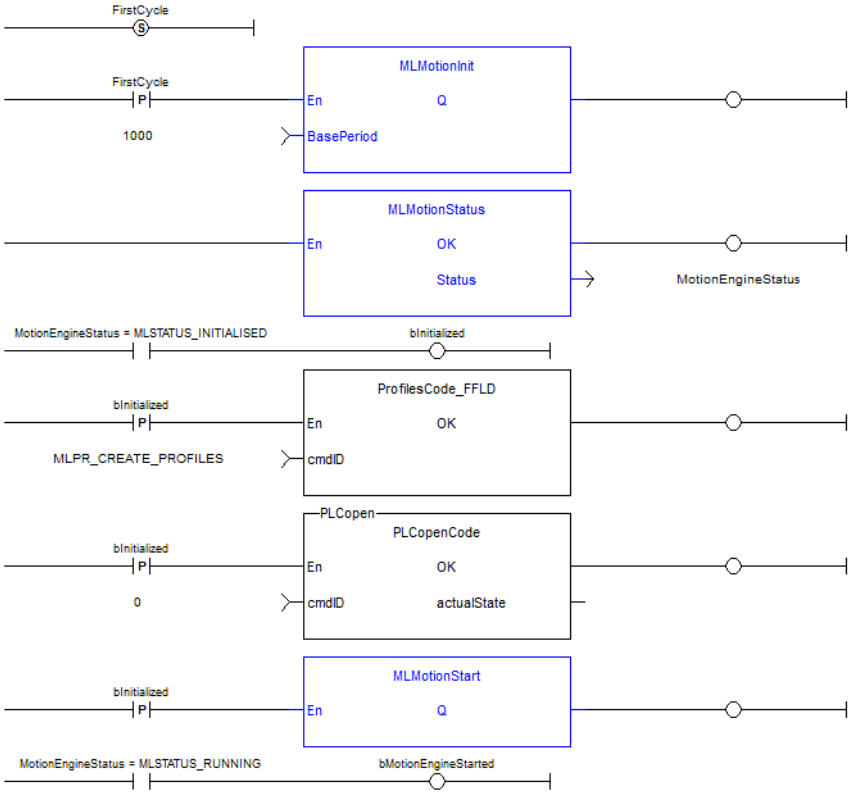
Output	Data Type	Range	Unit	Description
Q	BOOL	TRUE, FALSE	N/A	• Output is set to high with initial successful execution. • It remains high until the function does NOT execute successfully.

4.3.7.3.3 Remarks

- Must be called before any other Motion Library function.
- Returns TRUE if the function succeeded.

4.3.7.3.4 FBD Language Example

4.3.7.3.5 FFLD Language Example



4.3.7.3.6 IL Language Example

Not available.

4.3.7.3.7 ST Language Example

```
//Initialization code to start EtherCAT network.
//First initialize network with MLMotionInit command
//Then wait for command to finish by monitoring MLMotionStatus output
//Once initialized, create any cam profiles and PLCopen or Pipenetwork
devices
//Then call MLMotionStart and monitor MLMotionStatus again before beginning
rest of program
FirstCycle := TRUE;
```

```
On FirstCycle DO //Initialize the motion engine
   MLMotionInit( 1000);
END_DO;

MotionEngineStatus := MLMotionStatus();//Check the current status of the
motion engine
//Once motion engine is initialized, create CAM profiles and defined Axis,
then start the motion engine
ON MotionEngineStatus = MLSTATUS_INITIALISED DO
   Profiles( MLPR_CREATE_PROFILES );
   PLCopen( 0 );
   MLMotionStart();
END_DO;

IF MotionEngineStatus = MLSTATUS_RUNNING THEN
   bMotionEngineStarted := TRUE;
ELSE
   bMotionEngineStarted := FALSE;
END_IF;
```

4.3.7.4 MLMotionRstErr

PLCopen ✓ Pipe Network ✓

Function - Clears motion engine errors, motion bus driver errors, and EtherCAT network errors.

4.3.7.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description

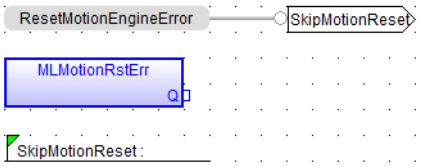
4.3.7.4.2 Outputs

Output	Data Type	Range	Unit	Description
	BOOL			

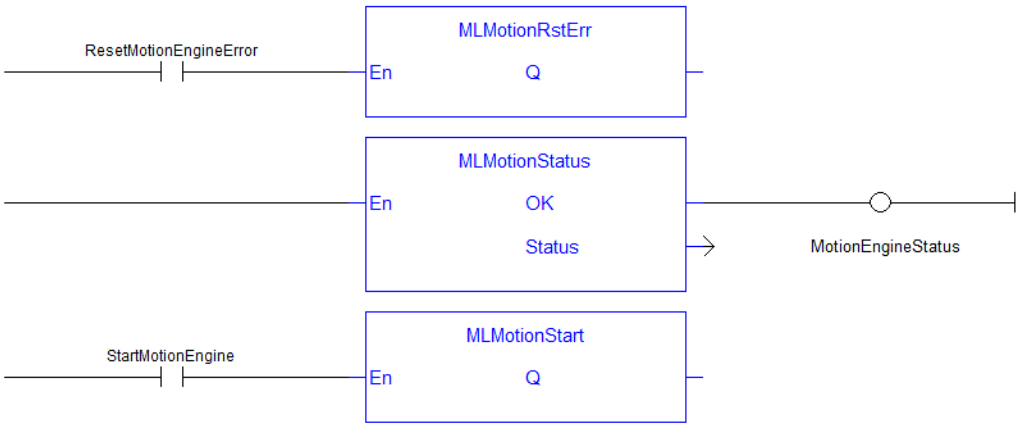
4.3.7.4.3 Remarks

- Re-initializes the motion engine after a motion error.
- **MLMotionRstErr** returns the motion engine status to the Stopped state if an error condition was cleared successfully.
- Motion errors are for example communication errors of the motion bus.
- Returns TRUE if the function succeeded.

4.3.7.4.4 FBD Language Example



4.3.7.4.5 FFLD Language Example



4.3.7.4.6 IL Language Example

Not available.

4.3.7.4.7 ST Language Example

```
//Reset and restart motion engine
//Done to restart EtherCAT after controller error such as
//E30 or E33 that stops network communication
//First have to reset error, then start network again

ON ResetMotionEngineError DO
MLMotionRstErr();
END_DO;

MotionEngineStatus:= MLMotionStatus();

ON StartMotionEngine DO
   MLMotionStart();
END_DO;
```

See Also

- "MLMotionStart" (→ p. 552)
- "MLMotionStatus" (→ p. 554)
- "MLMotionStop" (→ p. 556)

4.3.7.5 MLMotionStart

PLCopen ✓ Pipe Network ✓

Function - Starts the motion engine, motion bus driver, clears the EtherCAT diagnostic registers of all nodes, and initializes EtherCAT network to operational mode.

4.3.7.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description

4.3.7.5.2 Outputs

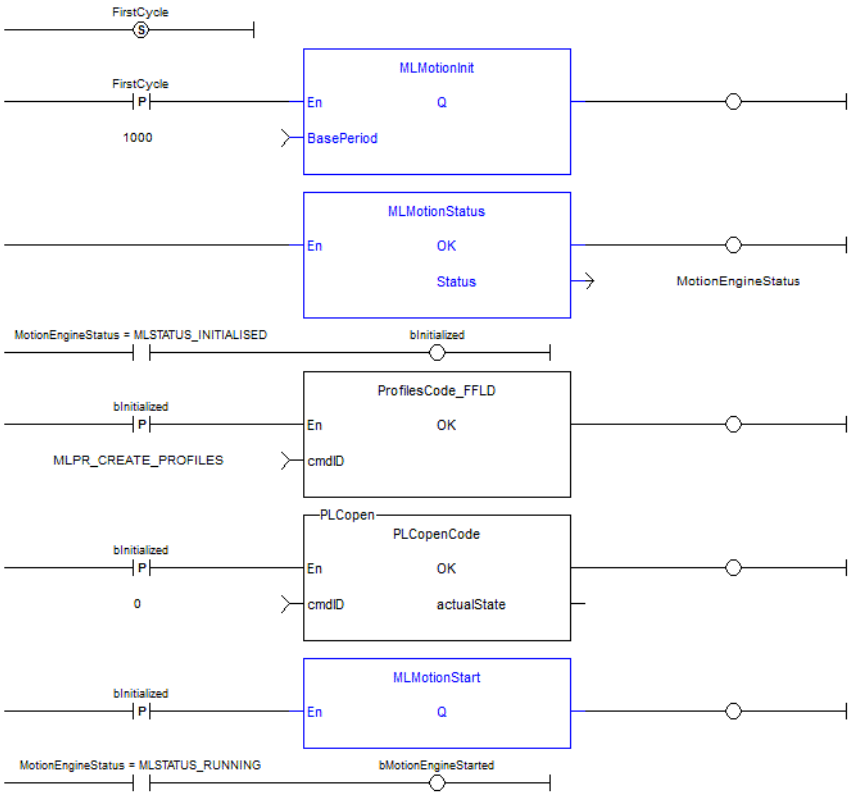
Output	Data Type	Range	Unit	Description
	BOOL			

4.3.7.5.3 Remarks

- Applicable to PLCopen and Pipe Network motion engines.
- MLMotionStart does not clear any pre-existing error conditions.
- Returns TRUE if the function succeeded.
- Returns FALSE if the motion engine is in the Error state.

4.3.7.5.4 FBD Language Example

4.3.7.5.5 FFLD Language Example



4.3.7.5.6 IL Language Example

Not available.

4.3.7.5.7 ST Language Example

```
//Initialization code to start EtherCAT network.
//First initialize network with MLMotionInit command
//Then wait for command to finish by monitoring MLMotionStatus output
//Once initialized, create any cam profiles and PLCopen or Pipenetwork
devices
//Then call MLMotionStart and monitor MLMotionStatus again before beginning
rest of program
FirstCycle := TRUE;

On FirstCycle DO //Initialize the motion engine
   MLMotionInit( 1000);
END_DO;

MotionEngineStatus := MLMotionStatus();//Check the current status of the
motion engine
//Once motion engine is initialized, create CAM profiles and defined Axis,
then start the motion engine
ON MotionEngineStatus = MLSTATUS_INITIALISED DO
   Profiles( MLPR_CREATE_PROFILES );
   PLCopen( 0 );
   MLMotionStart();
END_DO;

IF MotionEngineStatus = MLSTATUS_RUNNING THEN
   bMotionEngineStarted := TRUE;
ELSE
   bMotionEngineStarted := FALSE;
END_IF;
```

See Also

- "MLMotionRstErr" (→ p. 551)
- "MLMotionStatus" (→ p. 554)
- "MLMotionStop" (→ p. 556)

4.3.7.6 MLMotionStatus

PLCopen ✓ Pipe Network ✓

Function - Returns the status of the motion engine 0 (zero).

4.3.7.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Enable function.

4.3.7.6.2 Outputs

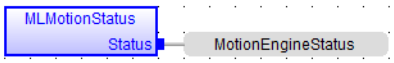
Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	• Indicates the initialization is complete. •
Status	DINT	0, 3	N/A	Status of the Motion Engine. Based on the Internal Defines, the status is one of these: • #define MLSTATUS_NOT_INITIALISED 0 (*Motion not initialized*) • #define MLSTATUS_RUNNING 1 (*Motion is running*) • #define MLSTATUS_STOPPED 2 (*Motion is stopped*) • #define MLSTATUS_ERROR 3 (*Motion is in error*)

4.3.7.6.3 Remarks

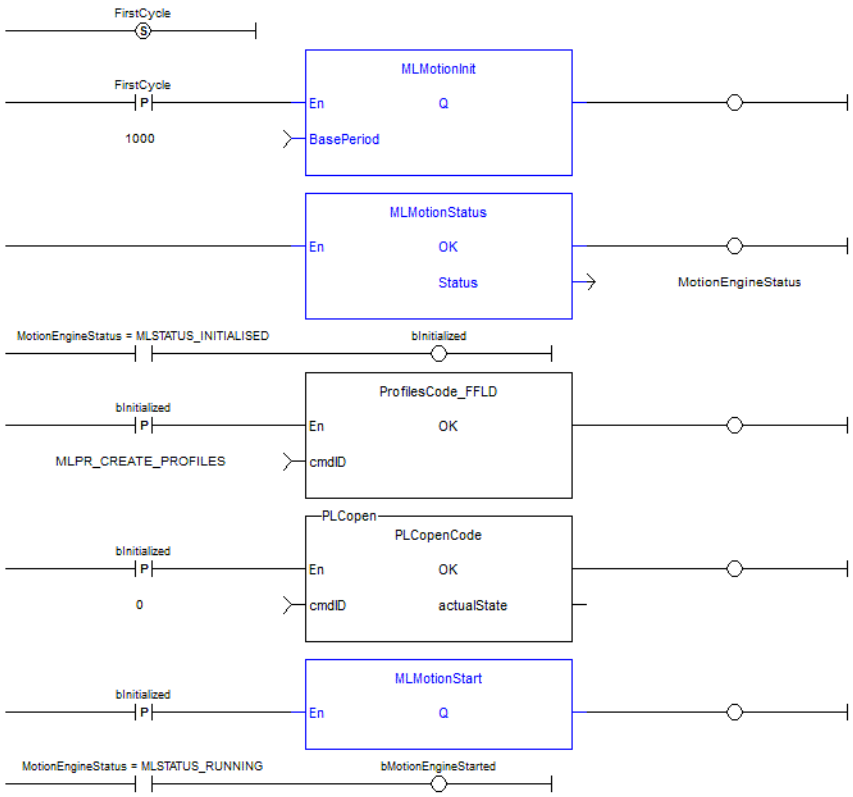
NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

4.3.7.6.4 FBD Language Example



4.3.7.6.5 FFLD Language Example



4.3.7.6.6 IL Language Example

Not available.

4.3.7.6.7 ST Language Example

```
//Initialization code to start EtherCAT network.
//First initialize network with MLMotionInit command
//Then wait for command to finish by monitoring MLMotionStatus output
//Once initialized, create any cam profiles and PLCopen or Pipenetwork
devices
//Then call MLMotionStart and monitor MLMotionStatus again before beginning
rest of program
FirstCycle := TRUE;

On FirstCycle DO //Initialize the motion engine
MLMotionInit( 1000);
END_DO;

MotionEngineStatus := MLMotionStatus();//Check the current status of the
motion engine
//Once motion engine is initialized, create CAM profiles and defined Axis,
then start the motion engine
ON MotionEngineStatus = MLSTATUS_INITIALISED DO
Profiles( MLPR_CREATE_PROFILES );
PLCopen( 0 );
MLMotionStart();
END_DO;

IF MotionEngineStatus = MLSTATUS_RUNNING THEN
bMotionEngineStarted := TRUE;
ELSE
bMotionEngineStarted := FALSE;
END_IF;
```

4.3.7.7 MLMotionStop

PLCopen ✓ Pipe Network ✓

Function - Stops the motion bus driver, motion engine, and EtherCAT network operation, resulting in the EtherCAT network transitioning to the Init state.

4.3.7.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description

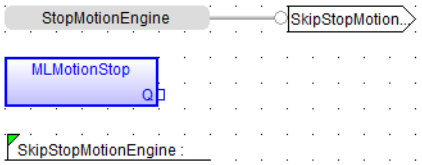
4.3.7.7.2 Outputs

Output	Data Type	Range	Unit	Description
	BOOL			

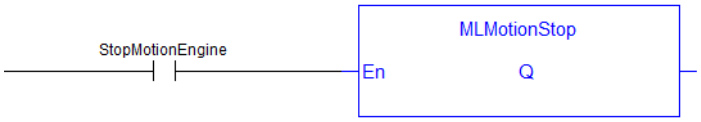
4.3.7.7.3 Remarks

- Results in the EtherCAT network transitioning to the Init state.
- Returns TRUE if the function succeeded.

4.3.7.7.4 FBD Language Example



4.3.7.7.5 FFLD Language Example



4.3.7.7.6 IL Language Example

Not available.

4.3.7.7.7 ST Language Example

```
//Stop the EtherCAT network
ON StopMotionEngine DO
   MLMotionStop();
END_DO;
```

See Also

- "MLMotionStart" (→ p. 552)
- "MLMotionRstErr" (→ p. 551)
- "MLMotionStatus" (→ p. 554)

4.3.8 Motion / Common - Coordinated Motion

These tables list the Coordinated Motion function blocks and function blocks:

- "Group Control" (→ p. 557)
- "Info" (→ p. 558)
- "Motion" (→ p. 558)
- "Reference" (→ p. 559)

4.3.8.1 Group Control

See "Coordinated Motion - Group Control Library" (→ p. 559) for more information.

Name	Description
MC_AddAxisToGrp	Adds an axis to an axes group.
MC_ CreateAxesGrp	Create an axis group for coordinated motion.
MC_GrpDisable	Changes the state of a group to GroupDisabled.
MC_GrpEnable	Changes the state of a group from GroupDisabled to GroupStandby.

Name	Description
MC_ GrpReadBoolPar	Reads a value from the specified Boolean group parameter.
MC_ GrpReadParam	Reads a value from the specified group parameter.
MC_GrpReset	Makes the transition from the state **GroupErrorStop** to **GroupStandby** by resetting all internal group-related errors.
MC_GrpStop	Performs a controlled motion stop of all axes in the group.
MC_ GrpWriteBoolPar	Writes a value to the specified Boolean group parameter.
MC_ GrpWriteParam	Writes a value to the specified group parameter.
MC_InitAxesGrp	Initializes the kinematic limits for the axis group.
MC_ RemAxisFromGrp	Removes an individual axis from an axis group.
MC_SetKinTra	Sets the kinematic transform between the Machine Coordinate System and the Axes Coordinate System.
MC_ UngroupAllAxes	Removes all axes from an axes group.

4.3.8.2 Info

See "Motion / Common - Coordinated Motion - Info" (→ p. 591) for more information.

Name	Description
MC_GrpReadActAcc	Reads the actual acceleration of the group and the axes in the group.
MC_GrpReadActPos	Reads the actual position of the axes in the group.
MC_GrpReadActVel	Reads the actual velocity of the group and the axes in the group.
MC_GrpReadCmdPos	Reads the command position of the axes in the group.
MC_GrpReadCmdVel	Reads the command velocity of the axes in the group and the path velocity.
MC_GrpReadError	Reads the Group ErrorID in state ERRORSTOP.
MC_GrpReadStatus	Returns the status of an axes group.

4.3.8.3 Motion

See "Motion / Common - Coordinated Motion - Motion" (→ p. 609) for more information.

Name	Description
MC_ AxisSetDefaults	Sets the default kinematic parameters for an axis.
MC_ ErrorDescription	Converts the PLCopen error IDs into message strings.
MC_GrpHalt	Performs a controlled motion stop of all axes in the group.
MC_ GrpSetOverride	Sets the velocity factor that is multiplied to the commanded velocity of all axes in the group.

Name	Description
MC_ MoveCircAbs	Commands interpolated circular movement on an axes group to the specified absolute positions in the coordinate system as specified by the CoordSystem argument.
MC_MoveCircRel	Commands interpolated circular movement on an axes group to the specified relative positions in the coordinate system as specified by the CoordSystem argument.
MC_MoveDirAbs	Commands movement of an axes group to an absolute position regardless of path.
MC_MoveDirRel	Commands movement of an axes group to a relative position regardless of path.
MC_MoveLinAbs	Commands interpolated linear movement on an axes group to the specified absolute positions.
MC_MoveLinRel	Commands interpolated linear movement on an axes group to the specified relative positions.

4.3.8.4 Reference

See "Coordinated Motion - Reference Library" (→ p. 647) for more information.

Name	Description
MC_ GrpSetPos	Sets the axis command position for all of the axes in an axes group to the positions specified in the `Position` input.

4.3.8.5 Coordinated Motion - Group Control Library

Name	Description
MC_AddAxisToGrp	Adds an axis to an axes group.
MC_ CreateAxesGrp	Create an axis group for coordinated motion.
MC_GrpDisable	Changes the state of a group to GroupDisabled.
MC_GrpEnable	Changes the state of a group from GroupDisabled to GroupStandby.
MC_ GrpReadBoolPar	Reads a value from the specified Boolean group parameter.
MC_ GrpReadParam	Reads a value from the specified group parameter.
MC_GrpReset	Makes the transition from the state **GroupErrorStop** to **GroupStandby** by resetting all internal group-related errors.
MC_GrpStop	Performs a controlled motion stop of all axes in the group.
MC_ GrpWriteBoolPar	Writes a value to the specified Boolean group parameter.
MC_ GrpWriteParam	Writes a value to the specified group parameter.
MC_InitAxesGrp	Initializes the kinematic limits for the axis group.
MC_ RemAxisFromGrp	Removes an individual axis from an axis group.
MC_SetKinTra	Sets the kinematic transform between the Machine Coordinate System and the Axes Coordinate System.

Name	Description
MC_ UngroupAllAxes	Removes all axes from an axes group.

4.3.8.5.1 MC_AddAxisToGrp

PLCopen ✓ Pipe Network ✓

Function Block - Adds an axis to an axes group.

4.3.8.5.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, the axis is added to the group.
AxesGroup	AXES_ GROUP_ REF	No range	N/A	No default	Reference to an axes group.
Axis	AXIS_REF	No range	N/A	No default	Reference to the axis to be added. An axes group cannot contain more than one instance of an axis.
IdentInGroup	UINT	0, MaxNumberOfAxes - 1	N/A	No default	The zero-based index of the axis in the group. • The axis slot in the group cannot be occupied by another axis. • The index must be less than the maximum number of axes the group can contain. • `MaxNumberOfAxes` is a property of the axes group and is set when the group is created. • To remove an axis from a group see "MC_ RemAxisFromGrp" (→ p. 579).

4.3.8.5.1.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	If TRUE, the command completed successfully.
Error	BOOL	No range	N/A	If TRUE, an error has occurred.
ErrorID	INT	No range	N/A	Indicates the error if the **Error** output is set to TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.8.5.1.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

TIP

- An axes group cannot contain more than one instance of an axis.
- Two active groups cannot contain the same axis. An "active" group is one in any state other than GroupDisabled.

- This function or function block does not generate any motion.
- See Coordinated Motion, the top-level topic for Coordinated Motion.
- Both the axis and the axes group must be created prior to calling this function block.
  - See "Example: Valid Setup" (→ p. 563) and Create a PLCopen Axis.
- The **IdentInGroup** input specifies the index of the axis in the group.
  - Axes do not need to be added in sequential order and gaps are acceptable.
  - Gaps are ignored when the group is used.
- The group must be in either the **GroupStandby** or **GroupDisabled** state when the axis is added.
  - The state of the group can be read with "MC_GrpReadStatus" (→ p. 606).
  - This implies that the group cannot be moving when the axis is added.

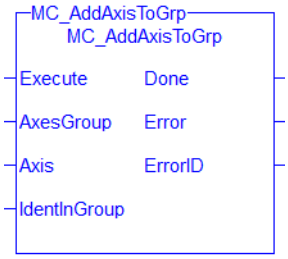
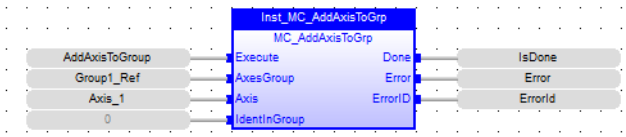
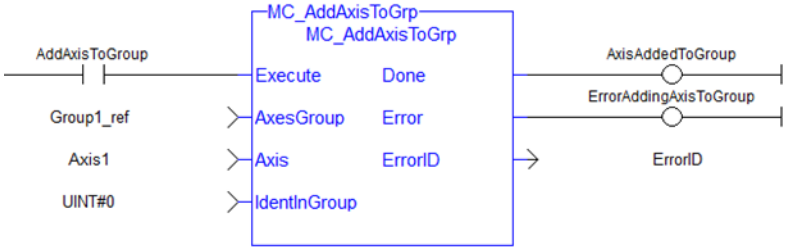


Figure 4-108: MC_AddAxisToGrp

4.3.8.5.1.4 FBD Language Example



4.3.8.5.1.5 FFLD Language Example



4.3.8.5.1.6 IL Language Example

```
            BEGIN_IL
     CAL Inst_MC_AddAxisToGrp( AddAxisToGrp, Group1_ref, Axis_1, 0 )
END_IL
```

4.3.8.5.1.7 ST Language Example

```
            (*MC_AddAxisToGrp ST example *)
Inst_MC_AddAxisToGrp (AddAxisToGrp, Group1_ref, Axis_1, 0);
```

See Also

- "Example: Valid Setup" (→ p. 563)
- "MC_GrpReadStatus" (→ p. 606)
- "MC_RemAxisFromGrp" (→ p. 579)
- "MC_UngroupAllAxes" (→ p. 583)
- "MC_ErrorDescription" (→ p. 533)

4.3.8.5.2 MC_CreateAxesGrp

PLCopen ✓ Pipe Network ✓

Function Block - Create an axis group for coordinated motion.

4.3.8.5.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, this function block creates a coordinated motion axes group.
Name	STRING	1 to 64 characters	N/A	No default	Axes Group Name The string length is limited to 64 characters for optimal controller performance.

UpdateRate	UINT	3, 9	N/A	No default	Update rate of the axes group.
					• The group update rate is the same as the **Base Period** specified in "MLMotionInit" (→ p. 549).
					• The update rate runs at the Base Period if it is a smaller time than the Base Period.
					• (0, 1, and 2 are reserved for future enhancements).
					  • 3 = 125 µsec
					  • 4 = 250 µsec
					  • 5 = 500 µsec
					  • 6 = 1 msec
					  • 7 = 2 msec
					  • 8 = 4 msec
					  • 9 = 8 msec

MaxNumberOfAxes	UINT	2, 256	N/A	No default	The maximum number of axes that can be controlled by the group.
AxesGroupRef	AXES_GROUP_REF	No range	N/A	No default	The axes group reference variable to be initialized with a reference to the new axes group.

4.3.8.5.2.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.

Output	Data Type	Range	Unit	Description
Error	BOOL			If TRUE, an error has occurred.
ErrorID	INT			Indicates the error if the **Error** output is set to TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.8.5.2.3 Remarks

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- Calls to this function block are automatically generated when the application is compiled.
  - Users should not manually call this function block.
- More than one axes group may be created and be active at the same time but each axis can only be a part of one group at a time.

4.3.9 Example: Valid Setup

```
AxesGroup1:   Axis0, Axis1, Axis2
AxesGroup2:   Axis3, Axis4
```

4.3.10 Example: Invalid Setup

```
AxesGroup1:   Axis0, Axis1, Axis2
AxesGroup2:   Axis2, Axis3, Axis4
```

The invalid setup is not allowed because Axis2 would be a part of two axes groups at the same time.

If an axis needs to be in more than one group, it can be removed from one and then added to another group. This is done using "MC_RemAxisFromGrp" (→ p. 579) and "MC_AddAxisToGrp" (→ p. 560).

IMPORTANT

MC_CreateAxesGrp must be called between "MLMotionInit" (→ p. 549) and "MLMotionStart" (→ p. 552).

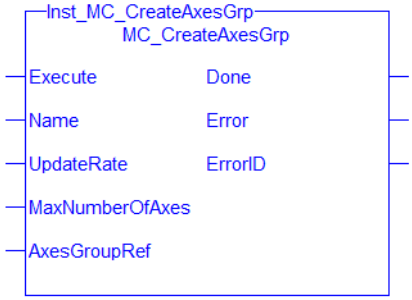
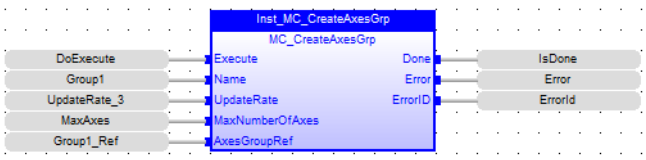
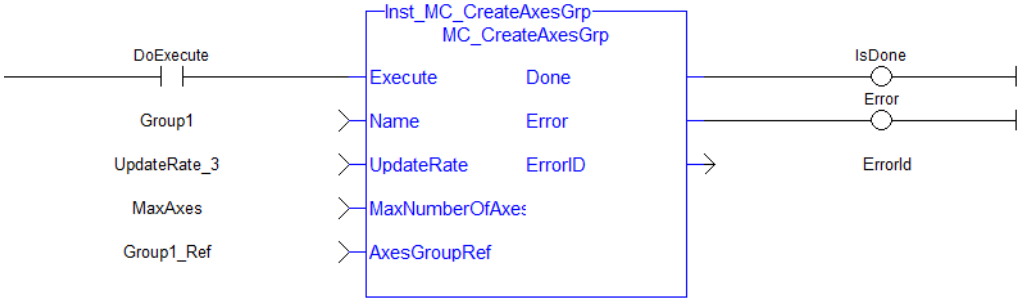


Figure 4-109: MC_CreateAxesGrp

4.3.10.0.0.1 FBD Language Example



4.3.10.0.0.2 FFLD Language Example



4.3.10.0.0.3 IL Language Example

```
            BEGIN_IL
     CAL Inst_MC_CreateAxesGrp1(DoExecute, 'Group1', UpdateRate_3, MaxAxes,
Group1_Ref)
END_IL
```

4.3.10.0.0.4 ST Language Example

```
Inst_MC_CreateAxesGrp( DoExecute, 'Group1', UpdateRate_3, MaxAxes, Group1_
Ref);
```

See Also

- "MC_InitAxesGrp" (→ p. 577)
- "MC_ErrorDescription" (→ p. 533)

4.3.10.0.1 MC_GrpDisable

PLCopen ✓ Pipe Network ✓

Function Block - Changes the state of a group to GroupDisabled.

4.3.10.0.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to disable the axis group.
AxesGroup	AXES_ GROUP_REF	No range	N/A	No default	The axis group to be disabled.

4.3.10.0.1.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	If TRUE, the command completed successfully.
Error	BOOL	No range	N/A	If TRUE, an error has occurred.
ErrorID	INT	No range	N/A	Indicates the error if the **Error** output is set to TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.10.0.1.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- If the group is already in **GroupDisabled**, then **MC_GrpDisable** does nothing.
- This function block can be issued in the group states: (**GroupDisabled**, **GroupStandby**, or **GroupErrorStop**).
- See Group State Diagrams for more information.

⚠CAUTION

MC_GrpDisable fails if the group is in any state other than GroupStandby or GroupDisabled.

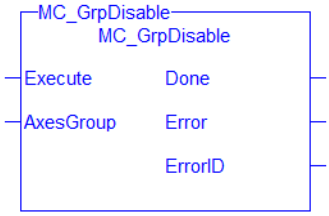
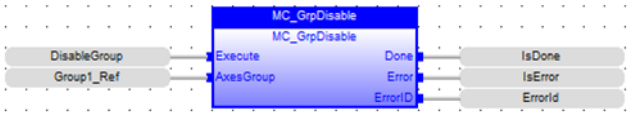
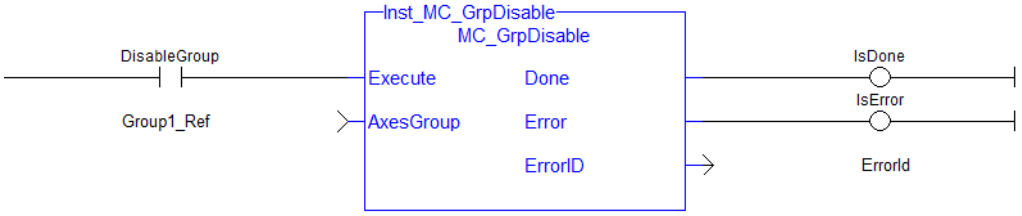


Figure 4-110: MC_GrpDisable

4.3.10.0.1.4 FBD Language Example



4.3.10.0.1.5 FFLD Language Example



4.3.10.0.1.6 IL Language Example

```
            BEGIN_IL

     CAL Inst_MC_GrpDisable( DisableGroup, Group1_Ref )
END_IL
```

4.3.10.0.1.7 ST Language Example

```
(* Inst_MC_GrpDisableST example *)
Inst_MC_GrpDisable( DisableGroup, Group1_Ref );
```

See Also

- "MC_ErrorDescription" (→ p. 533)
- "MC_GrpEnable" (→ p. 565)

4.3.10.0.2 MC_GrpEnable

PLCopen ✓ Pipe Network ✓

Function Block - Changes the state of a group from GroupDisabled to GroupStandby.

4.3.10.0.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to enable the axis group.
AxesGroup	AXES_GROUP_REF	N/A	N/A	No default	The axis group to be enabled.

4.3.10.0.2.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	N/A	N/A	If TRUE, the command completed successfully.
Error	BOOL	N/A	N/A	If TRUE, an error has occurred.
ErrorID	INT	N/A	N/A	Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.10.0.2.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- The group must be in **GroupStandby** to perform motion.
  - If the group is already in **GroupStandby**, then **MC_GrpEnable** does nothing.
- MC_GrpEnable fails under these conditions:
  - It contains no axes.
  - The group is not in GroupDisabled or GroupStandby.
  - One or more axes in the group are in another group that is not in GroupDisabled.
  - See Group State Diagrams for more information.

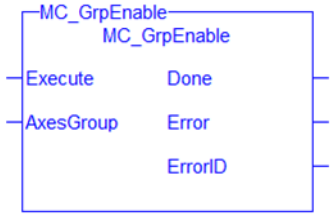
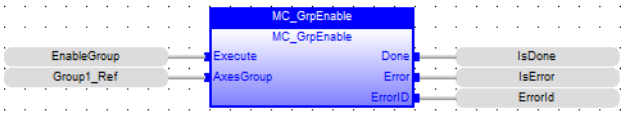
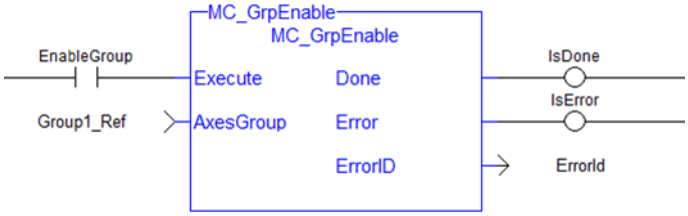


Figure 4-111: MC_GrpEnable

4.3.10.0.2.4 FBD Language Example



4.3.10.0.2.5 FFLD Language Example



4.3.10.0.2.6 IL Language Example

```
            BEGIN_IL

     CAL Inst_MC_GrpEnable( EnableGroup, Group1_Ref )
END_IL
```

4.3.10.0.2.7 ST Language Example

```
(* Inst_MC_GrpEnableST example *)
Inst_MC_GrpEnable( EnableGroup, Group1_Ref );
```

See Also

- "MC_ErrorDescription" (→ p. 533)
- "MC_GrpDisable" (→ p. 564)

4.3.10.0.3 MC_GrpReadBoolPar

PLCopen ✓ Pipe Network ✓

Function Block - Reads a value from the specified Boolean group parameter.

4.3.10.0.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	If TRUE, then request to read a value from the specified Boolean group parameter.
AxesGroupRef	AXES_GROUP_REF	N/A	N/A	No default	The axis group the Boolean parameter value is read from.

ParamNum	UINT	1000, 1001	UINT	No default	ParamNum can be one of these two defined Booleans:
					• IGNORE_AXIS_ESTOP:
					  • ID = 1000
					  • The value read is either TRUE or False as set by the MC_GrpWriteBoolPar function block.
					• AXIS_ESTOP_ACTIVE:
					  • ID = 1001
					  • This Read-only parameter is asserted TRUE whenever an axis in the group is experiencing an Axis Estop Error.
					  • When there are no Axis Estop Errors present on the axes in a group, this parameter will be set to FALSE.

4.3.10.0.3.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	If TRUE, the command completed successfully.
Error	BOOL	No range	N/A	If TRUE, an error has occurred.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.
ParamValue	BOOL	TRUE, FALSE	N/A	TRUE or FALSE.

4.3.10.0.3.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- See Recovery of the System State After an Axis Error for more information.
- MC_GrpReadBoolPar(Axesgroup_Ref GroupID, Uint BoolID) where BoolID can be one of these two currently defined Booleans:
  - IGNORE_AXIS_ESTOP:
    - ID = 1000
    - The value read is either TRUE or False as set by the MC_GrpWriteBoolPar function block.
  - AXIS_ESTOP_ACTIVE:
    - ID = 1001
    - This Read-only parameter is asserted TRUE whenever an axis in the group is experiencing an Axis Estop Error.
    - When there are no Axis Estop Errors present on the axes in a group, this parameter is set to FALSE.

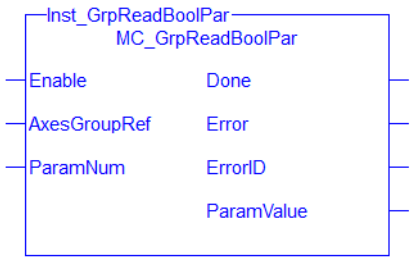
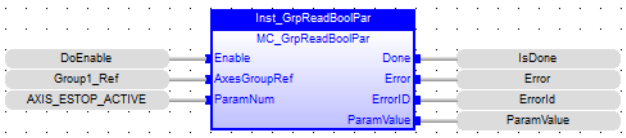
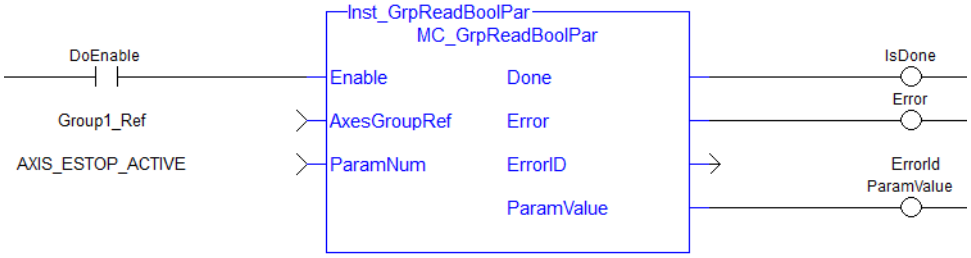


Figure 4-112: MC_GrpReadBoolPar

4.3.10.0.3.4 FBD Language Example



4.3.10.0.3.5 FFLD Language Example



4.3.10.0.3.6 IL Language Example

```
            BEGIN_IL
     Cal Inst_GrpReadBoolPar( DoEnable, Group1_Ref, AXIS_ESTOP_ACTIVE )
END_IL
```

4.3.10.0.3.7 ST Language Example

```
Inst_GrpReadBoolPar( DoEnable, Group1_Ref, AXIS_ESTOP_ACTIVE );
```

See Also

- "MC_ErrorDescription" (→ p. 533)
- "MC_GrpWriteBoolPar" (→ p. 574)

4.3.10.0.4 MC_GrpReadParam

PLCopen ✓ Pipe Network ✓

Function Block - Reads a value from the specified group parameter.

4.3.10.0.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	If TRUE, then request to read a value from the specified Boolean group parameter.
AxesGroupRef	AXES_GROUP_REF	No range	N/A	No default	The axis group the Boolean parameter value is read from.

ParamNum	LREAL	See Axes Group Parameters.	LREAL	No default	Currently, one parameter is supported:
					• MC_GRP_PARAM_CIRCLE_TOLERANCE:
					  • ID = 2000
					  • The value read is the axes group circle construction tolerance.
					  • See Precision Requirements for Circular Move Input Parameters for more information.

4.3.10.0.4.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	If TRUE, the command completed successfully.
Error	BOOL	No range	N/A	If TRUE, an error has occurred.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.
ParamValue	LREAL	No range	N/A	The value of the group parameter.

4.3.10.0.4.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- See Coordinated Motion, the top-level topic for Coordinated Motion.

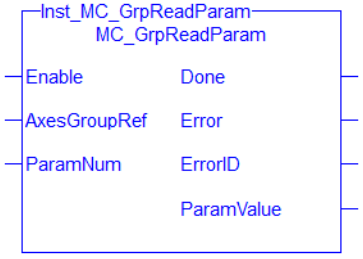
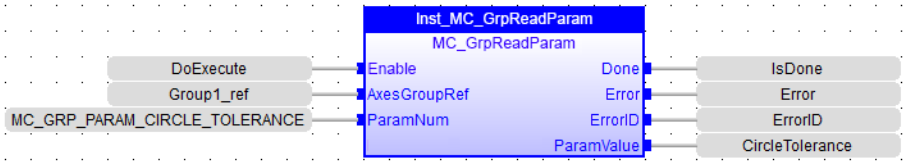
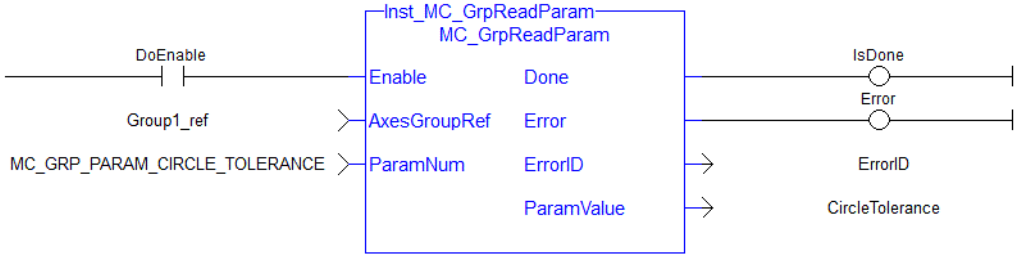


Figure 4-113: MC_GrpReadParam

4.3.10.0.4.4 FBD Language Example



4.3.10.0.4.5 FFLD Language Example



4.3.10.0.4.6 IL Language Example

Not available.

4.3.10.0.4.7 ST Language Example

```
Inst_MC_GrpReadParam( DoEnable, Group1_ref, MC_GRP_PARAM_CIRCLE_TOLERANCE );

CircleTolerance := Inst_MC_GrpReadParam.ParamValue;
```

See Also

- "MC_ErrorDescription" (→ p. 533)
- "MC_GrpWriteParam" (→ p. 576)

4.3.10.0.5 MC_GrpReset

PLCopen ✓ Pipe Network ✓

Function Block - Makes the transition from the state **GroupErrorStop** to **GroupStandby** by resetting all internal group-related errors.

4.3.10.0.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, resets group-related errors and all of the axes in the group.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axes group the axes are reset in.

4.3.10.0.5.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	If TRUE, the command completed successfully.
Busy	BOOL	No range	N/A	If TRUE, the FB is executing.
Error	BOOL	No range	N/A	If TRUE, an error has occurred.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.10.0.5.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- This function or function block does not generate any motion.
- This function block:
  - Does not affect the output of the FB instances.
  - Resets axis errors and drive faults for each axis in the group.

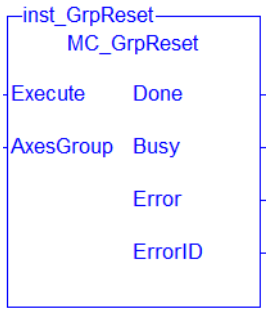
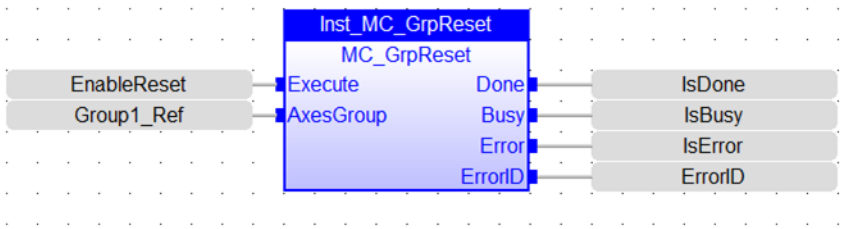
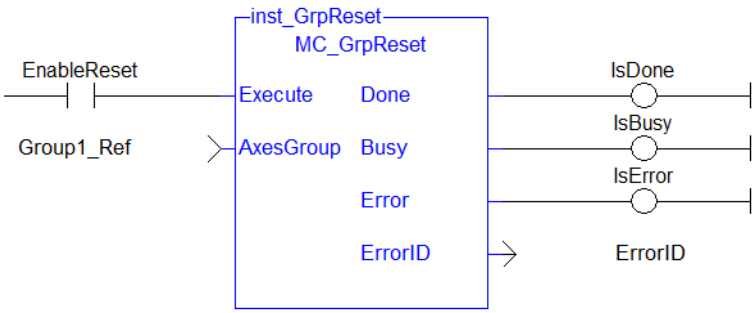


Figure 4-114: MC_GrpReset

4.3.10.0.5.4 FBD Language Example



4.3.10.0.5.5 FFLD Language Example



4.3.10.0.5.6 IL Language Example

```
            BEGIN_IL
     CAL Inst_MC_GrpReset ( EnableReset, Group1_Ref )
END_IL
```

4.3.10.0.5.7 ST Language Example

```
Inst_MC_GrpReset ( EnableReset, Group1_Ref );
```

See Also

- "MC_ErrorDescription" (→ p. 533)
- "MC_GrpReadError" (→ p. 605)
- "MC_GrpReadStatus" (→ p. 606)

4.3.10.0.6 MC_GrpStop

PLCopen ✓ Pipe Network ✓

Function Block - Performs a controlled motion stop of all axes in the group.

4.3.10.0.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, the command to stop all of the axes in the group is initiated.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axes group the axes are stopped in.
Deceleration	LREAL	0 < Deceleration	N/A	No default	The path deceleration rate for all axes in the group. See Limitations on Acceleration and Jerk for more information.
Jerk	LREAL	0 ≤ Jerk	N/A	No default	Not supported. See Limitations on Acceleration and Jerk for more information.

4.3.10.0.6.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Busy	BOOL			TRUE from the moment the EXECUTE input is TRUE until the stop is complete.
Active	BOOL			If TRUE, the stop is still executing.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	INT			Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.10.0.6.3 Remarks

⚠CAUTION

MC_GrpStop can not be aborted.

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- When the path velocity reaches 0 (zero), any queued moves are flushed from the buffer and the Done output is set.
- When both the Done output is TRUE and the application has cleared, the Execute input the state transitions to GroupStandby.
- **MC_GrpStop** does NOT prevent:
  - A single axis from executing.
  - Other Coordinated Motion moves from executing once MC_GrpStop has completed.

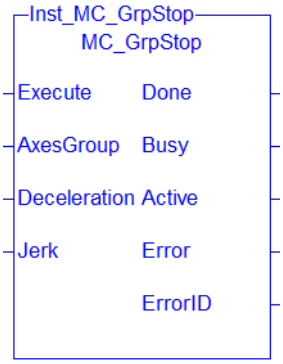
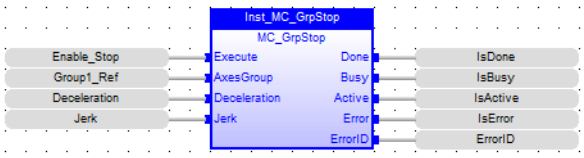
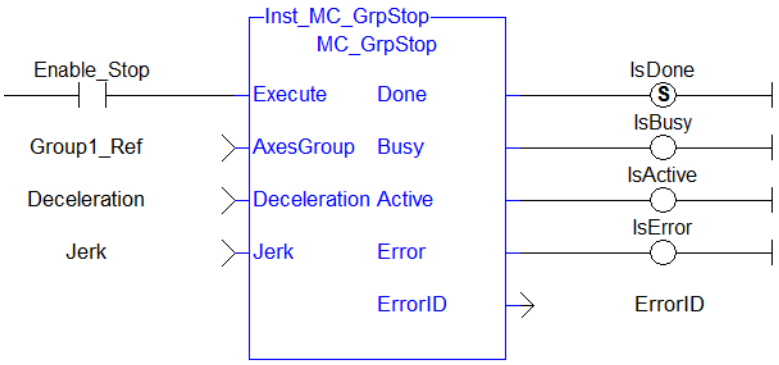


Figure 4-115: MC_GrpStop

4.3.10.0.6.4 FBD Language Example



4.3.10.0.6.5 FFLD Language Example



4.3.10.0.6.6 IL Language Example

```
            BEGIN_IL
     CAL Inst_MC_GrpStop ( EnableStop, Group1_Ref, Deceleration, Jerk )
END_IL
```

4.3.10.0.6.7 ST Language Example

```
Inst_MC_GrpStop ( EnableStop, Group1_Ref, Deceleration, Jerk );
```

4.3.11 Related Functions

- "MC_ErrorDescription" (→ p. 533)
- "MC_GrpHalt" (→ p. 614)

4.3.11.0.1 MC_GrpWriteBoolPar

PLCopen ✓ Pipe Network ✓

Function Block - Writes a value to the specified Boolean group parameter.

4.3.11.0.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to write a value to the specified Boolean group parameter.
AxesGroupRef	AXES_GROUP_REF	No range	N/A	No default	The axis group the Boolean parameter value is written to.
ParamNum	UINT	No range	N/A	No default	The ID number of the Boolean parameter that is to be written. • IGNORE_AXIS_ESTOP • BoolID = 1000
ParamValue	BOOL	0, 1	N/A	No default	TRUE, FALSE

4.3.11.0.1.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Busy	BOOL			If TRUE, the function block is executing.

Output	Data Type	Range	Unit	Description
Error	BOOL			If TRUE, an error has occurred.
ErrorID	INT			Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.11.0.1.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- See Recovery of the System State After an Axis Error for more information.
- IGNORE_AXIS_ESTOP
  - BoolID = 1000.
  - Value can be either TRUE or FALSE.
- Setting this Boolean Parameter to TRUE results in the Coordinated Motion Engine NOT stopping all axes in a group when one of them is stopped due to an Axis Estop Error.
  - Only the axis experiencing the error stops when this Parameter is set to TRUE.
- When this parameter is FALSE (Default), all axes in a group are stopped and the power off request is asserted for each axis.

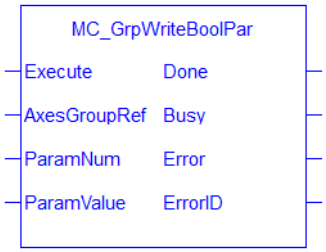
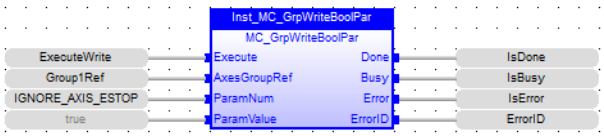
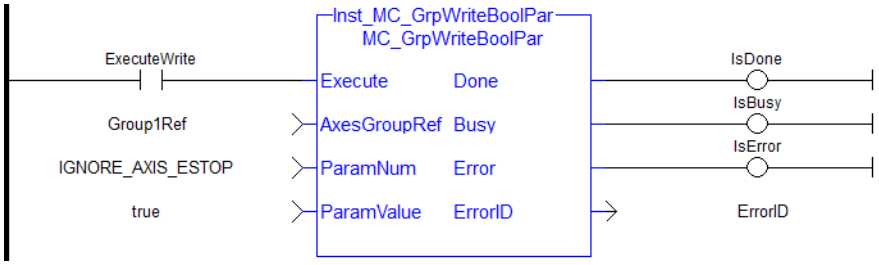


Figure 4-116: MC_GrpWriteBoolPar

4.3.11.0.1.4 FBD Language Example



4.3.11.0.1.5 FFLD Language Example



4.3.11.0.1.6 IL Language Example

```
            BEGIN_IL
     Cal Inst_MC_GrpWriteBoolPar( ExecuteWrite, Group1Ref, IGNORE_AXIS_ESTOP,
true )
END_IL
```

4.3.11.0.1.7 ST Language Example

```
Inst_MC_GrpWriteBoolPar( ExecuteWrite, Group1Ref, IGNORE_AXIS_ESTOP, true );
```

See Also

- "MC_ErrorDescription" (→ p. 533)
- "MC_GrpReadBoolPar" (→ p. 567)"MC_ErrorDescription" (→ p. 533)

4.3.11.0.2 MC_GrpWriteParam

PLCopen ✓ Pipe Network ✓

Function Block - Writes a value to the specified group parameter.

4.3.11.0.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to write a value to the specified Boolean group parameter.
AxesGroupRef	AXES_GROUP_REF	No range	N/A	No default	The axis group the Boolean parameter value is written to.

ParamNum	LREAL	See Axes Group Parameters.	LREAL	No default	Currently, one parameter is supported:
					• MC_GRP_PARAM_CIRCLE_TOLERANCE:
					  • ID = 2000
					  • The value read is the axes group circle construction tolerance.
					  • See Precision Requirements for Circular Move Input Parameters for more information.

ParamValue	LREAL	Parameter dependent.	Parameter dependent.	No default	The new value for the group parameter.

4.3.11.0.2.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Busy	BOOL			If TRUE, the function block is executing.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	INT			Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.11.0.2.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- See Coordinated Motion, the top-level topic for Coordinated Motion.

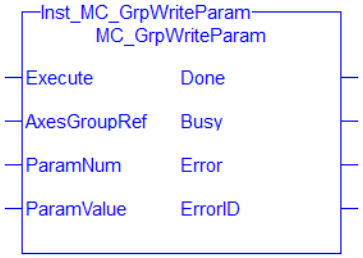
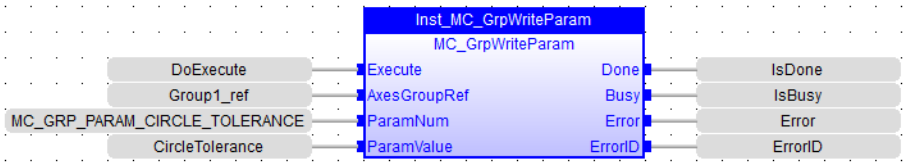
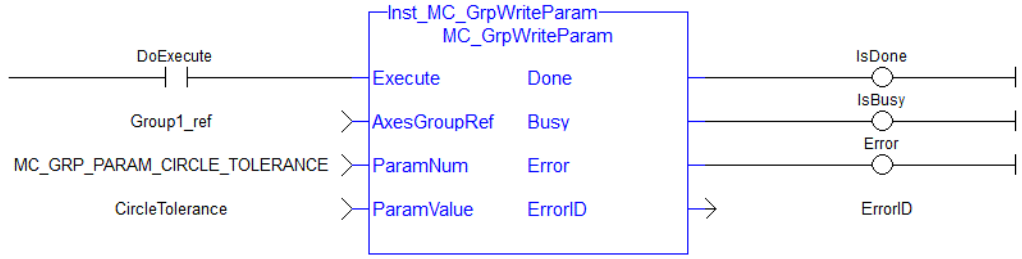


Figure 4-117: MC_GrpWriteParam

4.3.11.0.2.4 FBD Language Example



4.3.11.0.2.5 FFLD Language Example



4.3.11.0.2.6 IL Language Example

Not available.

4.3.11.0.2.7 ST Language Example

```
Inst_MC_GrpWriteParam( DoExecute, Group1_ref, MC_GRP_PARAM_CIRCLE_TOLERANCE,
CircleTolerance );
```

See Also

- "MC_ErrorDescription" (→ p. 533)
- "MC_GrpReadParam" (→ p. 569)

4.3.11.0.3 MC_InitAxesGrp

PLCopen ✓ Pipe Network ✓

Function Block - Initializes the kinematic limits for the axis group.

4.3.11.0.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, this function block initializes the axis group.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axes group to be initialized.
VelocityLimit	LREAL	0 < Velocity < (20 * Acceleration) and 0 < Velocity < (20 * Deceleration) 0 < Velocity < (20 * Acceleration) and 0 < Velocity < (20 * Deceleration)	User unit/sec	No default	Velocity limit. See Limitations on Acceleration and Jerk for more information.
AccelerationLimit	LREAL	0 < Velocity < (20 * Acceleration)	User unit/sec^2	No default	Acceleration limit. See Limitations on Acceleration and Jerk for more information.
DecelerationLimit	LREAL	0 < Velocity < (20 * Deceleration)	User unit/sec^2	No default	Deceleration limit. See Limitations on Acceleration and Jerk for more information.
JerkLimit	LREAL	(Velocity / 20) < Acceleration < (2 * Jerk) and (Velocity / 20) < Deceleration < (2 * Jerk)	User unit/sec^2	No default	Jerk limit. See Limitations on Acceleration and Jerk for more information.

4.3.11.0.3.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	If TRUE, the command completed successfully.
Error	BOOL	No range	N/A	If TRUE, an error has occurred.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.11.0.3.3 Remarks

NOTE

The function block returns an error if the group state is not GroupStandby or GroupDisabled.

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- During a move, the motion engine verifies that the limits are not exceeded.

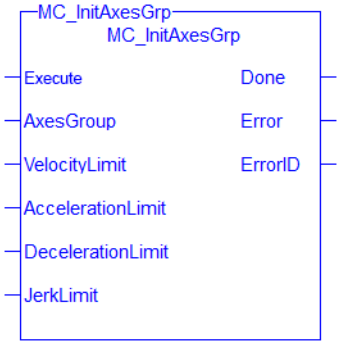
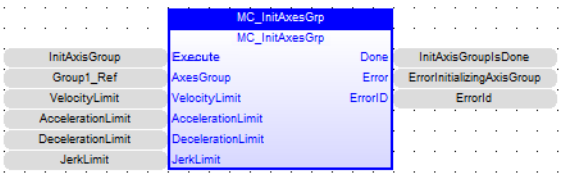
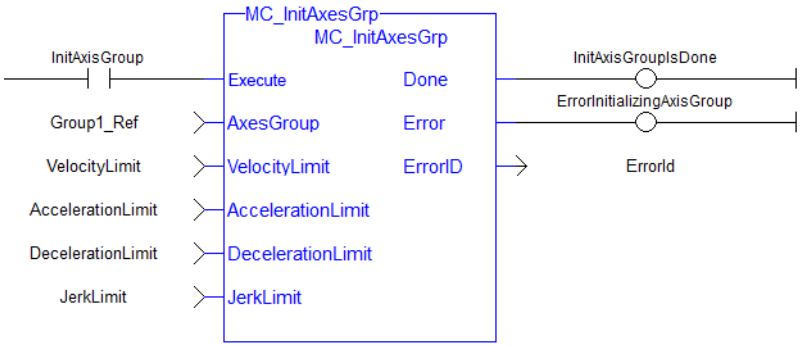


Figure 4-118: MC_InitAxesGrp

4.3.11.0.3.4 FBD Language Example



4.3.11.0.3.5 FFLD Language Example



4.3.11.0.3.6 IL Language Example

```
            BEGIN_IL
     CAL Inst_MC_InitAxesGrp( initAxesGrp, grp, velLim, accelLim, decelLim,
jerkLim )
END_IL
```

4.3.11.0.3.7 ST Language Example

```
            (* Inst_MC_InitAxesGrpST example *)
Inst_MC_InitAxesGrp( initAxesGrp, grp, velLim, accelLim, decelLim, jerkLim );
```

See Also

- "Example: Valid Setup" (➔ p. 563)
- "MC_ErrorDescription" (➔ p. 533)

4.3.11.0.4 MC_RemAxisFromGrp

PLCopen ✓ Pipe Network ✓

Function Block - Removes an individual axis from an axis group.

4.3.11.0.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to remove an axis from the group.
AxesGroupRef	AXES_GROUP_REF	No range	N/A	No default	The axis group from which the axis is removed.
IdentInGroup	UINT	0, MaxNumberOfAxes - 1	N/A	No default	The zero-based index of the axis in the group. • The axis index in the group must contain a valid axis. • The index must be less than the maximum number of axes the group can contain. • `MaxNumberOfAxes` is a property of the axes group and is set when the group is created.

4.3.11.0.4.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	INT			Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.11.0.4.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- This function or function block does not generate any motion.
- This function block can be issued in the group states: (GroupDisabled, GroupStandby, or GroupErrorStop).
  - The group's state changes to GroupDisabled if the axis removed is the last valid axis in the group.
- **MC_RemAxisFromGrp** fails if the group is in any state other than GroupStandby or GroupDisabled.
  - See Group State Diagrams for more information.

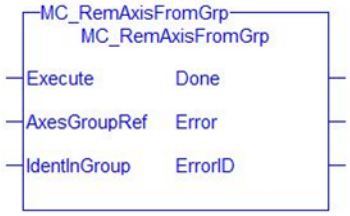
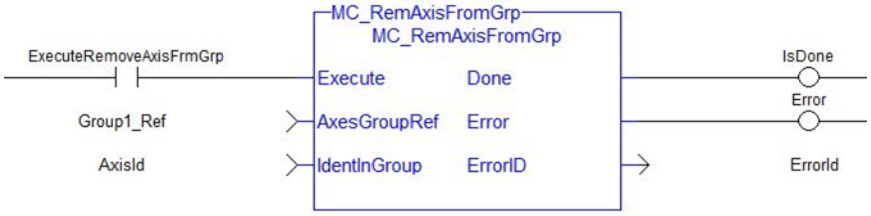


Figure 4-119: MC_RemAxisFromGrp

4.3.11.0.4.4 FBD Language Example

4.3.11.0.4.5 FFLD Language Example



4.3.11.0.4.6 IL Language Example

```
            BEGIN_IL

     CAL Inst_MC_RemAxisFromGrp( ExecuteRemAxisFromGrp, Group1_Ref, AxisId )
END_IL
```

4.3.11.0.4.7 ST Language Example

```
(* Inst_MC_InitAxisGrpST example *)
Inst_MC_RemAxisFromGrp( ExecuteRemAxisFromGrp, Group1_Ref, AxisId );
```

See Also

- "MC_AddAxisToGrp" (➔ p. 560)
- "MC_ErrorDescription" (➔ p. 533)
- "MC_UngroupAllAxes" (➔ p. 583)

4.3.11.0.5 MC_SetKinTra

PLCopen

Function Block - Sets the kinematic transform between the Machine Coordinate System and the Axes Coordinate System.

4.3.11.0.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to load the kinematic transform.

Input	Data Type	Range	Unit	Default	Description
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axes group that receives the axis trajectory values from the kinematic transform.
KinTransform	MC_KIN_REF	No range	N/A	No default	Kinematic robotic transform defined by the "MC_TRANSFORM_TYPE_GANTRY" (→ p. 586).

4.3.11.0.5.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	INT			Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.11.0.5.3 Remarks

- It is useful for robotics, allowing the application to command motion in Cartesian coordinates for the robotic system.
- After **MC_SetKinTra(...)** is called, the controller automatically calculates the inverse kinematics for the robot axes, converting the robot path motion into the individual robot joint axis trajectories.
- Several transform types are available for common robotic systems and are configurable with the "MC_TRANSFORM_TYPE_GANTRY" (→ p. 586).
- The parameters in the MC_KIN_REF structure define the specific robot geometry.
- See "MC_TRANSFORM_TYPE_GANTRY" (→ p. 586) for a description of the structure.

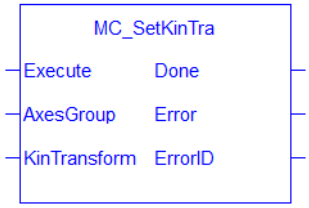
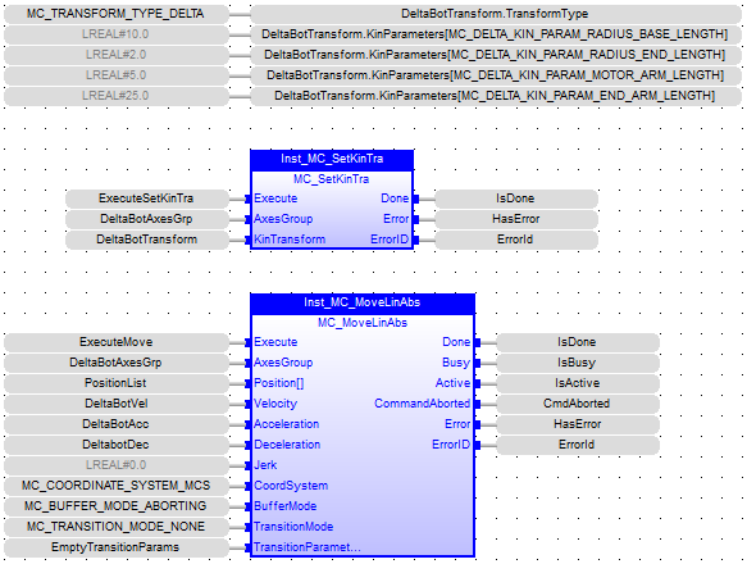
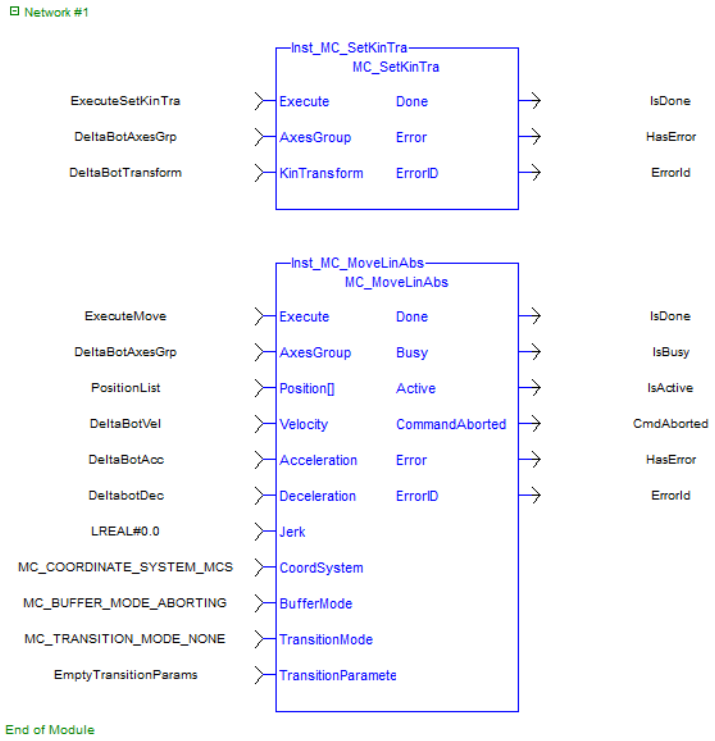


Figure 4-120: MC_SetKinTra

4.3.11.0.5.4 FBD Language Example



4.3.11.0.5.5 FFLD Language Example



4.3.11.0.5.6 IL Language Example

Not available.

4.3.11.0.5.7 ST Language Example

```
// MC_SetKinTra ST Example

// DeltaBotTransform is of type MC_KIN_REF

DeltaBotTransform.TransformType := MC_TRANSFORM_TYPE_DELTA;
DeltaBotTransform.KinParameters[MC_DELTA_KIN_PARAM_RADIUS_BASE_LENGTH] :=
10.0;
DeltaBotTransform.KinParameters[MC_DELTA_KIN_PARAM_RADIUS_END_LENGTH]  :=
2.0;
DeltaBotTransform.KinParameters[MC_DELTA_KIN_PARAM_MOTOR_ARM_LENGTH]   :=
5.0;
DeltaBotTransform.KinParameters[MC_DELTA_KIN_PARAM_END_ARM_LENGTH]     :=
25.0;

Inst_MC_SetKinTra(True, DeltaBotAxesGrp, DeltaBotTransform);

// ...

Inst_MC_MoveLinAbs(True, DeltaBotAxesGrp, PositionList, DeltaBotVel,
DeltaBotAcc, DeltaBotDec, LREAL#0.0, MC_COORDINATE_SYSTEM_MCS, MC_BUFFER_
MODE_ABORTING, MC_TRANSITION_MODE_NONE, EmptyTransitionParams);
```

4.3.11.0.6 MC_UngroupAllAxes

PLCopen ✓ Pipe Network ✓

Function Block - Removes all axes from an axes group.

4.3.11.0.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to remove all axes in the axes group.
AxesGroupRef	AXES_GROUP_REF	No range	N/A	No default	The axis group from which to remove all axes.

4.3.11.0.6.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	INT			Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.11.0.6.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- This function or function block does not generate any motion.
- This function block can be issued in the group states: (GroupDisabled, GroupStandby, or GroupErrorStop).
  - The axes group state is changed to GroupDisabled upon successful completion.
- **MC_UngroupAllAxes** fails if the group is in any state other than GroupStandby or GroupDisabled.
  - See Group State Diagrams for more information.

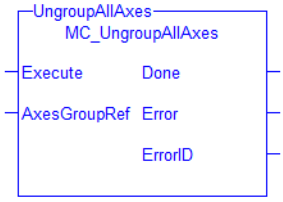
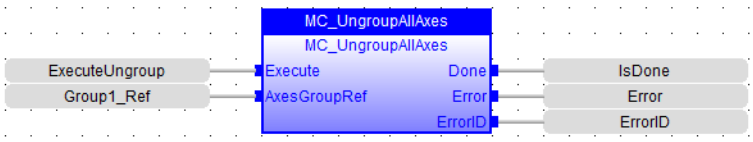
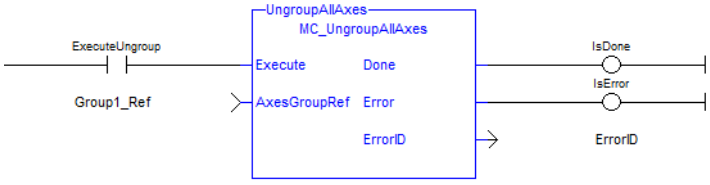


Figure 4-121: MC_UngroupAllAxes

4.3.11.0.6.4 FBD Language Example



4.3.11.0.6.5 FFLD Language Example



4.3.11.0.6.6 IL Language Example

```
            BEGIN_IL

                 CAL Inst_MC_UngroupAllAxes( ExecuteUngroup, Group1_Ref )
END_IL
```

4.3.11.0.6.7 ST Language Example

```
            Inst_MC_UngroupAllAxes( ExecuteUngroup, Group1_Ref );
```

See Also

- "MC_AddAxisToGrp" (→ p. 560)
- "MC_ErrorDescription" (→ p. 533)
- "MC_RemAxisFromGrp" (→ p. 579)

4.3.11.0.7 MC_KIN_REF Structure

The MC_KIN_REF structure defines the robotic system transform type and its parameters.

The parameters are specific to each transform type.

4.3.11.0.7.1 General MC_KIN_REF Structure

Member	Data Type	Description	Related Function Block
TransformType	UINT	A number that identifies the specific robotic system transform. The #defines for the transform types are: • "MC_TRANSFORM_TYPE_GANTRY" (→ p. 586) • "MC_TRANSFORM_TYPE_GANTRY_WITH_SKEW" (→ p. 586) • "MC_TRANSFORM_TYPE_HBOT" (→ p. 586) • "MC_TRANSFORM_TYPE_TBOT" (→ p. 587) • "MC_TRANSFORM_TYPE_DELTA (*Experimental*)" (→ p. 587) • "MC_TRANSFORM_TYPE_SCARA_ELBOW_POS (*Experimental*)" (→ p. 589) • "MC_TRANSFORM_TYPE_SCARA_ELBOW_NEG (*Experimental*)" (→ p. 590)	"MC_SetKinTra" (→ p. 581)
KinParameters [0 - 31]	LREAL	• Array to define the robotic system and its kinematic transform. • A maximum of 32 parameters are allowed. • The parameter count (0 to 32) and the definition of each parameter is determined by the specific **TransformType**.	"MC_SetKinTra" (→ p. 581)

4.3.11.0.7.2 Robotic Transform Parameters

- These parameters must be specified for all ACS axes in the AxisGroup.
- There are two parameters for each ACS axis.

4.3.12 MC_TRANSFORM_TYPE_GANTRY

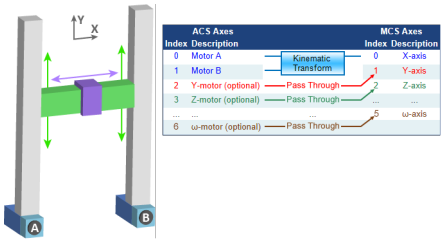


Figure 4-122: Axis Mapping - MC_TRANSFORM_TYPE_GANTRY

Parameter Count	Parameter #	Description	Range
1	MC_GANTRY_KIN_PARAM_ REVERSE_X	Reverse X-axis.	• 0 = Do Not Reverse. • 1 = Reverse.

4.3.13 MC_TRANSFORM_TYPE_GANTRY_WITH_SKEW

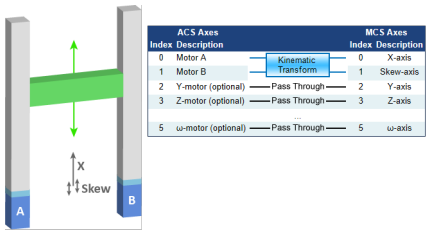


Figure 4-123: Axis Mapping - MC_TRANSFORM_TYPE_GANTRY_WITH_SKEW

Parameter Count	Parameter #	Description	Range
3	MC_GANTRY_WITH_SKEW_KIN_ PARAM_REVERSE_X	Reverse X-axis.	• 0 = Do Not Reverse. • 1 = Reverse.
	MC_GANTRY_WITH_SKEW_KIN_ PARAM_SKEW_SCALE	Skew to linear unit ratio.	Positive value.
	MC_GANTRY_WITH_SKEW_KIN_ PARAM_REVERSE_SKEW	Reverse Skew-axis.	• 0 = Do Not Reverse. • 1 = Reverse.

4.3.14 MC_TRANSFORM_TYPE_HBOT

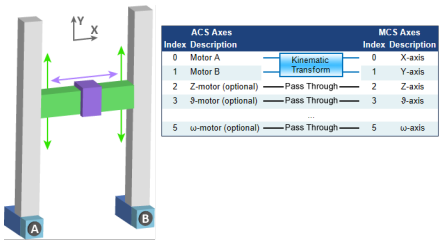


Figure 4-124: Axis Mapping - MC_TRANSFORM_TYPE_HBOT

Parameter Count	Parameter #	Description	Range
3	MC_HBOT_KIN_PARAM_ LIN_TO_ROT_RATIO	Linear to rotational unit ratio. Example: cm/degree.	Positive value.
	MC_HBOT_KIN_PARAM_ REVERSE_X	Reverse X-axis.	• 0 = Do Not Reverse. • 1 = Reverse.
	MC_HBOT_KIN_PARAM_ REVERSE_Y	Reverse Y-axis.	• 0 = Do Not Reverse. • 1 = Reverse.

4.3.15 MC_TRANSFORM_TYPE_TBOT

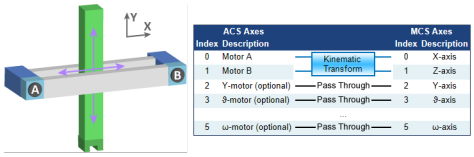


Figure 4-125: Axis Mapping - MC_TRANSFORM_TYPE_TBOT

Parameter Count	Parameter #	Description	Range
3	MC_TBOT_KIN_PARAM_ LIN_TO_ROT_RATIO	Linear to rotational unit ratio. Example: cm/degree.	Positive value.
	MC_TBOT_KIN_PARAM_ REVERSE_X	Reverse X-axis.	• 0 = Do Not Reverse. • 1 = Reverse.
	MC_TBOT_KIN_PARAM_ REVERSE_Z	Reverse Y-axis.	• 0 = Do Not Reverse. • 1 = Reverse.

4.3.16 MC_TRANSFORM_TYPE_DELTA (*Experimental*)

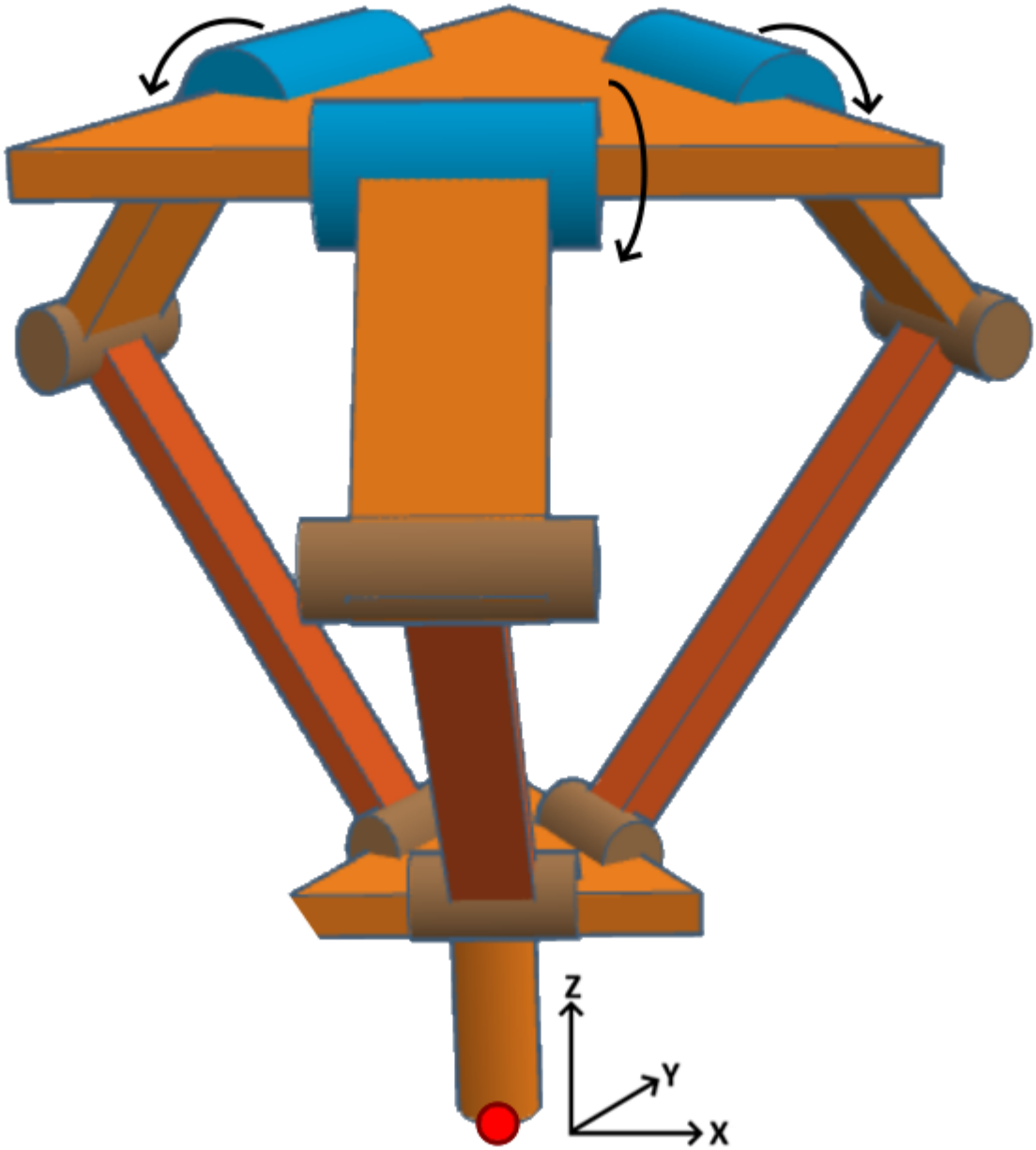


Figure 4-126: Axis Mapping - MC_TRANSFORM_TYPE_DELTA

Parameter Count	Parameter #	Description	Range
4	MC_DELTA_KIN_PARAM_RADIUS_BASE_LENGTH	RadiusBaseLength	Positive, non-zero value.
	MC_DELTA_KIN_PARAM_RADIUS_END_LENGTH	RadiusEndLength	Positive, non-zero value.
	MC_DELTA_KIN_PARAM_MOTOR_ARM_LENGTH	MotorArmLength	Positive, non-zero value.
	MC_DELTA_KIN_PARAM_END_ARM_LENGTH	EndArmLength	Positive, non-zero value.

4.3.17 MC_TRANSFORM_TYPE_SCARA_ELBOW_POS (*Experimental*)

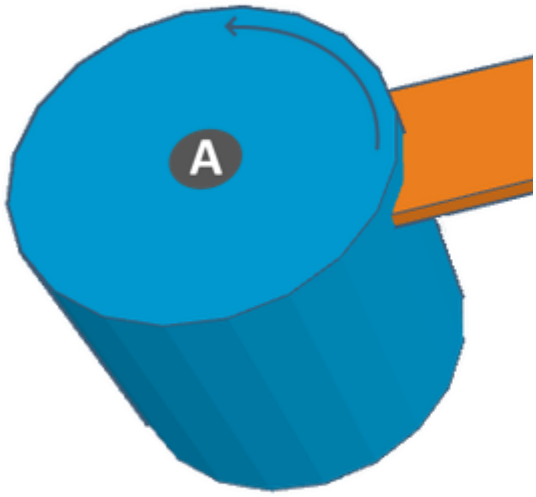


Figure 4-127: Axis Mapping - MC_TRANSFORM_TYPE_SCARA_ELBOW_POS no Wrist

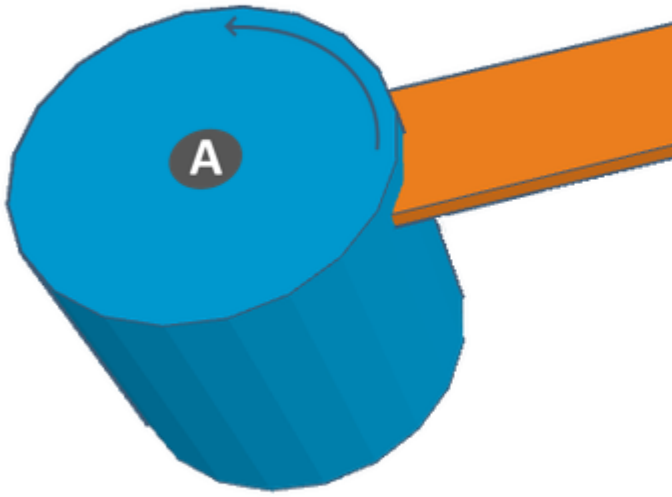


Figure 4-128: Axis Mapping - MC_TRANSFORM_TYPE_SCARA_ELBOW_POS with Wrist

Parameter Count	Parameter #	Description	Range
2 or 3 The parameter count depends on the axis number of the AxisGroup. • 2 AxesGroup: 2 Parameters. (No Wrist) • 3 AxesGroup: 3 Parameters. (with Wrist)	MC_SCARA_KIN_PARAM_UPPER_ARM_LENGTH	RobotUpperArmLength	Positive, non-zero value.
	MC_SCARA_KIN_PARAM_LOWER_ARM_LENGTH	RobotLowerArmLength	Positive, non-zero value.

Parameter Count	Parameter #	Description	Range
	MC_SCARA_KIN_PARAM_WRIST_LENGTH	RobotWristLength Available for SCARA with wrist only.	Positive, non-zero value.

4.3.18 MC_TRANSFORM_TYPE_SCARA_ELBOW_NEG (*Experimental*)

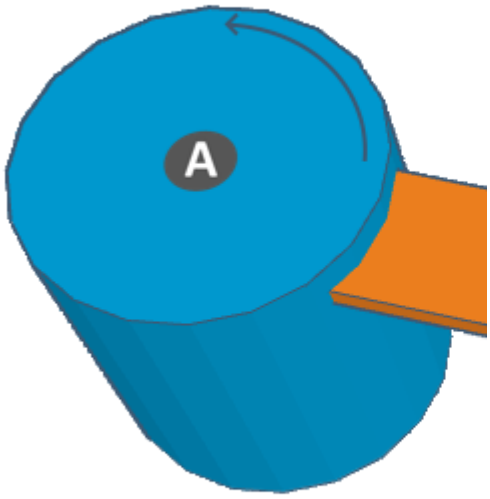


Figure 4-129: Axis Mapping - MC_TRANSFORM_TYPE_SCARA_ELBOW_NEG no Wrist

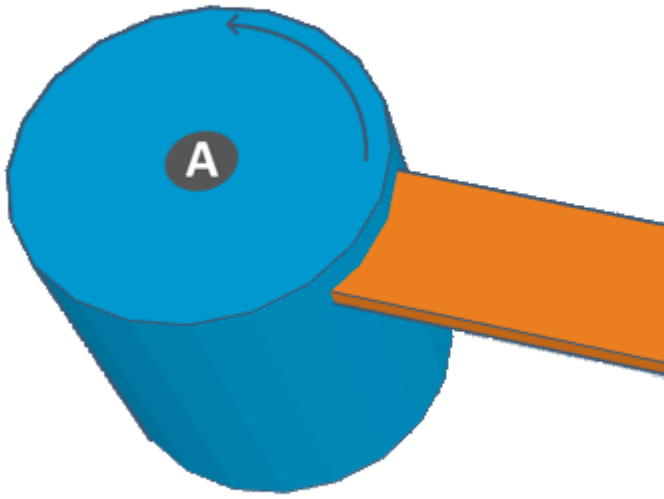


Figure 4-130: Axis Mapping - MC_TRANSFORM_TYPE_SCARA_ELBOW_NEG with Wrist

Parameter Count	Parameter #	Description	Range
2 or 3 The parameter count depends on the axis number of the AxisGroup. • 2 AxesGroup: 2 Parameters. (No Wrist) • 3 AxesGroup: 3 Parameters. (with Wrist)	MC_SCARA_KIN_PARAM_UPPER_ARM_LENGTH	RobotUpperArmLength	Positive, non-zero value.
	MC_SCARA_KIN_PARAM_LOWER_ARM_LENGTH	RobotLowerArmLength	Positive, non-zero value.

Parameter Count	Parameter #	Description	Range
	MC_SCARA_KIN_PARAM_WRIST_LENGTH	RobotWristLength Available for SCARA with wrist only.	Positive, non-zero value.

4.3.18.1 Motion / Common - Coordinated Motion - Info

Name	Description
MC_GrpReadActAcc	Reads the actual acceleration of the group and the axes in the group.
MC_GrpReadActPos	Reads the actual position of the axes in the group.
MC_GrpReadActVel	Reads the actual velocity of the group and the axes in the group.
MC_GrpReadCmdPos	Reads the command position of the axes in the group.
MC_GrpReadCmdVel	Reads the command velocity of the axes in the group and the path velocity.
MC_GrpReadError	Reads the Group ErrorID in state ERRORSTOP.
MC_GrpReadStatus	Returns the status of an axes group.

4.3.18.1.1 MC_GrpReadActAcc

PLCopen ✓ Pipe Network ✓

Function Block - Reads the actual acceleration of the group and the axes in the group.

4.3.18.1.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	If TRUE, this function block reads the current actual acceleration of the group and the axes in the group.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axes group the actual acceleration is read from.
CoordSystem	SINT	One of these enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2	N/A	No default	The coordinate system used when reading the actual acceleration.

Input	Data Type	Range	Unit	Default	Description
Acceleration[ ]	LREAL	No range	User unit/sec^2	No default	An array where the acceleration data is written. • The length of the array must equal the maximum number of axes allowed in the group. • The maximum number of axes is an argument to the "Example: Valid Setup" (→ p. 563) function block used to create axes groups.

4.3.18.1.1.2 Outputs

Output	Data Type	Range	Unit	Description
Valid	BOOL			If TRUE, the accelerations have been read without error.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	INT			Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.
PathAcceleration	LREAL		User unit/sec^2	The current measured path acceleration of the group. This is measured by taking the square root of the sum of the squared accelerations of each axis.

4.3.18.1.1.3 Remarks

NOTE

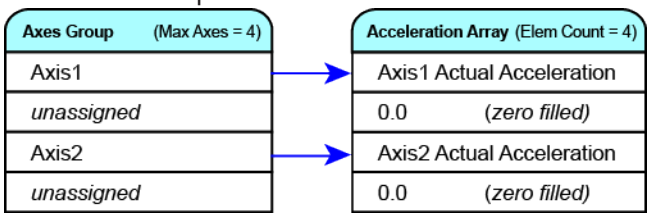
This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- The **MC_GrpReadActAcc** function block fills the array specified by the Acceleration argument with the actual acceleration of the system in the coordinate system specified by the `CoordSystem` argument.
  - The measured path acceleration is calculated and reported via the PathAcceleration output.
- This function or function block does not generate any motion.

NOTE

- The actual acceleration is smoothed over the last 10 samples.
  This reduces the error in acceleration estimation but introduces a small amount of phase delay in the reported accelerations.
- Currently, only the ACS coordinate system is supported.
  See Coordinate Systems for more information.

- There is a one-to-one correspondence between the axes in the Axes Group array and the acceleration values in the Acceleration array.
- Each element in the Acceleration array corresponds to the axis element in the Axes Group array.
- If an index in the Axes Group is unassigned, the acceleration value for that array element in the Acceleration array is 0 (zero).
- If the element does contain an axis, the acceleration value is filled with the current actual acceleration for that axis.
  - This is an example of how this works:



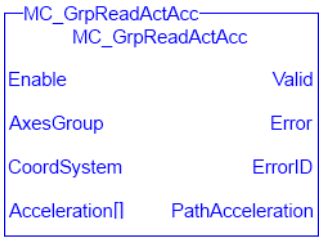
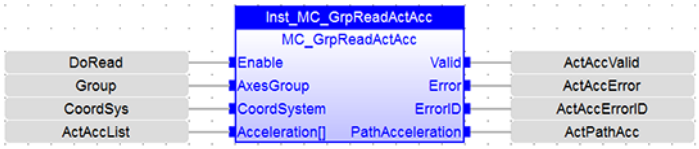
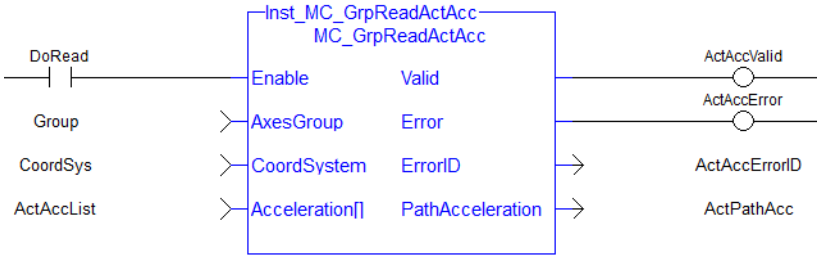


Figure 4-131: MC_GrpReadActAcc

4.3.18.1.1.4 FBD Language Example



4.3.18.1.1.5 FFLD Language Example



4.3.18.1.1.6 IL Language Example

```
            BEGIN_IL
     CAL Inst_MC_GrpReadActAcc( DoRead, Group, CoordSys, AccList )
END_IL
```

4.3.18.1.1.7 ST Language Example

```
Inst_MC_GrpReadActAcc( DoRead, Group, CoordSys, AccList );
```

See Also

- "MC_GrpReadActPos" (→ p. 594)
- "MC_GrpReadActVel" (→ p. 596)
- "MC_GrpReadCmdPos" (→ p. 599)
- "MC_GrpReadCmdVel" (→ p. 602)

4.3.18.1.2 MC_GrpReadActPos

PLCopen ✓ Pipe Network ✓

Function Block - Reads the actual position of the axes in the group.

4.3.18.1.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	If TRUE, this function block reads the current actual position of the axes in the group.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axes group the actual position is read from.
CoordSystem	SINT	One of these enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2	N/A	No default	The coordinate system used when reading the actual position.

Input	Data Type	Range	Unit	Default	Description
Position[]	LREAL	No range	User units	No default	An array where the position data is written. • The length of the array must equal the maximum number of axes allowed in the group. • The maximum number of axes is an argument to the "Example: Valid Setup" (→ p. 563) function block used to create axes groups.

4.3.18.1.2.2 Outputs

Output	Data Type	Range	Unit	Description
Valid	BOOL	No range	N/A	If TRUE, the positions have been read without error.
Error	BOOL	No range	N/A	If TRUE, an error has occurred.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.18.1.2.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

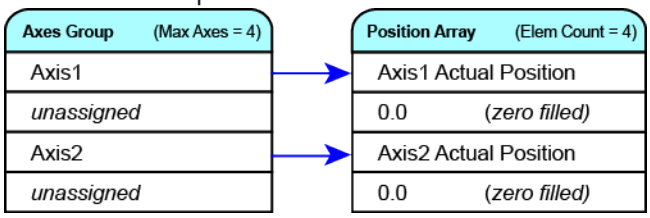
- See Coordinated Motion, the top-level topic for Coordinated Motion.
- The **MC_GrpReadActPos** function block fills the array specified by the Position argument with the actual position of the system in the coordinate system specified by the `CoordSystem` argument.
- This function or function block does not generate any motion.

NOTE

Currently, only the ACS coordinate system is supported.
See Coordinate Systems for more information.

- There is a one-to-one correspondence between the axes in the Axes Group and the position values in the Position Array.
- Each element in the Position Array corresponds to the axis element in the Axis Group array.
- If an index in the Axes Group is unassigned, the position value for that array element in the Position Array is 0 (zero).
- If the element does contain an axis, the position value is filled with the current actual position for that axis.

- This is an example of how this works:



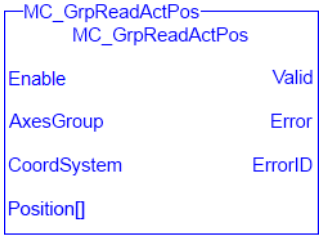
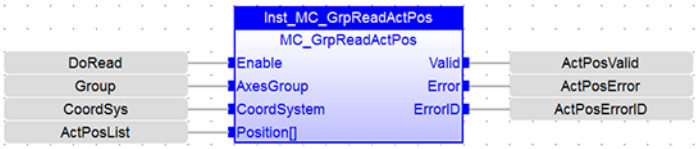
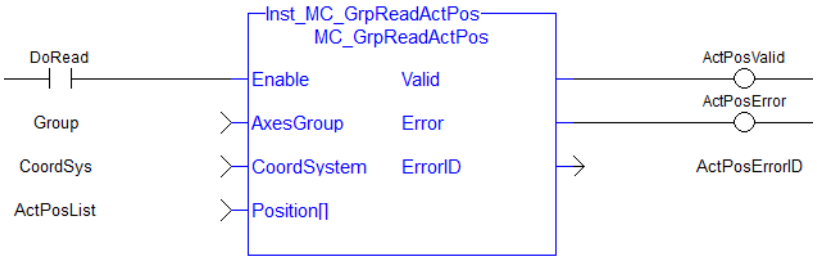


Figure 4-132: MC_GrpReadActPos

4.3.18.1.2.4 FBD Language Example



4.3.18.1.2.5 FFLD Language Example



4.3.18.1.2.6 IL Language Example

```
            BEGIN_IL
     CAL Inst_MC_GrpReadActPos( DoRead, Group, CoordSys, PosList )
END_IL
```

4.3.18.1.2.7 ST Language Example

```
Inst_MC_GrpReadActPos( DoRead, Group, CoordSys, PosList );
```

See Also

- "MC_GrpReadActAcc" (→ p. 591)
- "MC_GrpReadActVel" (→ p. 596)
- "MC_GrpReadCmdPos" (→ p. 599)
- "MC_GrpReadCmdVel" (→ p. 602)

4.3.18.1.3 MC_GrpReadActVel

PLCopen ✓ Pipe Network ✓

Function Block - Reads the actual velocity of the group and the axes in the group.

4.3.18.1.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	If TRUE, this function block reads the current actual velocity of the group and the axes in the group.
AxesGroup	AXES_GROUP_REF	N/A	N/A	No default	The axes group the actual velocity is read from.
CoordSystem	SINT	One of these enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2	N/A	No default	The coordinate system used when reading the actual velocity.
Velocity[ ]	LREAL	No range	User unit/sec	No default	An array where the velocity data is written. • The length of the array must equal the maximum number of axes allowed in the group. • The maximum number of axes is an argument to the "Example: Valid Setup" (→ p. 563) function block used to create axes groups.

4.3.18.1.3.2 Outputs

Output	Data Type	Range	Unit	Description
Valid	BOOL	No range	N/A	If TRUE, the velocities were read without error.
Error	BOOL	No range	N/A	If TRUE, an error has occurred.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

Output	Data Type	Range	Unit	Description
PathVelocity	LREAL	No range	User unit/sec	The current measured path velocity of the group. This is measured by taking the square root of the sum of the squared velocities of each axis.

4.3.18.1.3.3 Remarks

NOTE

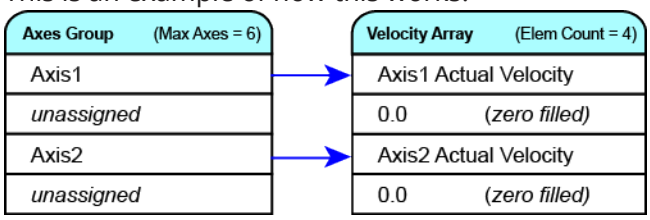
This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- The **MC_GrpReadActVel** function block fills the array specified by the Velocity argument with the actual velocity of the system in the coordinate system specified by the `CoordSystem` argument.
  - The measured path velocity is also calculated and reported by the PathVelocity output.
- This function or function block does not generate any motion.

NOTE

- The actual velocity is smoothed over the last 10 samples.
  This reduces the error in velocity estimation but introduces a small amount of phase delay in the reported velocities.
- Currently, only the ACS coordinate system is supported.
  See Coordinate Systems for more information.

- There is a one-to-one correspondence between the axes in the Axes Group and the velocity values in the Velocity Array.
- Each element in the Velocity Array corresponds to the axis element in the Axis Group array.
- If a index in the Axes Group is unassigned, the velocity value for that array element in the Velocity array is 0 (zero).
- If the element does contain an axis, the velocity value is filled with the current actual velocity for that axis.
  - This is an example of how this works:



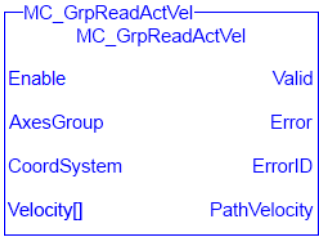
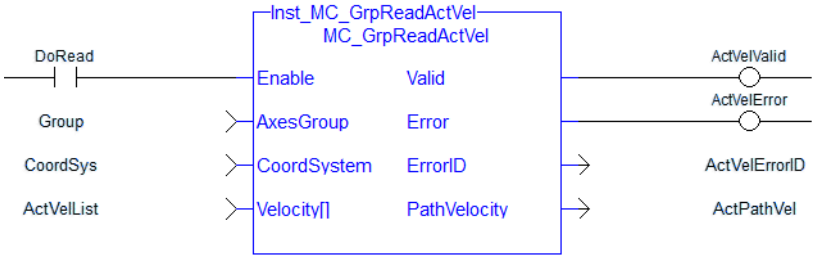


Figure 4-133: MC_GrpReadActVel

4.3.18.1.3.4 FBD Language Example

4.3.18.1.3.5 FFLD Language Example



4.3.18.1.3.6 IL Language Example

```
            BEGIN_IL
     CAL Inst_MC_GrpReadActVel(DoRead, Group, CoordsSys, VelList)
END_IL
```

4.3.18.1.3.7 ST Language Example

```
Inst_MC_GrpReadActVel(DoRead, Group, CoordSys, VelList);
```

See Also

- "MC_GrpReadActAcc" (→ p. 591)
- "MC_GrpReadActPos" (→ p. 594)
- "MC_GrpReadCmdPos" (→ p. 599)
- "MC_GrpReadCmdVel" (→ p. 602)

4.3.18.1.4 MC_GrpReadCmdPos

PLCopen ✓ Pipe Network ✓

Function Block - Reads the command position of the axes in the group.

4.3.18.1.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	If TRUE, this function block reads the current commanded position of the axes in the group.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axes group the commanded position is read from.

Input	Data Type	Range	Unit	Default	Description
CoordSystem	SINT	One of these enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2	N/A	No default	The coordinate system used when reading the commanded position.
Position[]	LREAL	No range	User units	No default	An array where the position data is written. • The length of the array must equal the maximum number of axes allowed in the group. • The maximum number of axes is an argument to the "Example: Valid Setup" (→ p. 563) function block used to create axes groups.

4.3.18.1.4.2 Outputs

Output	Data Type	Range	Unit	Description
Valid	BOOL	No range	N/A	If TRUE, the positions have been read without error.
Error	BOOL	No range	N/A	If TRUE, an error has occurred.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.18.1.4.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- The **MC_GrpReadCmdPos** function block fills the array (specified by the `Position` argument) with the commanded position of the coordinate system specified by the `CoordSystem` argument.
- This function or function block does not generate any motion.

NOTE

Currently, only the ACS coordinate system is supported.
See Coordinate Systems for more information.

- There is a one-to-one correspondence between the axes in the Axes Group and the position values in the Position Array.
- Each element in the Position Array corresponds to the axis element in the Axis Group array.
- If an index in the Axes Group is unassigned, the position value for that array element in the Position Array is 0 (zero).
- If the element does contain an axis, the position value is filled with the current actual position for that axis.
  - This is an example of how this works:

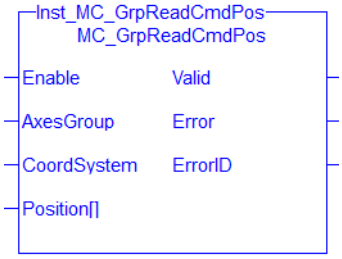
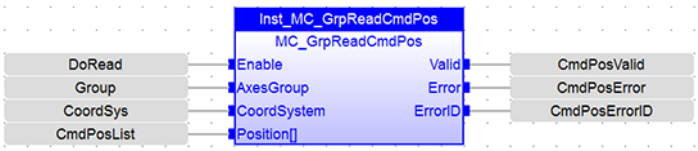
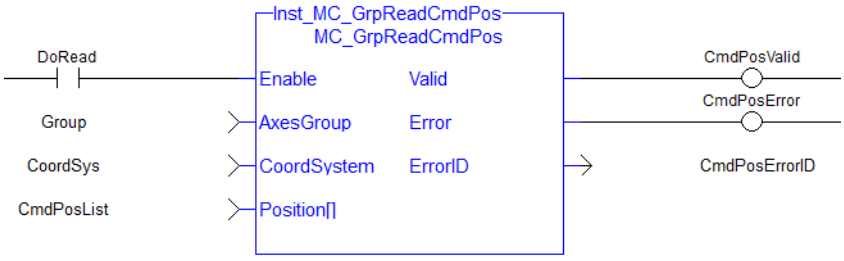


Figure 4-134: MC_GrpReadCmdPos

4.3.18.1.4.4 FBD Language Example



4.3.18.1.4.5 FFLD Language Example



4.3.18.1.4.6 IL Language Example

```
            BEGIN_IL

     CAL Inst_MC_GrpReadCmdPos( DoRead, Group, CoordSys, PosList )
END_IL
```

4.3.18.1.4.7 ST Language Example

```
            (*MC_GrpReadCmdPos ST example *)
Inst_MC_GrpReadCmdPos(DoRead, Group, CoordSys, PosList );
```

See Also

- "MC_GrpReadActAcc" (→ p. 591)
- "MC_GrpReadActPos" (→ p. 594)
- "MC_GrpReadActVel" (→ p. 596)
- "MC_GrpReadCmdVel" (→ p. 602)

4.3.18.1.5 MC_GrpReadCmdVel

PLCopen ✓ Pipe Network ✓

Function Block - Reads the command velocity of the axes in the group and the path velocity.

4.3.18.1.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	If TRUE, this function block reads the current commanded velocity of the group and the axes in the group.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axes group the commanded velocity is read from.
CoordSystem	SINT	One of these enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2	N/A	No default	The coordinate system used when reading the commanded velocity.

Input	Data Type	Range	Unit	Default	Description
Velocity[ ]	LREAL	No range	User unit/sec	No default	An array where the velocity data is written. • The length of the array must equal the maximum number of axes allowed in the group. • The maximum number of axes is an argument to the "Example: Valid Setup" (→ p. 563) function block used to create axes groups.

4.3.18.1.5.2 Outputs

Output	Data Type	Range	Unit	Description
Valid	BOOL	No range	N/A	If TRUE, the velocities were read without error.
Error	BOOL	No range	N/A	If TRUE, an error has occurred.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.
PathVelocity	LREAL	No range	User unit/sec	The current commanded path velocity of the group. This is measured by taking the square root of the sum of the squared velocities of each axis.

4.3.18.1.5.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

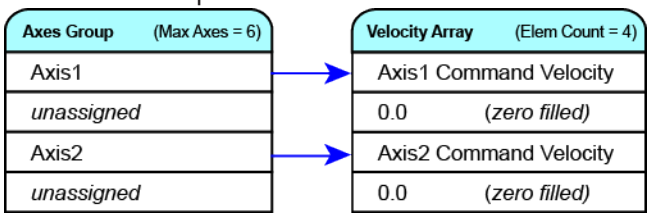
- See Coordinated Motion, the top-level topic for Coordinated Motion.
- The **MC_GrpReadCmdVel** function block fills the array specified by the `Velocity` argument with the commanded velocity for the coordinate system, which is specified by the `CoordSystem` argument.
  - The path velocity is reported by the PathVelocity output.
- This function or function block does not generate any motion.

NOTE

Currently, only the ACS coordinate system is supported.
See Coordinate Systems for more information.

- There is a one-to-one correspondence between the axes in the Axes Group and the velocity values in the Velocity Array.
- Each element in the Velocity Array corresponds to the axis element in the Axis Group array.
- If a index in the Axes Group is unassigned, the velocity value for that array element in the Velocity array is 0 (zero).

- If a index in the Axes Group is unassigned, the velocity value for that array element in the Velocity array is 0 (zero).
- If the element does contain an axis, the velocity value is filled with the current velocity for that axis.
  - This is an example of how this works:



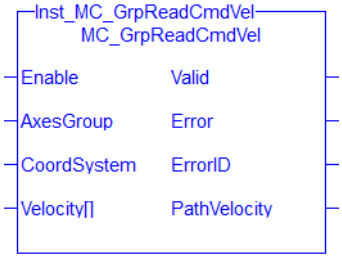
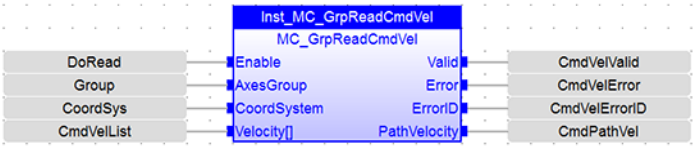
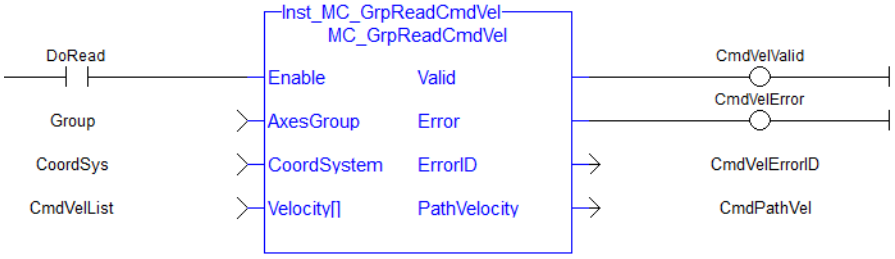


Figure 4-135: MC_GrpReadCmdVel

4.3.18.1.5.4 FBD Language Example



4.3.18.1.5.5 FFLD Language Example



4.3.18.1.5.6 IL Language Example

```
            BEGIN_IL

     CAL Inst_MC_GrpReadCmdVel( DoRead, Group, CoordSys, VelList )
END_IL
```

4.3.18.1.5.7 ST Language Example

```
            (*MC_GrpReadCmdVel ST example *)
Inst_MC_GrpReadCmdVel(DoRead, Group, CoordSys, VelList );
```

See Also

- "MC_GrpReadActAcc" (→ p. 591)
- "MC_GrpReadActPos" (→ p. 594)

- "MC_GrpReadActVel" (→ p. 596)
- "MC_GrpReadCmdPos" (→ p. 599)

4.3.18.1.6 MC_GrpReadError

PLCopen ✓ Pipe Network ✓

Function - Reads the Group ErrorID in state ERRORSTOP.

4.3.18.1.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Enables execution.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axes group the GroupErrorID is read from.

4.3.18.1.6.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL	No range	N/A	Indicates the function executed successfully.
GroupErrorID	INT	No range	N/A	Shows the Error ID for the given Axis Group. See the table in PLCopen Function Block ErrorIDs.

4.3.18.1.6.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- This function or function block does not generate any motion.

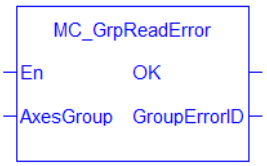
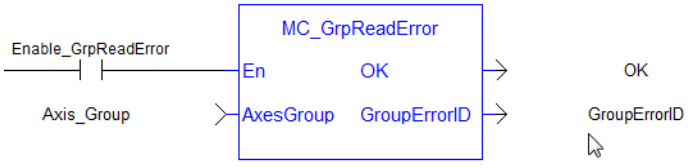


Figure 4-136: MC_GrpReadError

4.3.18.1.6.4 FBD Language Example

4.3.18.1.6.5 FFLD Language Example



4.3.18.1.6.6 IL Language Example

Not available.

4.3.18.1.6.7 ST Language Example

```
//Read a group error number
GroupErrorID:= MC_GrpReadError( Axis_Group );
```

See Also

- "MC_ErrorDescription" (→ p. 533)
- "MC_GrpReset" (→ p. 570)

4.3.18.1.7 MC_GrpReadStatus

PLCopen ✓ Pipe Network ✓

Function Block - Returns the status of an axes group.

4.3.18.1.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	If TRUE, the axes group status is read.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axes group the status is read from.

4.3.18.1.7.2 Outputs

Output	Data Type	Range	Unit	Description
Valid	BOOL			TRUE of valid outputs are available.
GroupMoving[1]	BOOL			The axes group is in the Moving state. This indicates the group is enabled and currently executing a coordinated motion command.
GroupHoming[1]	BOOL			Not supported.
GroupErrorStop[1]	BOOL			The axes group is in the ErrorStop state due to an axis error or group error. • The group cannot accept coordinated motion commands. • The execution of MC_GrpReset is required to change the group's state from ErrorStop to Standby.

GroupStandby[1]	BOOL			The axes group is in the Standby state.
				• The group:
				  • Is enabled and all its axes are enabled.
				  • Is not currently executing a coordinated motion command.
				• The axes group is ready to accept coordinated motion commands.

Output	Data Type	Range	Unit	Description

GroupStopping[1]	BOOL			The axes group is in the Stopping state due the execution of MC_GrpStop.
				• The axes group is enabled but cannot accept coordinated motion commands while in the Stopping state.
				• The axes group:
				  • Remains in the Stopping state while MC_GrpStop is executing.
				  • Will remain in the Stopping state while MC_GrpStop's Execute input is held high.

GroupDisabled[1]	BOOL			The axis group is in the Disabled state and cannot accept coordinated motion commands.
ConstantVelocity	BOOL			TRUE if the commanded path velocity is the same between the current scan of the application program and the previous scan. • ConstantVelocity is always TRUE for Direct moves. • The commanded path velocity of Direct moves is always 0 (zero).
Accelerating	BOOL	No range	N/A	TRUE if the commanded path velocity is accelerating between the current scan of the application program and the previous scan.
Decelerating	BOOL	No range	N/A	TRUE if the commanded path velocity is decelerating between the current scan of the application program and the previous scan.
InPosition	BOOL	No range	N/A	TRUE indicates that the axes group is "in position". These must be TRUE for the axes group to be in position: • The axes group is enabled. • There are no moves in the group's queue. • The servo loop is closed for each axis in the group. • There are no moves in the individual axis queue for each axis in the group. • The command delta is zero for each axis in the group. • The actual position is within the In-Position Bandwidth of the command position for each axis in the group.
Error	BOOL	No range	N/A	If TRUE, an error has occurred.
ErrorID	BOOL	No range	N/A	Indicates the error identifier if the Error output is set to TRUE. See the table in PLCopen Function Block ErrorIDs.

1 These outputs are mutually exclusive. Only one is TRUE at a time.
All others are FALSE.
See Group State Diagrams for more information.

4.3.18.1.7.3 Remarks

NOTE

This function or function block returns cached data.
See Programming a Dual Core Controller for more information.

See Coordinated Motion, the top-level topic for Coordinated Motion.

MC_GrpReadStatus returns the status of an axes group. This function block does not cause any motion. Refer to Group State Diagrams for details.

NOTE

This output is not currently supported. It will be supported in a future release.

- GroupHoming

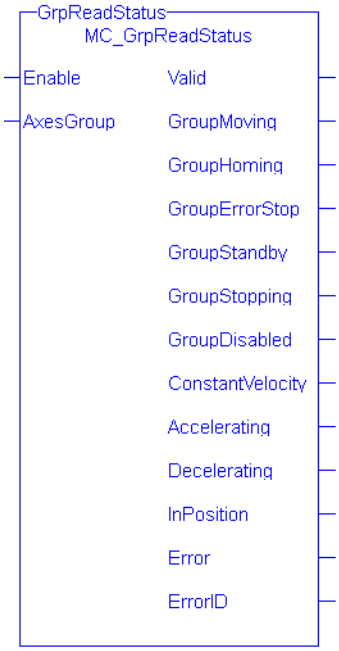
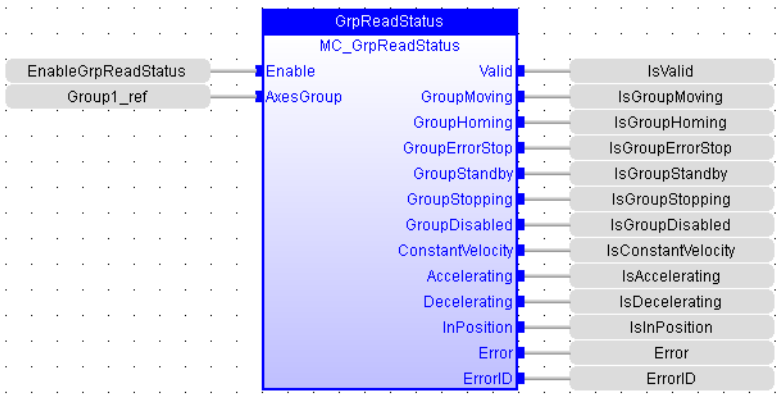
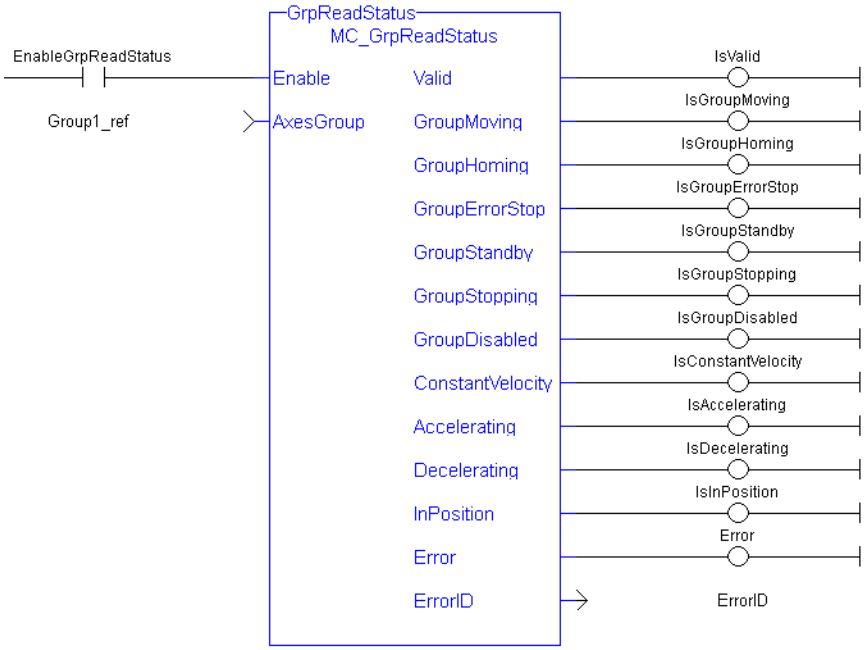


Figure 4-137: MC_GrpReadStatus

4.3.18.1.7.4 FBD Language Example



4.3.18.1.7.5 FFLD Language Example



4.3.18.1.7.6 IL Language Example

```
            BEGIN_IL

     CAL Inst_MC_GrpReadStatus( EnableGrpReadStatus, Group1_Ref )
END_IL
```

4.3.18.1.7.7 ST Language Example

```
//Check boolean status bits for an Axis Group
Inst_MC_GrpReadStatus( EnableGrpReadStatus, Group1_Ref );

IsGroupMoving:= Inst_MC_GrpReadStatus.GroupMoving;
IsGroupErrorStop:= Inst_MC_GrpReadStatus.GroupErrorStop;
IsGroupStandby:= Inst_MC_GrpReadStatus.GroupStandby;
IsGroupDisabled:= Inst_MC_GrpReadStatus.GroupDisabled;
Accelerating:= Inst_MC_GrpReadStatus.Accelerating;
IsConstantVelocity:= Inst_MC_GrpReadStatus.ConstantVelocity;
IsInPosition:= Inst_MC_GrpReadStatus.InPosition;
```

See Also

"MC_ErrorDescription" (→ p. 533)

4.3.18.2 Motion / Common - Coordinated Motion - Motion

Name	Description
MC_ AxisSetDefaults	Sets the default kinematic parameters for an axis.
MC_ ErrorDescription	Converts the PLCopen error IDs into message strings.
MC_GrpHalt	Performs a controlled motion stop of all axes in the group.

Name	Description
MC_ GrpSetOverride	Sets the velocity factor that is multiplied to the commanded velocity of all axes in the group.
MC_ MoveCircAbs	Commands interpolated circular movement on an axes group to the specified absolute positions in the coordinate system as specified by the CoordSystem argument.
MC_MoveCircRel	Commands interpolated circular movement on an axes group to the specified relative positions in the coordinate system as specified by the CoordSystem argument.
MC_MoveDirAbs	Commands movement of an axes group to an absolute position regardless of path.
MC_MoveDirRel	Commands movement of an axes group to a relative position regardless of path.
MC_MoveLinAbs	Commands interpolated linear movement on an axes group to the specified absolute positions.
MC_MoveLinRel	Commands interpolated linear movement on an axes group to the specified relative positions.

4.3.18.2.1 MC_AxisSetDefaults

PLCopen ✓ Pipe Network ✓

Function Block - Sets the default kinematic parameters for an axis.

4.3.18.2.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to set the default kinematic parameters.
Axis	AXIS_ REF	No range	N/A	No default	Reference to the axis which has its default kinematic parameters set.
Velocity	LREAL	0 < Velocity < (20 * Acceleration) and 0 < Velocity < (20 * Deceleration)	User unit/sec	No default	The default velocity. See Limitations on Acceleration and Jerk for more information.

Input	Data Type	Range	Unit	Default	Description
Acceleration	LREAL	(Velocity / 20) < Acceleration < (2 * Jerk)	User unit/sec^2	No default	Trapezoidal: Acceleration rate. S-curve: Maximum acceleration. • See S-curve and Trapezoidal Acceleration / Deceleration for more information. • See Acceleration and Jerk Parameters for Function Blocks for more information. • See Limitations on Acceleration and Jerk for more information.
Deceleration	LREAL	(Velocity / 20) < Deceleration < (2 * Jerk)	User unit/sec^2	No default	Trapezoidal: Deceleration rate. S-curve: Unused.

Input	Data Type	Range	Unit	Default	Description
Jerk	LREAL	(Velocity / 20) < Acceleration < (2 * Jerk) and (Velocity / 20) < Deceleration < (2 * Jerk)	User unit/sec^3	No default	Trapezoidal: 0 (zero). S-curve: Constant jerk. **NOTE** Currently the Jerk value is ignored for motion. Only trapezoidal motion is supported. • See S-curve and Trapezoidal Acceleration / Deceleration for more information. • See Acceleration and Jerk Parameters for Function Blocks for more information.

4.3.18.2.1.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.18.2.1.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- The **MC_AxisSetDefaults** function block sets the default kinematic variables for "MC_MoveDirAbs" (→ p. 631) and "MC_MoveDirRel" (→ p. 634).
  - These variables are only used with the MC_MoveDir function blocks.
- Each axis in the group must have the default kinematic parameters of Velocity, Acceleration, Deceleration, and Jerk set to values greater than 0 (zero).
  - A non-zero Jerk value performs an S-Curve instead of a trapezoidal move.
  - Each axis in the group must have these values set before a direct move can be started.
- The function block returns an error if the group state is not GroupStandby or GroupDisabled.

NOTE

Jerk with a non-zero value is currently not supported for coordinated motion.
Jerk parameters are currently ignored.

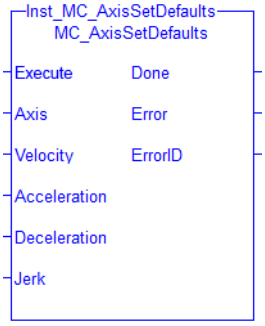
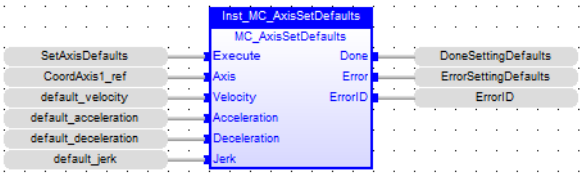
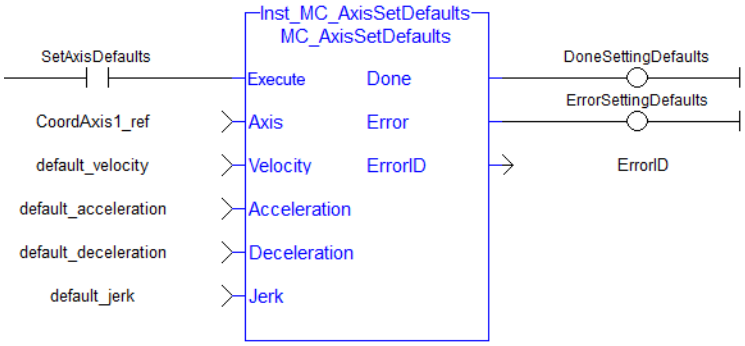


Figure 4-138: MC_AxisSetDefaults

4.3.18.2.1.4 FBD Language Example



4.3.18.2.1.5 FFLD Language Example



4.3.18.2.1.6 IL Language Example

```
            BEGIN_IL
     CAL Inst_MC_AxisSetDefaults( TRUE, CoordAxis1_Ref, default_velocity,
default_acceleration, default_deceleration, default_jerk)
END_IL
```

4.3.18.2.1.7 ST Language Example

```
 (* ST MC_AxisSetDefaults Example *)

default_velocity       := 50.0;
default_acceleration   := 250.0;
default_deceleration   := 300.0;
default_jerk           := 1000.0;

Inst_MC_AxisSetDefaults ( TRUE, CoordAxis1_ref, default_velocity, default_
acceleration, default_deceleration, default_jerk);
```

See Also

- Differences Between Functions and Function Blocks
- Function Call
- "MC_ErrorDescription" (→ p. 533)
- "MC_MoveDirAbs" (→ p. 631)
- "MC_MoveDirRel" (→ p. 634)

4.3.18.2.2 MC_GrpHalt

PLCopen ✓ Pipe Network ✓

Function Block - Performs a controlled motion stop of all axes in the group.

4.3.18.2.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, the command to halt all of the axes in the group is initiated.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axes group the axes are stopped in.
Deceleration	LREAL	0 < Deceleration	N/A	No default	The path deceleration rate for all axes in the group. See Limitations on Acceleration and Jerk for more information.
Jerk	LREAL	0 ≤ Jerk	N/A	No default	Not supported. See Limitations on Acceleration and Jerk for more information.

4.3.18.2.2.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	If TRUE, the command completed successfully.
Busy	BOOL	No range	N/A	TRUE from the moment the EXECUTE input is TRUE until the halt is complete.
Active	BOOL	No range	N/A	If TRUE, the halt is still executing.
CommandAborted	BOOL	No range	N/A	If TRUE, the command was aborted by another function block.
Error	BOOL	No range	N/A	If TRUE, an error has occurred.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

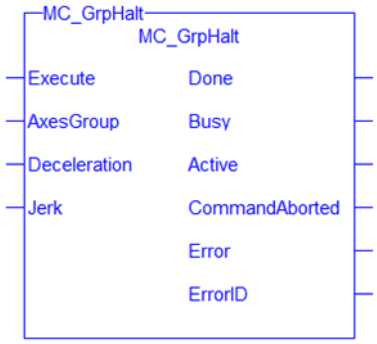
4.3.18.2.2.3 Remarks

NOTE

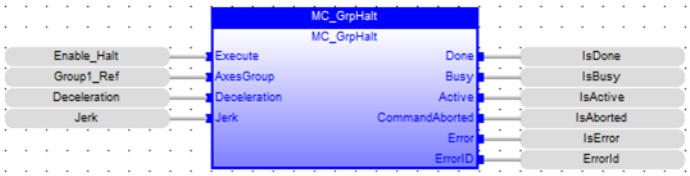
This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- When the path velocity reaches 0 (zero):
  - Any queued moves are flushed from the buffer.
  - The Done output is set.
  - The state transitions to GroupStandby.

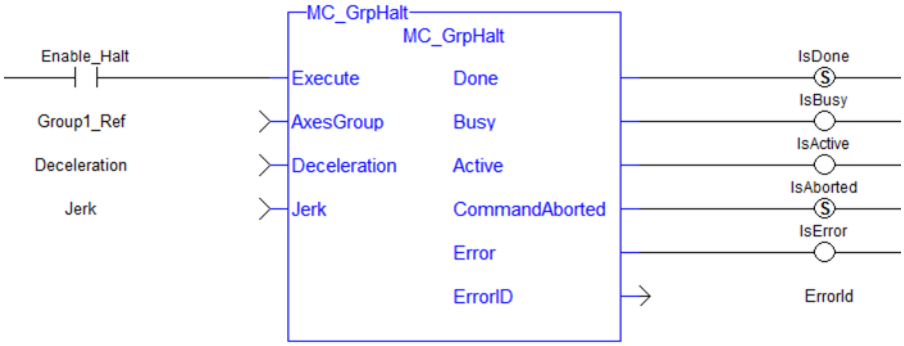
- **MC_GrpHalt** can be aborted.
- **MC_GrpHalt** does NOT prevent:
  - A single axis from executing.
  - Other Coordinated Motion moves from executing once MC_GrpHalt has completed.



4.3.18.2.2.4 FBD Language Example



4.3.18.2.2.5 FFLD Language Example



4.3.18.2.2.6 IL Language Example

```
            BEGIN_IL
     CAL Inst_MC_GrpHalt ( EnableHalt, Group1_Ref, Deceleration, Jerk )
END_IL
```

4.3.18.2.2.7 ST Language Example

```
Inst_MC_GrpHalt ( EnableHalt, Group1_Ref, Deceleration, Jerk );
```

See Also

- "MC_ErrorDescription" (→ p. 533)
- "Related Functions" (→ p. 574)

4.3.18.2.3 MC_GrpSetOverride

PLCopen ✓ Pipe Network ✓

Function Block - Sets the velocity factor that is multiplied to the commanded velocity of all axes in the group.

4.3.18.2.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Enable	BOOL	0, 1	N/A	No default	On the rising edge, changes the velocity multiplier for the axes group.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axes group the velocity multiplier is applied to.
VelFactor	REAL	0.0 to 2.0	N/A	No default	The new multiplier factor for the commanded velocity of the axes group.

4.3.18.2.3.2 Outputs

Output	Data Type	Range	Unit	Description
Enabled	BOOL			Indicates that the override was successful.
Busy	BOOL			If TRUE, the function block is executing.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	INT			Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.18.2.3.3 Remarks

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- MC_GrpSetOverride sets the velocity factor that is multiplied to the commanded velocity of all axes in the group.
- This function or function block does not generate any motion.

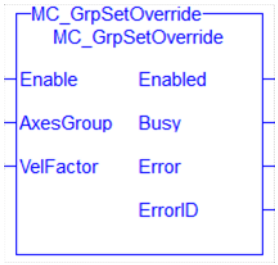
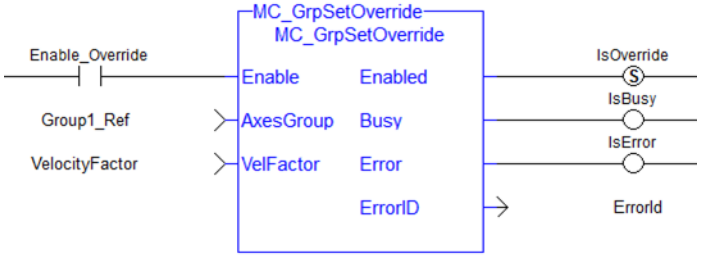


Figure 4-139: MC_GrpSetOverride

4.3.18.2.3.4 FBD Language Example

4.3.18.2.3.5 FFLD Language Example



4.3.18.2.3.6 IL Language Example

```
            BEGIN_IL

     CAL Inst_MC_GrpSetOverride( EnableOverride, Group1_Ref, VelocityFactor )
END_IL
```

4.3.18.2.3.7 ST Language Example

```
Inst_MC_GrpSetOverride( EnableOverride, Group1_Ref, VelocityFactor );
```

See Also

"MC_ErrorDescription" (→ p. 533)

4.3.18.2.4 MC_MoveCircAbs

PLCopen ✓ Pipe Network ✓

Function Block - Commands interpolated circular movement on an axes group to the specified absolute positions in the coordinate system as specified by the CoordSystem argument.

4.3.18.2.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to perform a circular absolute move.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axis group that performs the circular absolute move.
CircMode	SINT One of these enumeration values: • MC_CIRC_MODE_BORDER = 0 • MC_CIRC_MODE_CENTER = 1	No range	N/A	No default	Specifies the meaning of the AuxPoint[] input.

Input	Data Type	Range	Unit	Default	Description

AuxPoint[]	LREAL	0, Number of axes in group - 1	N/A	No default	**Required Precision**
					• 1 part in 100,000.
					• See Precision Requirements for Circular Move Input Parameters for more information.
					Description
					Array of absolute positions for each axis in the group.
					The meaning depends on the value of the CircMode input:
					• MC_CIRC_MODE_BORDER
					  • AuxPoint defines a point on the circle which is crossed on the path from the starting to the end point.
					• MC_CIRC_MODE_CENTER
					  • AuxPoint defines the center point of the circle.

EndPoint[]	LREAL	0, Number of axes in group - 1	N/A	No default	**Required Precision** • 1 part in 100,000. • See Precision Requirements for Circular Move Input Parameters for more information. **Description** Array of absolute end positions for each axis in the group.

Input	Data Type	Range	Unit	Default	Description
PathChoice	SINT One of these enumeration values: • MC_CIRC_PATHCHOICE_CLOCKWISE = 0 = Clockwise • MC_CIRC_PATHCHOICE_COUNTERCLOCKWISE = 1 = Counterclockwise	No range	N/A	No default	Specifies the direction of the path. This argument is only relevant when **CircMode** is MC_CIRC_MODE_CENTER.
Velocity	LREAL	0 < Velocity < (20 * Acceleration) and 0 < Velocity < (20 * Deceleration)	User unit/sec	No default	Maximum velocity of the defined path. See Limitations on Acceleration and Jerk for more information.
Acceleration	LREAL	0 < Velocity < (20 * Acceleration)	User unit/sec^2	No default	Maximum acceleration. See Limitations on Acceleration and Jerk for more information.
Deceleration	LREAL	0 < Velocity < (20 * Deceleration)	User unit/sec^2	No default	Maximum deceleration. See Limitations on Acceleration and Jerk for more information.
Jerk	LREAL	**Trapezoidal velocity profiles**: 0 (zero) **S-Curve velocity profiles**: (Velocity / 20) < Acceleration < (2 * Jerk) and (Velocity / 20) < Deceleration < (2 * Jerk)	User unit/sec^3	No default	Maximum jerk. See Limitations on Acceleration and Jerk for more information.

Input	Data Type	Range	Unit	Default	Description
CoordSystem	SINT	One of these enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2	N/A	No default	The coordinate system used when commanding the circular absolute move. • Currently, only the ACS coordinate system is supported. • See Coordinate Systems for more information.
BufferMode	SINT One of these enumeration values: • MC_BUFFER_MODE_BUFFERED = 1 = Buffered • MC_BUFFER_MODE_BLENDING_PREVIOUS = 2 = Blending Previous • MC_BUFFER_MODE_BLENDING_NEXT = 3 = Blending Next • MC_BUFFER_MODE_BLENDING_LOW = 4 = Blending Low • MC_BUFFER_MODE_BLENDING_HIGH = 5 = Blending High	No range	N/A	No default	Defines the chronological sequence of the function block relative to the previous block. • BufferMode = Abort = 0 is not allowed with this function block. • See Buffer Modes for more information. • The blending modes (2, 3, 4, & 5) match the **path velocity** at the active move's endpoint. • Some individual **axis velocities** can make an abrupt change if the path of the next move travels in a different direction. • A transition move can be programmed in the `TransitionMode` input to avoid this.

Input	Data Type	Range	Unit	Default	Description
TransitionMode	SINT	The value is limited to: • MC_TRANSITION_MODE_NONE = 0 • MC_TRANSITION_MODE_CORNER_DISTANCE = 3	N/A	No default	Coupled with the TransitionParameter[ ], this input defines the shape and dynamics of the inserted contour to connect the current motion with the next motion in the queue. See Transition Between Moves for more information.
TransitionParameter[ ]	LREAL	1, N	N/A	No default	This array is dependent on the specified TransitionMode. • N values are supplier specified dependent on the selected TransitionMode. • Transition parameter values are applied to the axis group. • See Transition Mode Parameters for more information.

4.3.18.2.4.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	If TRUE, the command completed successfully.
Busy	BOOL	No range	N/A	If TRUE, the function block is executing.
Active	BOOL	No range	N/A	If TRUE, the function block is controlling motion.
CommandAborted	BOOL	No range	N/A	If TRUE, the command was aborted by another function block.
Error	BOOL	No range	N/A	If TRUE, an error has occurred.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.18.2.4.3 Remarks

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- See Circular Moves Diagrams for more information about movement options.

- An error is returned if:
  - The group is in the GroupDisabled state.
  - The input parameters do not meet the required precision.
    - See Precision Requirements for Circular Move Input Parameters for more information.
- When all motion has completed successfully, the axes group state is GroupStandby.

NOTE

- Circular motion is only supported for axes groups with only two attached axes.
- S-Curve motion is not currently supported.
- The Jerk input is currently ignored.
- S-Curve motion and the Jerk argument will be supported in a future release.

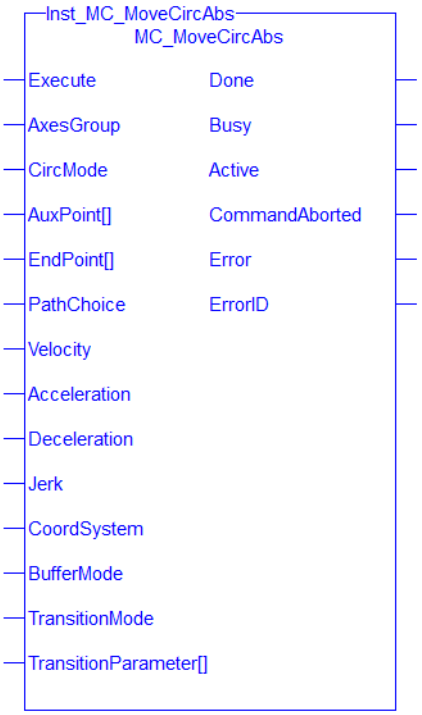
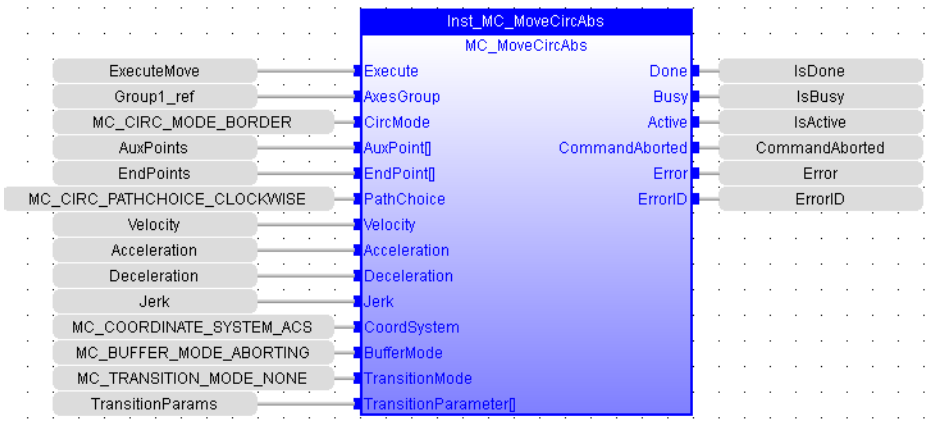
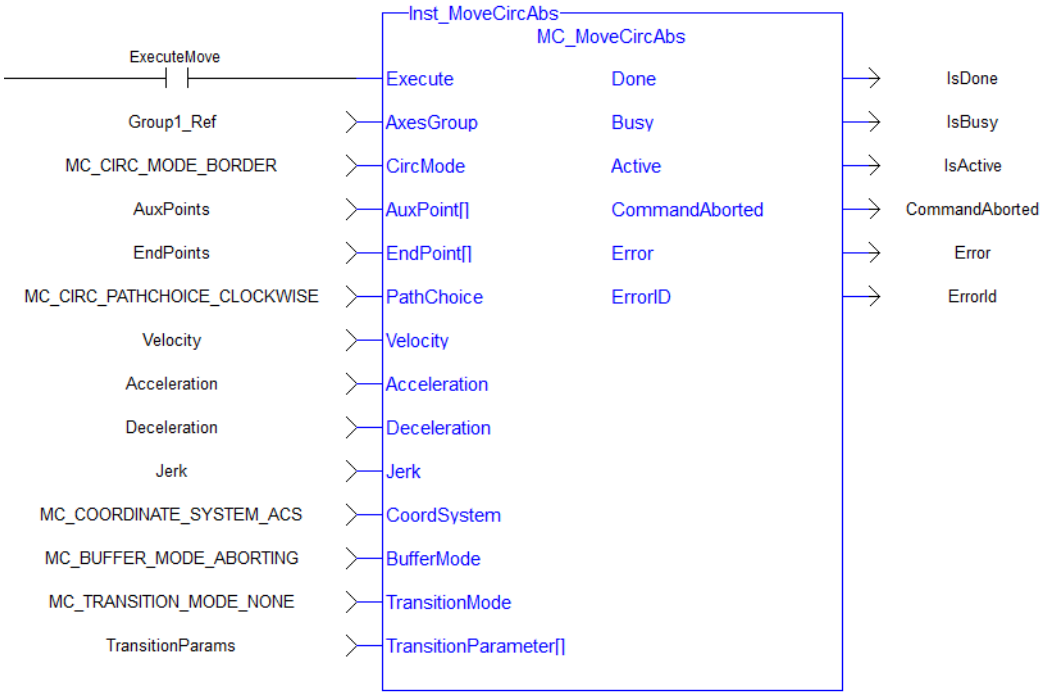


Figure 4-140: MC_MoveCircAbs

4.3.18.2.4.4 FBD Language Example



4.3.18.2.4.5 FFLD Language Example



4.3.18.2.4.6 IL Language Example

```
      BEGIN_IL

     CAL Inst_MC_MoveCircAbs( ExecuteMove, Group1_Ref, MC_CIRC_MODE_BORDER,
AuxPoints, EndPoints, MC_CIRC_PATHCHOICE_CLOCKWISE, Velocity, Acceleration,
Deceleration, Jerk, MC_COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_ABORTING, MC_
TRANSITION_MODE_NONE, TransitionParams )
END_IL
```

4.3.18.2.4.7 ST Language Example

```
Inst_MC_MoveCircAbs( ExecuteMove, Group1_Ref, MC_CIRC_MODE_BORDER, AuxPoints,
EndPoints, MC_CIRC_PATHCHOICE_CLOCKWISE, Velocity, Acceleration,
Deceleration, Jerk, MC_COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_ABORTING, MC_
TRANSITION_MODE_NONE, TransitionParams );
```

See Also

- "MC_ErrorDescription" (→ p. 533)
- "MC_MoveCircRel" (→ p. 623)

4.3.18.2.5 MC_MoveCircRel

PLCopen ✓ Pipe Network ✓

Function Block - Commands interpolated circular movement on an axes group to the specified relative positions in the coordinate system as specified by the CoordSystem argument.

4.3.18.2.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to perform a circular relative move.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axis group that performs the circular relative move.
CircMode	SINT One of these enumeration values: • MC_CIRC_MODE_BORDER = 0 • MC_CIRC_MODE_CENTER = 1	No range	N/A	No default	Specifies the meaning of the AuxPoint[] input.

Input	Data Type	Range	Unit	Default	Description

AuxPoint[]	LREAL	0, Number of axes in group - 1	N/A	No default	**Required Precision**
					• 1 part in 100,000.
					• See Precision Requirements for Circular Move Input Parameters for more information.
					Description
					Array of relative positions for each axis in the group.
					In all cases the points are relative to the starting point.
					The meaning depends on the value of the CircMode input:
					• MC_CIRC_MODE_BORDER
					  • AuxPoint defines a point on the circle which is crossed on the path from the starting to the end point.
					• MC_CIRC_MODE_CENTER
					  • AuxPoint defines the center point of the circle.

Input	Data Type	Range	Unit	Default	Description
EndPoint[]	LREAL	0, Number of axes in group - 1	N/A	No default	**Required Precision** • 1 part in 100,000. • See Precision Requirements for Circular Move Input Parameters for more information. **Description** Array of relative end positions for each axis in the group.
PathChoice	SINT One of these enumeration values: • MC_CIRC_PATHCHOICE_CLOCKWISE = 0 = Clockwise • MC_CIRC_PATHCHOICE_COUNTERCLOCKWISE = 1 = Counterclockwise	No range	N/A	No default	Specifies the direction of the path. This argument is only relevant when **CircMode** is MC_CIRC_MODE_CENTER.
Velocity	LREAL	0 < Velocity < (20 * Acceleration) and 0 < Velocity < (20 * Deceleration)	User unit/sec	No default	Maximum velocity of the defined path. See Limitations on Acceleration and Jerk for more information.
Acceleration	LREAL	0 < Velocity < (20 * Acceleration)	User unit/sec^2	No default	Maximum acceleration. See Limitations on Acceleration and Jerk for more information.
Deceleration	LREAL	0 < Velocity < (20 * Deceleration)	User unit/sec^2	No default	Maximum deceleration. See Limitations on Acceleration and Jerk for more information.

Input	Data Type	Range	Unit	Default	Description
Jerk	LREAL	**Trapezoidal velocity profiles**: 0 (zero) **S-Curve velocity profiles**: (Velocity / 20) < Acceleration < (2 * Jerk) and (Velocity / 20) < Deceleration < (2 * Jerk)	User unit/sec^3	No default	Maximum jerk. See Limitations on Acceleration and Jerk for more information.
CoordSystem	SINT	One of these enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2	N/A	No default	The coordinate system used when commanding the circular relative move. • Currently, only the ACS coordinate system is supported. • See Coordinate Systems for more information.

Input	Data Type	Range	Unit	Default	Description
BufferMode	SINT One of these enumeration values: • MC_BUFFER_MODE_ABORTING = 0 = Abort • MC_BUFFER_MODE_BUFFERED = 1 = Buffered • MC_BUFFER_MODE_BLENDING_PREVIOUS = 2 = Blending Previous • MC_BUFFER_MODE_BLENDING_NEXT = 3 = Blending Next • MC_BUFFER_MODE_BLENDING_LOW = 4 = Blending Low • MC_BUFFER_MODE_BLENDING_HIGH = 5 = Blending High	No range	N/A	No default	Defines the chronological sequence of the function block relative to the previous block. • See Buffer Modes for more information. • The blending modes (2, 3, 4, & 5) match the **path velocity** at the active move's endpoint. • Some individual **axis velocities** can make an abrupt change if the path of the next move travels in a different direction. • A transition move can be programmed in the `TransitionMode` input to avoid this.
TransitionMode	SINT	The value is limited to: • MC_TRANSITION_MODE_NONE = 0 • MC_TRANSITION_MODE_CORNER_DISTANCE = 3	N/A	No default	Coupled with the TransitionParameter[ ], this input defines the shape and dynamics of the inserted contour to connect the current motion with the next motion in the queue. See Transition Between Moves for more information.

Input	Data Type	Range	Unit	Default	Description
TransitionParameter[ ]	LREAL	1, N	N/A	No default	This array is dependent on the specified TransitionMode. • N values are supplier specified dependent on the selected TransitionMode. • Transition parameter values are applied to the axis group. • See Transition Mode Parameters for more information.

4.3.18.2.5.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	If TRUE, the command completed successfully.
Busy	BOOL	No range	N/A	If TRUE, the function block is executing.
Active	BOOL	No range	N/A	If TRUE, the function block is controlling motion.
CommandAborted	BOOL	No range	N/A	If TRUE, the command was aborted by another function block.
Error	BOOL	No range	N/A	If TRUE, an error has occurred.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.18.2.5.3 Remarks

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- See Circular Moves Diagrams for more information about movement options.
- An error is returned if the group is in the GroupDisabled state.
- When all motion has completed successfully, the axes group state is GroupStandby.

NOTE

- Circular motion is only supported for axes groups with only two attached axes.
- S-Curve motion is not currently supported.
- The Jerk input is currently ignored.
- S-Curve motion and the Jerk argument will be supported in a future release.

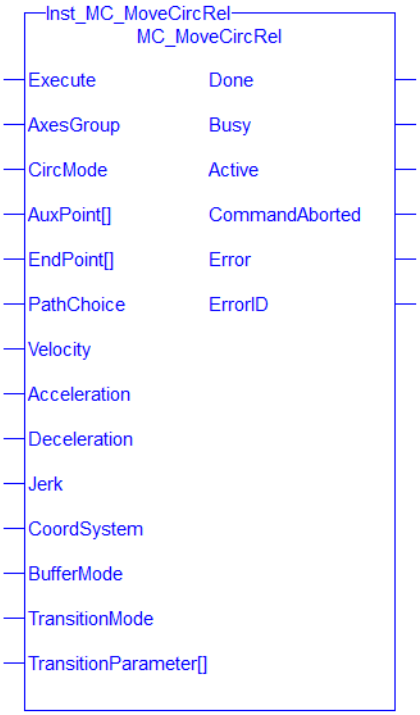
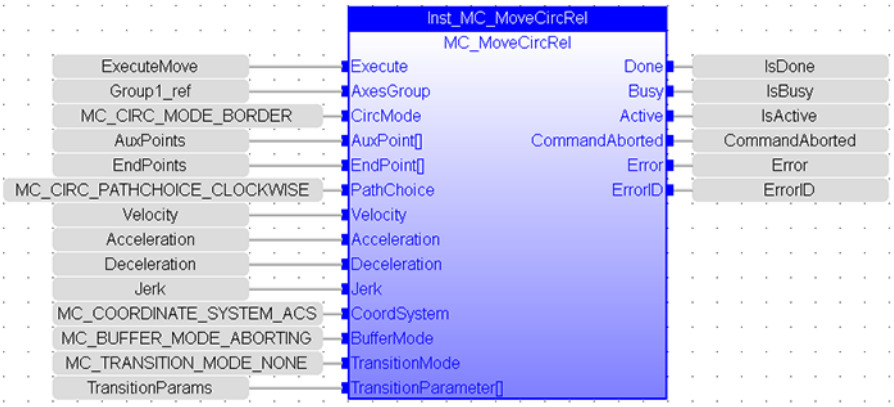
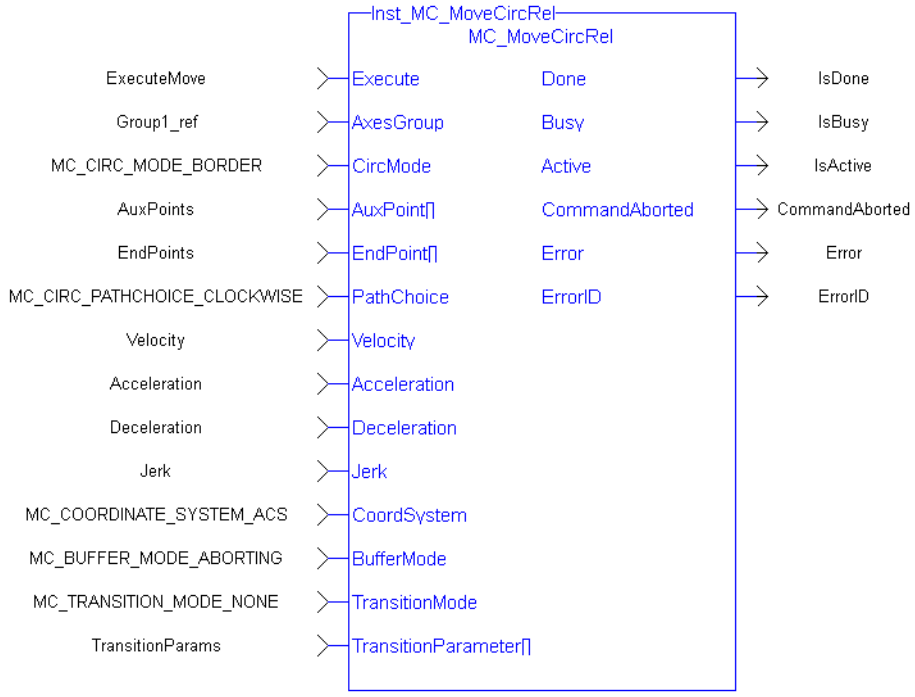


Figure 4-141: MC_MoveCircRel

4.3.18.2.5.4 FBD Language Example



4.3.18.2.5.5 FFLD Language Example



4.3.18.2.5.6 IL Language Example

```
      BEGIN_IL
     CAL Inst_MC_MoveCircRel( ExecuteMove, Group1_Ref, MC_CIRC_MODE_BORDER,
AuxPoints, EndPoints, MC_CIRC_PATHCHOICE_CLOCKWISE, Velocity, Acceleration,
Deceleration, Jerk, MC_COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_ABORTING, MC_
TRANSITION_MODE_NONE, TransitionParams )
END_IL
```

4.3.18.2.5.7 ST Language Example

```
Inst_MC_MoveCircRel( ExecuteMove, Group1_Ref, MC_CIRC_MODE_BORDER, AuxPoints,
EndPoints, MC_CIRC_PATHCHOICE_CLOCKWISE, Velocity, Acceleration,
Deceleration, Jerk, MC_COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_ABORTING, MC_
TRANSITION_MODE_NONE, TransitionParams );
```

See Also

- "MC_ErrorDescription" (→ p. 533)
- "MC_MoveCircAbs" (→ p. 617)

4.3.18.2.6 MC_MoveDirAbs

PLCopen ✓ Pipe Network ✓

Function Block - Commands movement of an axes group to an absolute position regardless of path.

4.3.18.2.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to perform a direct absolute move.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	Reference to an axes group.
Position[]	LREAL	0, Number of axes in group - 1	N/A	No default	Array of absolute end positions for each axis in the group.
CoordSystem	SINT	One of these enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2	N/A	No default	The coordinate system used when commanding the direct absolute move. • Currently, only the ACS coordinate system is supported. • See Coordinate Systems for more information.
BufferMode	SINT One of these enumeration values: • MC_BUFFER_MODE_ABORTING = 0 = Abort • MC_BUFFER_MODE_BUFFERED = 1 = Buffered	No range	N/A	No default	Defines the chronological sequence of the function block relative to the previous block. See Buffer Modes for more information.

4.3.18.2.6.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	If TRUE, the command completed successfully.
Busy	BOOL	No range	N/A	If TRUE, the function block is executing.
Active	BOOL	No range	N/A	If TRUE, the function block is controlling motion.
CommandAborted	BOOL	No range	N/A	If TRUE, the command was aborted by another function block.
Error	BOOL	No range	N/A	If TRUE, an error has occurred.
ErrorID	INT	No range	N/A	Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.18.2.6.3 Remarks

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- When all motion has completed successfully, the axes group state is GroupStandby.
- **MC_MoveDirAbs** commands the movement of an axes group to a specified absolute position in the specified coordinate system regardless of how (on which path) the target position is reached.

NOTE

- An error is returned if the group is in the GroupDisabled state.
- This function block does not have its own Acceleration, Deceleration, Velocity, and Jerk arguments.
  - These are set using "MC_AxisSetDefaults" (→ p. 610).
- The maximum number of axes is set by the **MaxNumberOfAxes** input set in the "Example: Valid Setup" (→ p. 563) function block.

NOTE

- Circular motion is only supported for axes groups with only two attached axes.
- S-Curve motion is not currently supported.
- The Jerk input is currently ignored.
- S-Curve motion and the Jerk argument will be supported in a future release.

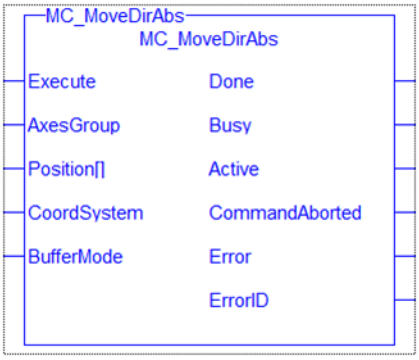
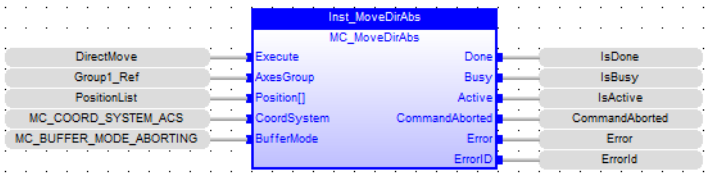
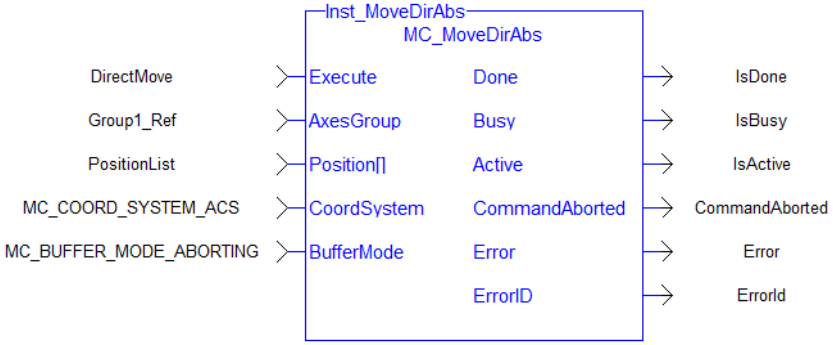


Figure 4-142: MC_MoveDirAbs

4.3.18.2.6.4 FBD Language Example



4.3.18.2.6.5 FFLD Language Example



4.3.18.2.6.6 IL Language Example

```
            BEGIN_IL
     CAL Inst_MC_MoveDirAbs( DirectMove, Group1_Ref, PositionList, MC_COORD_
SYSTEM_ACS, MC_BUFFER_MODE_ABORTING)
END_IL
```

4.3.18.2.6.7 ST Language Example

```
Inst_MC_MoveDirAbs( DirectMove, Group1_Ref, PositionList, MC_COORDSYSTEM_ACS,
MC_BUFFER_MODE_ABORTING);
```

See Also

- "MC_ErrorDescription" (→ p. 533)
- "MC_MoveDirRel" (→ p. 634)

4.3.18.2.7 MC_MoveDirRel

PLCopen ✓ Pipe Network ✓

Function Block - Commands movement of an axes group to a relative position regardless of path.

4.3.18.2.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to perform a direct relative move.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	Reference to an axes group.
Distance[]	LREAL	0, Number of axes in group - 1	N/A	No default	An array containing the distance for each axis in the group.
CoordSystem	SINT	One of these enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2	N/A	No default	The coordinate system used when commanding the direct relative move. • Currently, only the ACS coordinate system is supported. • See Coordinate Systems for more information.

Input	Data Type	Range	Unit	Default	Description
BufferMode	SINT One of these enumeration values: • MC_BUFFER_MODE_ABORTING = 0 = Abort • MC_BUFFER_MODE_BUFFERED = 1 = Buffered	No range	N/A	No default	Defines the chronological sequence of the function block relative to the previous block. See Buffer Modes for more information.

4.3.18.2.7.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Busy	BOOL			If TRUE, the function block is executing.
Active	BOOL			If TRUE, the function block is controlling motion.
CommandAborted	BOOL			If TRUE, the command was aborted by another function block.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	INT			Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.18.2.7.3 Remarks

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- When all motion has completed successfully, the axes group state is GroupStandby.
- **MC_MoveDirRel** commands a movement of an axes group to a relative position in the specified coordinate system regardless of how (on which path) the target position is reached.

NOTE

- An error is returned if the group is in the GroupDisabled state.
- This function block does not have its own Acceleration, Deceleration, Velocity, and Jerk arguments.
  - These are set using "MC_AxisSetDefaults" (→ p. 610).
- The maximum number of axes is set by the **MaxNumberOfAxes** input set in the "Example: Valid Setup" (→ p. 563) function block.

NOTE

- Circular motion is only supported for axes groups with only two attached axes.
- S-Curve motion is not currently supported.
- The Jerk input is currently ignored.
- S-Curve motion and the Jerk argument will be supported in a future release.

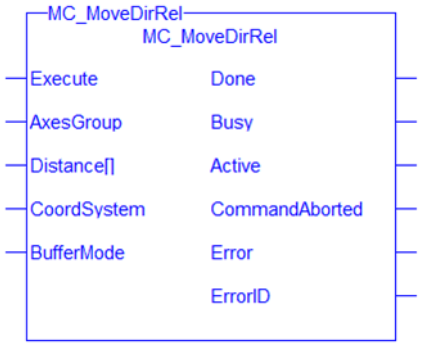
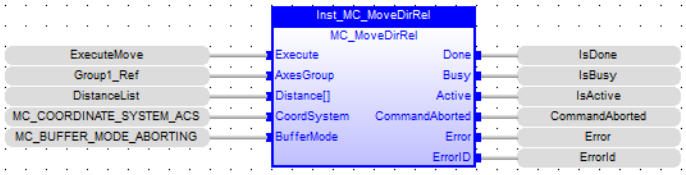
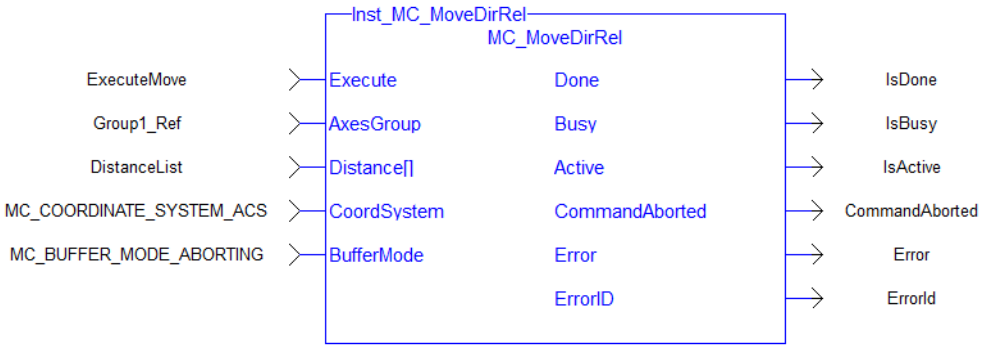


Figure 4-143: MC_MoveDirRel

4.3.18.2.7.4 FBD Language Example



4.3.18.2.7.5 FFLD Language Example



4.3.18.2.7.6 IL Language Example

```
            BEGIN_IL
     CAL Inst_MC_MoveDirRel( ExecuteMove, Group1_Ref, DistanceList, MC_
COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_ABORTING )
END_IL
```

4.3.18.2.7.7 ST Language Example

```
Inst_MC_MoveDirRel( ExecuteMove, Group1_Ref, DistanceList, MC_COORDINATE_
SYSTEM_ACS, MC_BUFFER_MODE_ABORTING );
```

See Also

- "MC_ErrorDescription" (→ p. 533)
- "MC_MoveDirAbs" (→ p. 631)

4.3.18.2.8 MC_MoveLinAbs

PLCopen ✓ Pipe Network ✓

Function Block - Commands interpolated linear movement on an axes group to the specified absolute positions.

4.3.18.2.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to perform a linear absolute move.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axis group that performs the linear absolute move.
Position[]	LREAL	No range	User units	No default	Array of absolute end positions for each axis in the group.
Velocity	LREAL	0 < Velocity < (20 * Acceleration) and 0 < Velocity < (20 * Deceleration)	User unit/sec	No default	Maximum velocity of the defined path. See Limitations on Acceleration and Jerk for more information.
Acceleration	LREAL	0 < Velocity < (20 * Acceleration)	User unit/sec^2	No default	Maximum acceleration. See Limitations on Acceleration and Jerk for more information.
Deceleration	LREAL	0 < Velocity < (20 * Deceleration)	User unit/sec^2	No default	Maximum deceleration. See Limitations on Acceleration and Jerk for more information.
Jerk	LREAL	**Trapezoidal velocity profiles**: 0 (zero) **S-Curve velocity profiles**: (Velocity / 20) < Acceleration < (2 * Jerk) and (Velocity / 20) < Deceleration < (2 * Jerk)	User unit/sec^3	No default	Maximum jerk. See Limitations on Acceleration and Jerk for more information.

Input	Data Type	Range	Unit	Default	Description
CoordSystem	SINT	One of these enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2	N/A	No default	The coordinate system used when commanding the linear absolute move. • Currently, only the ACS coordinate system is supported. • See Coordinate Systems for more information.

Input	Data Type	Range	Unit	Default	Description
BufferMode	SINT One of these enumeration values: • MC_BUFFER_MODE_ABORTING = 0 = Abort • MC_BUFFER_MODE_BUFFERED = 1 = Buffered • MC_BUFFER_MODE_BLENDING_PREVIOUS = 2 = Blending Previous • MC_BUFFER_MODE_BLENDING_NEXT = 3 = Blending Next • MC_BUFFER_MODE_BLENDING_LOW = 4 = Blending Low • MC_BUFFER_MODE_BLENDING_HIGH = 5 = Blending High	No range	N/A	No default	Defines the chronological sequence of the function block relative to the previous block. • The buffer mode is limited to MC_BUFFER_MODE_BUFFERED for groups with more than two axes. • See Buffer Modes for more information. • The blending modes (2, 3, 4, & 5) match the **path velocity** at the active move's endpoint. • Some individual **axis velocities** can make an abrupt change if the path of the next move travels in a different direction. • A transition move can be programmed in the `Transition Mode` input to avoid this.

Input	Data Type	Range	Unit	Default	Description
TransitionMode	SINT	The value is limited to: • MC_TRANSITION_MODE_NONE = 0 • MC_TRANSITION_MODE_CORNER_DISTANCE = 3	N/A	No default	Coupled with the TransitionParameter[ ], this input defines the shape and dynamics of the inserted contour to connect the current motion with the next motion in the queue. • The transition mode is limited to MC_TRANSITION_MODE_NONE for groups with more than three axes. • See Transition Between Moves for more information.
TransitionParameter[ ]	LREAL	1, N	N/A	No default	This array is dependent on the specified TransitionMode. • N values are supplier specified dependent on the selected TransitionMode. • Transition parameter values are applied to the axis group. • See Transition Mode Parameters for more information.

4.3.18.2.8.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Busy	BOOL			If TRUE, the function block is executing.
Active	BOOL			If TRUE, the function block is controlling motion.
CommandAborted	BOOL			If TRUE, the command was aborted by another function block.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	INT			Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.18.2.8.3 Remarks

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- When all motion has completed successfully, the axes group state is GroupStandby.
- **MC_MoveLinAbs** commands interpolated linear movement on an axes group to the specified absolute positions in the coordinate system as specified by the CoordSystem argument.
  - The dimensionality of the move is determined by the number of axes mapped to the group.

NOTE

- An error is returned if the group is in the GroupDisabled state.
- The maximum number of axes is set by the **MaxNumberOfAxes** input set in the "Example: Valid Setup" (→ p. 563) function block.

NOTE

- Circular motion is only supported for axes groups with only two attached axes.
- S-Curve motion is not currently supported.
- The Jerk input is currently ignored.
- S-Curve motion and the Jerk argument will be supported in a future release.

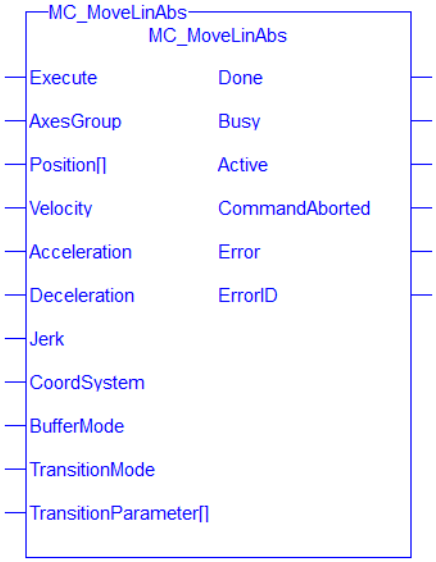
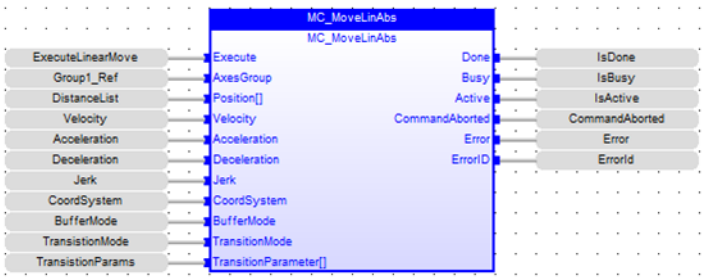
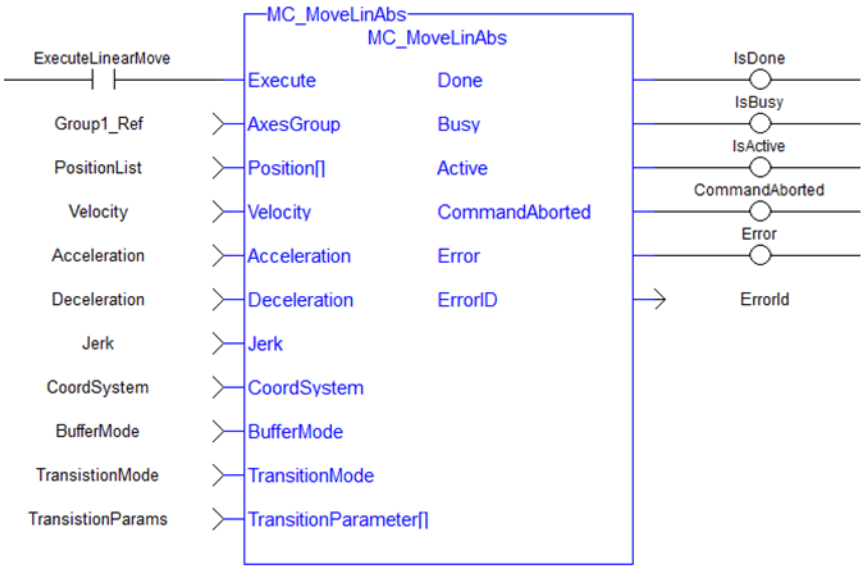


Figure 4-144: MC_MoveLinAbs

4.3.18.2.8.4 FBD Language Example



4.3.18.2.8.5 FFLD Language Example



4.3.18.2.8.6 IL Language Example

```
      BEGIN_IL
     CAL Inst_MC_MoveLinAbs( ExecuteLinearMove, Group1_Ref, PositionList,
Velocity, Acceleration, Deceleration, Jerk, CoordSystem, BufferMode,
TransitionMode, TransitionParams )
END_IL
```

4.3.18.2.8.7 ST Language Example

```
      (* Inst_MC_MoveLinAbsST example *)
Inst_MC_MoveLinAbs( ExecuteLinearMove, Group1_Ref, PositionList, Velocity,
Acceleration, Deceleration, Jerk, CoordSystem, BufferMode, TransitionMode,
TransitionParams );
```

See Also

- "MC_ErrorDescription" (→ p. 533)
- "MC_MoveLinRel" (→ p. 642)

4.3.18.2.9 MC_MoveLinRel

PLCopen ✓ Pipe Network ✓

Function Block - Commands interpolated linear movement on an axes group to the specified relative positions.

4.3.18.2.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to perform a linear relative move.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axis group that performs the linear relative move.

Input	Data Type	Range	Unit	Default	Description
Distance[]	LREAL	No range	User units	No default	Array of distances for each axis in the group.
Velocity	LREAL	0 < Velocity < (20 * Acceleration) and 0 < Velocity < (20 * Deceleration)	User unit/sec	No default	Maximum velocity of the defined path. See Limitations on Acceleration and Jerk for more information.
Acceleration	LREAL	0 < Velocity < (20 * Acceleration)	User unit/sec^2	No default	Maximum acceleration. See Limitations on Acceleration and Jerk for more information.
Deceleration	LREAL	0 < Velocity < (20 * Deceleration)	User unit/sec^2	No default	Maximum deceleration. See Limitations on Acceleration and Jerk for more information.
Jerk	LREAL	**Trapezoidal velocity profiles**: 0 (zero) **S-Curve velocity profiles**: (Velocity / 20) < Acceleration < (2 * Jerk) and (Velocity / 20) < Deceleration < (2 * Jerk)	User unit/sec^3	No default	Maximum jerk. See Limitations on Acceleration and Jerk for more information.
CoordSystem	SINT	One of these enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2	N/A	No default	The coordinate system used when commanding the linear relative move. • Currently, only the ACS coordinate system is supported. • See Coordinate Systems for more information.

Input	Data Type	Range	Unit	Default	Description
BufferMode	SINT One of these enumeration values: • MC_BUFFER_MODE_ABORTING = 0 = Abort • MC_BUFFER_MODE_BUFFERED = 1 = Buffered • MC_BUFFER_MODE_BLENDING_PREVIOUS = 2 = Blending Previous • MC_BUFFER_MODE_BLENDING_NEXT = 3 = Blending Next • MC_BUFFER_MODE_BLENDING_LOW = 4 = Blending Low • MC_BUFFER_MODE_BLENDING_HIGH = 5 = Blending High	No range	N/A	No default	Defines the chronological sequence of the function block relative to the previous block. • The buffer mode is limited to MC_BUFFER_MODE_BUFFERED for groups with more than two axes. • See Buffer Modes for more information. • The blending modes (2, 3, 4, & 5) match the **path velocity** at the active move's endpoint. • Some individual **axis velocities** can make an abrupt change if the path of the next move travels in a different direction. • A transition move can be programmed in the `Transition Mode` input to avoid this.

Input	Data Type	Range	Unit	Default	Description
TransitionMode	SINT	The value is limited to: • MC_TRANSITION_MODE_NONE = 0 • MC_TRANSITION_MODE_CORNER_DISTANCE = 3	N/A	No default	Coupled with the TransitionParameter[ ], this input defines the shape and dynamics of the inserted contour to connect the current motion with the next motion in the queue. • The transition mode is limited to MC_TRANSITION_MODE_NONE for groups with more than three axes. • See Transition Between Moves for more information.
TransitionParameter[ ]	LREAL	1, N	N/A	No default	This array is dependent on the specified TransitionMode. • N values are supplier specified dependent on the selected TransitionMode. • Transition parameter values are applied to the axis group. • See Transition Mode Parameters for more information.

4.3.18.2.9.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Busy	BOOL			If TRUE, the function block is executing.
Active	BOOL			If TRUE, the function block is controlling motion.
CommandAborted	BOOL			If TRUE, the command was aborted by another function block.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	INT			Indicates the error if **Error** output is TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.18.2.9.3 Remarks

- See Coordinated Motion, the top-level topic for Coordinated Motion.
- The dimensionality of the move is determined by the number of axes mapped to the group.

- When all motion has completed successfully, the axes group state is GroupStandby.
- MC_MoveLinRel commands interpolated linear movement of an axes group to the specified relative positions.

NOTE

- An error is returned if the group is in the GroupDisabled state.
- The maximum number of axes is set by the **MaxNumberOfAxes** input set in the "Example: Valid Setup" (→ p. 563) function block.

NOTE

- Circular motion is only supported for axes groups with only two attached axes.
- S-Curve motion is not currently supported.
- The Jerk input is currently ignored.
- S-Curve motion and the Jerk argument will be supported in a future release.

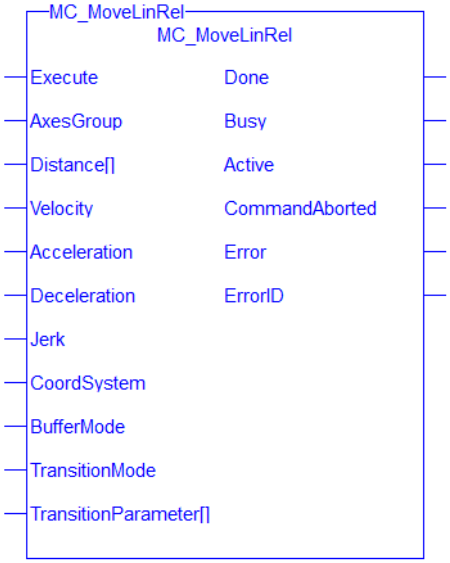
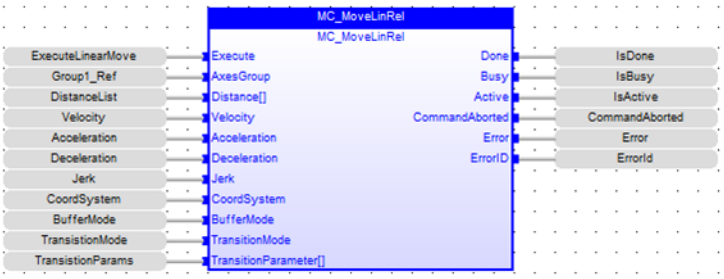
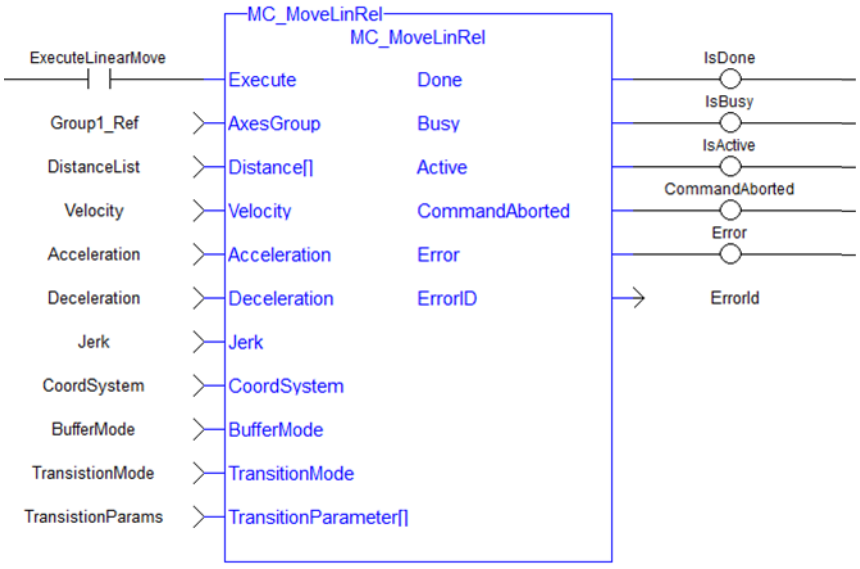


Figure 4-145: MC_MoveLinRel

4.3.18.2.9.4 FBD Language Example



4.3.18.2.9.5 FFLD Language Example



4.3.18.2.9.6 IL Language Example

```
BEGIN_IL
     CAL Inst_MC_MoveLinRel( ExecuteLinearMove, Group1_Ref, DistanceList,
Velocity, Acceleration, Deceleration, Jerk, CoordSystem, BufferMode,
TransitionMode, TransitionParams )
END_IL
```

4.3.18.2.9.7 ST Language Example

```
      (* Inst_MC_MoveLinRelST example *)

Inst_MC_MoveLinRel( ExecuteLinearMove, Group1_Ref, DistanceList, Velocity,
Acceleration, Deceleration, Jerk, CoordSystem, BufferMode, TranstionMode,
TransitionParams );
```

See Also

- "MC_ErrorDescription" (→ p. 533)
- "MC_MoveLinAbs" (→ p. 636)

4.3.18.3 Coordinated Motion - Reference Library

Name	Description
MC_ GrpSetPos	Sets the axis command position for all of the axes in an axes group to the positions specified in the `Position` input.

4.3.18.3.1 MC_GrpSetPos

PLCopen ✓ Pipe Network ✓

Function Block - Sets the axis command position for all of the axes in an axes group to the positions specified in the `Position` input.

4.3.18.3.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to set the position of the group.
AxesGroup	AXES_GROUP_REF	No range	N/A	No default	The axis group to set the positions.
Position[]	LREAL	0, Number of axes in group-1	N/A	No default	An array containing the position for each axis in the group. • If **Relative** is set, position represents a distance rather than an absolute position. • The length of the array must equal the maximum number of axes allowed in the group. • The maximum number of axes is an argument to MC_CreateAxesGrp, which is used to create axes groups.
Relative	BOOL	0, 1	N/A	No default	Request to set position: • (1) relative. • (0) absolute.
CoordSystem	SINT	One of these enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2	N/A	No default	The coordinate system used when setting the positions. • Currently, only the ACS coordinate system is supported. • See Coordinate Systems for more information.
BufferMode	SINT	0, 0	N/A	No default	Currently unused.

4.3.18.3.1.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Busy	BOOL			Currently unused. Returns FALSE.
CommandAborted	BOOL			Currently unused. Returns FALSE.
Error	BOOL			If TRUE, an error has occurred.

Output	Data Type	Range	Unit	Description
ErrorID	INT			Indicates the error if the **Error** output is set to TRUE. See the table in PLCopen Function Block ErrorIDs.

4.3.18.3.1.3 Remarks

NOTE

This function block starts a motion-related action and stores data for calculations and error checking. See Calling Function Blocks Multiple Times in the Same Cycle if using a dual-core controller.

- This function or function block does not generate any motion.
- See Coordinated Motion, the top-level topic for Coordinated Motion.
- The axes group must be enabled and in Standby mode for MC_GrpSetPos to execute.
  - If it is not, this FB returns an error and the axis positions remains unchanged.
  - The command position is that returned by the FB **MC_GrpReadCmdPos**.

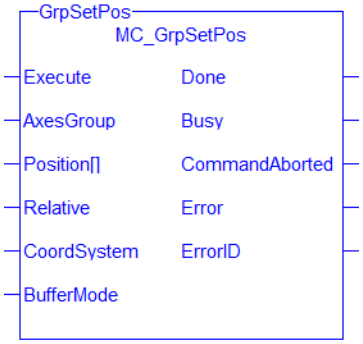
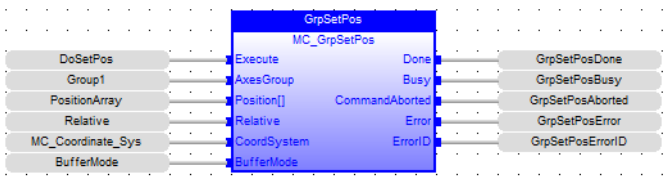
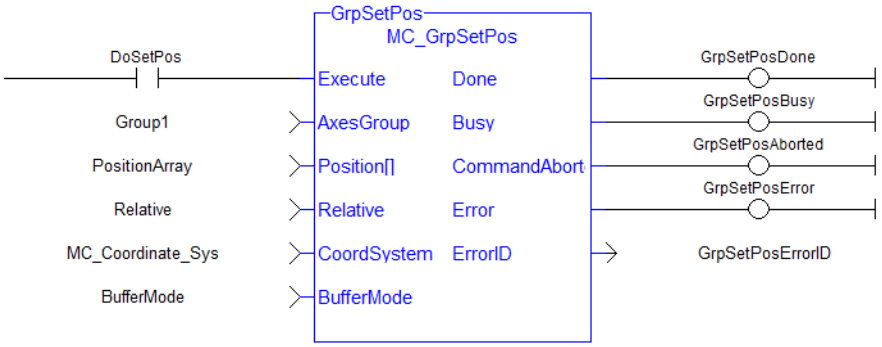


Figure 4-146: MC_GrpSetPos

4.3.18.3.1.4 FBD Language Example



4.3.18.3.1.5 FFLD Language Example



4.3.18.3.1.6 IL Language Example

```
            BEGIN_IL
     CAL   Inst_MC_GrpSetPos( DoSetPos, Group1, PositionArray, Relative, MC_
COORDINATE_SYSTEM_ACS, BufferMode);
END_IL
```

4.3.18.3.1.7 ST Language Example

```
Inst_MC_GrpSetPos( DoSetPos, Group1, PositionArray, Relative, MC_COORDINATE_
SYSTEM_ACS, 0 );
```

See Also

- "MC_ErrorDescription" (➔ p. 533)
- "MC_GrpReadCmdPos" (➔ p. 599)

5 System Library

These tables list the System functions and function blocks:

- "Controller" (→ p. 651)
- "File" (→ p. 651)
- "TCP/IP Sockets" (→ p. 651)
- "UDP" (→ p. 652)

See "PrintMessage" (→ p. 715) - Generates a log message with any wanted strings in the Log Messages window.

5.1 Controller

Name	Description
ClearCtrlErrors	Clears the list of active errors and alarms on the controller.
GetCtrlErrors	Get a list of the active errors and alarms on the controller.
GetCtrlInfo	Get the serial, model, and/or part number of the controller.
GetCtrlPerf	Returns controller CPU performance statistics.

5.2 File

Name	Description
FileClose	Closes an open file.
FileCopy	Copies a file's contents to a new file.
FileDelete	Removes a file from the file system.
FileEOF	Test if the end of the file is reached in a file that is open for reading.
FileExists	Tests if a file exists.
FileOpenA	Create or open a file in append mode.
FileOpenR	Open a file for reading.
FileOpenW	Create or reset a file and open it for writing.
FileReadBinData	Read binary data from a file.
FileReadLine	Reads a string value from a text file.
FileRename	Renames a file.
FileSeek	Sets the current position in an open file.
FileSize	Gets the size of a file.
FileWriteBinData	Write binary data to a file.
FileWriteLine	Writes a string value to a text file.

5.3 TCP/IP Sockets

Name	Description
TcpAccept	Performs the accept operation.
TcpBinReceive	Receives characters over a socket connection.
TcpBinSend	Sends characters over a socket.
TcpClose	Closes and releases a socket.
TcpConnect	Creates a new socket and performs the connect operation.
TcpIsConnected	Tests if a client socket is connected.
TcpIsValid	Tests if a socket is valid.
TcpListen	Creates a new socket by performing the bind and listen operations.
TcpReceive	Receives characters over a socket connection.
TcpSend	Sends characters over a socket.

5.4 UDP

Name	Description
udpAddrMake	Build an address buffer for UDP functions.
udpClose	Closes a socket.
udpCreate	Creates a UDP socket.
udpIsValid	Tests if a socket is valid.
udpRcvFrom	Receives a UDP telegram.
udpRcvFromArray	Receives a byte array through UDP.
udpRcvFromVar	Receives the contents of a variable from another controller through UDP.
udpSendTo	Sends a UDP telegram.
udpSendToArray	Sends a byte array through UDP.
udpSendToVar	Sends the contents of a local variable to another controller through UDP.

5.5 Controller Library

Name	Description
ClearCtrlErrors	Clears the list of active errors and alarms on the controller.
GetCtrlErrors	Get a list of the active errors and alarms on the controller.
GetCtrlInfo	Get the serial, model, and/or part number of the controller.
GetCtrlPerf	Returns controller CPU performance statistics.

5.5.1 ClearCtrlErrors

Function - Clears the list of active errors and alarms on the controller.

5.5.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN					Enable the function.

5.5.1.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			

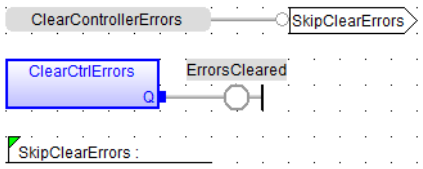
5.5.1.3 Remarks

- Only clearable errors will be cleared.
- See Errors for a list of errors and alarms that may be generated.

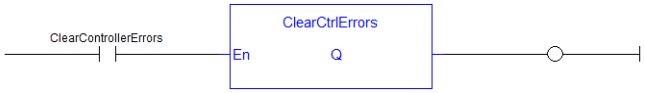
NOTE

If clearable and non-clearable errors are present and this function is called, the Output Q is turned to TRUE but the non-clearable errors remain.

5.5.1.4 FBD Language Example



5.5.1.5 FFLD Language Example



5.5.1.6 IL Language Example

Not available.

5.5.1.7 ST Language Example

```
//Attempt to clear active controller level errors and alarms(such as E33)
IF ClearControllerErrors THEN
ClearCtrlErrors();
END_IF;
```

5.5.2 GetCtrlErrors

PLCopen ✓ Pipe Network ✓

Function - Get a list of the active errors and alarms on the controller.

5.5.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL				Enable

Input	Data Type	Range	Unit	Default	Description
ActiveError	BOOL[100]				Array of BOOL with the size equal to 100.
ActiveAlarm	BOOL[100]				Array of BOOL with the size equal to 100.

5.5.2.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL			
Q	DINT			Status of the execution.

5.5.2.3 Remarks

- Returns active errors and alarms on the controller in two arrays of hundred Booleans.
- Every index in the array corresponds to the error and alarm numbers in the tables.
  - See Controller Errors and Alarms for a list of possible generated errors and alarms.

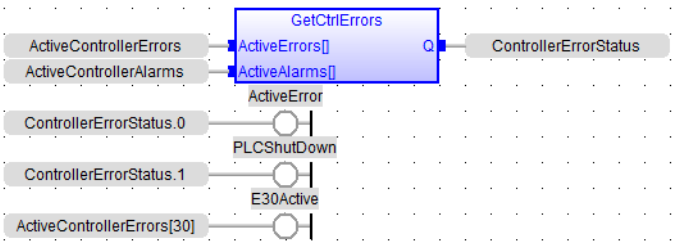
5.5.2.3.1 Status Meaning

Bit	Value	Description
Bit 0	Value 0	No error, no alarm, no shut down.
Bit 0	Value 1	There is an active error or an active alarm (i.e., there is something in the array ActiveError/ActiveAlarm).
Bit 1	Value 0	No shut down.
Bit 1	Value 1	The PLC processes is shut down. This starts 10 seconds after the error is triggered.
Bit 2-15	Value 2 | 4 | 9 to 2147483648	Reserved.

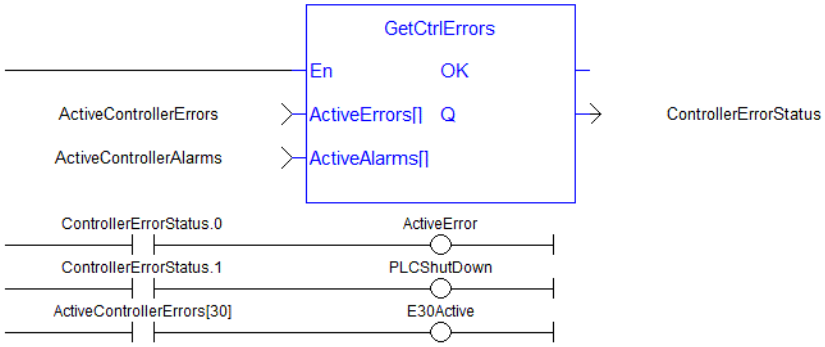
TIP

See Check for existing EtherCAT Alarms and Errors in the EtherCAT Communication Diagnosis Steps section for an example of implementing this function.

5.5.2.4 FBD Language Example



5.5.2.5 FFLD Language Example



5.5.2.6 IL Language Example

Not available.

5.5.2.7 ST Language Example

```
//Retrieve active controller level alarm and errors.
//Check status output to see if any error or alarm is active and if PLC is
shutting down
ControllerErrorStatus:= GetCtrlErrors( ActiveControllerErrors,
ActiveControllerAlarms);
ActiveError:= ControllerErrorStatus.0;
PLCShutDown:= ControllerErrorStatus.1;
E30Active:= ActiveControllerErrors[30];
```

5.5.3 GetCtrlInfo

PLCopen ✓ Pipe Network ✓

Function Block - Get the serial, model, and/or part number of the controller.

5.5.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	Rising edge of enable initiates read of parameter.

<table>
<tr><th>Input</th><th>Data Type</th><th>Range</th><th>Unit</th><th>Default</th><th>Description</th></tr>
<tr><td>Parameter Number</td><td>INT</td><td>1, 6</td><td>N/A</td><td>No default</td><td>Parameter number to read.
These parameters are also Internal Defines:
<table>
<tr><th>Value</th><th>Integer Representation</th><th>Description</th></tr>
<tr><td>CTRLINFO_SERIAL_NUMBER</td><td>1</td><td>Controller serial number.</td></tr>
<tr><td>CTRLINFO_MODEL_NUMBER</td><td>2</td><td>Controller model number.</td></tr>
<tr><td>CTRLINFO_PART_NUMBER</td><td>3</td><td>Controller part number.</td></tr>
<tr><td>CTRLINFO_CPU_CORE_COUNT</td><td>4</td><td>Number of CPU cores in the controller.</td></tr>
<tr><td>CTRLINFO_PROJECT_BUILD_NO</td><td>5</td><td>The project build number of the running application.</td></tr>
<tr><td>CTRLINFO_PROJECT_COMPILE_TIME</td><td>6</td><td>The project compile time of the running application.</td></tr>
<tr><td>CTRLINFO_MODEL_FAMILY</td><td>7</td><td>Controller model family.</td></tr>
</table></td></tr>
</table>

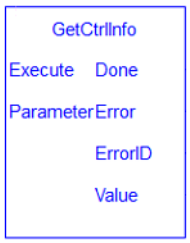
5.5.3.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	Indicates whether this function block has completed **without** error.
Error	BOOL	No range	N/A	Indicates whether this function block has completed **with** error.

<table>
<tr><th>Output</th><th>Data Type</th><th>Range</th><th>Unit</th><th>Description</th></tr>
<tr><td>ErrorID</td><td>INT</td><td>No range</td><td>N/A</td><td>Error value to indicate error condition.
These parameters are also Internal Defines:
<table>
<tr><th>Value</th><th>Integer Representation</th><th>Description</th></tr>
<tr><td>CTRLINFO_ERROR_NO_ERROR</td><td>0</td><td>No error.</td></tr>
<tr><td>CTRLINFO_ERROR_INV_PARAMETER</td><td>1</td><td>Invalid parameter.</td></tr>
<tr><td>CTRLINFO_ERROR_CANT_READ_DATA</td><td>2</td><td>Error reading data.</td></tr>
<tr><td>CTRLINFO_ERROR_NOT_PDMM</td><td>3</td><td>Not valid on a non-AKD PDMM, PCMM, or PCMM2G controller.</td></tr>
</table></td></tr>
<tr><td>Value</td><td>STRING</td><td>No range</td><td>N/A</td><td>String containing data that was read.
IF Parameter Number = CTRLINFO_MODEL_FAMILY (7):
<table>
<tr><th>Value</th><th>Integer Representation</th><th>Description</th></tr>
<tr><td>CTRLINFO_MODEL_FAMILY_PXMM</td><td>0</td><td>AKD PDMM or PCMM - 1st generation controller.</td></tr>
<tr><td>CTRLINFO_MODEL_FAMILY_PCMM2G</td><td>1</td><td>PCMM2G - 2nd generation controller.</td></tr>
<tr><td>CTRLINFO_MODEL_FAMILY_SIMULATOR</td><td>2</td><td>KAS Simulator</td></tr>
<tr><td>CTRLINFO_MODEL_FAMILY_UNKNOWN</td><td>3</td><td>Unknown controller family.</td></tr>
</table></td></tr>
</table>

5.5.3.3 Remarks

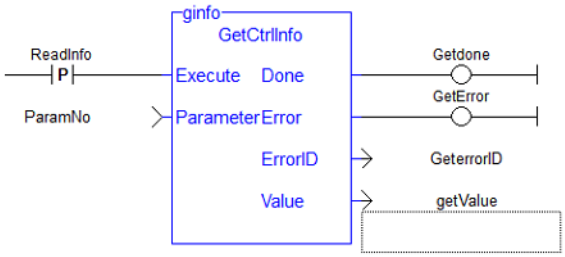
Returns a String containing the value of the control parameter requested.



5.5.3.4 FBD Language Example

Not available.

5.5.3.5 FFLD Language Example



5.5.3.6 IL Language Example

Not available.

5.5.3.7 ST Language Example

```
Inst_GetCtrlInfo( ExecuteRead, CTRLINFO_SERIAL_NUMBER);

if Inst_GetCtrlInfo.Done then
   serialNumber := Inst_GetCtrlInfo.Value;
end_if;
```

5.5.4 GetCtrlPerf

PLCopen ✓ Pipe Network ✓

Function Block - Returns controller CPU performance statistics.

5.5.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to collect the controller's performance data.
DataPointCount	UDINT	2 - 240,000	Cycles	No default	The number of motion manager cycles over which performance statistics are gathered.

5.5.4.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	No range	N/A	If TRUE, the command completed successfully.
Error	BOOL	No range	N/A	If TRUE, an error has occurred.
ErrorID	INT	No range	N/A	• ErrorID = 0 indicates no error. • ErrorID = 1 indicates an error.
MissedCycles	UDINT	No range	N/A	Indicates the number of missing VM cycles.
MeanVmTime	LREAL	≥ 0 (zero)	Microseconds	The mean VM execution time.

Output	Data Type	Range	Unit	Description
StdDevVmTime	LREAL	≥ 0 (zero)	Microseconds	The standard deviation of the VM execution time.
MinVmTime	LREAL	≥ 0 (zero)	Microseconds	The minimum VM execution time.
MaxVmTime	LREAL	≥ 0 (zero)	Microseconds	The maximum VM execution time.
MeanMotMgrTime	LREAL	≥ 0 (zero)	Microseconds	The mean motion manager execution time.
StdDevMotMgrTime	LREAL	≥ 0 (zero)	Microseconds	The standard deviation of the motion manager execution time.
MinMotMgrTime	LREAL	≥ 0 (zero)	Microseconds	The minimum motion manager execution time.
MaxMotMgrTime	LREAL	≥ 0 (zero)	Microseconds	The maximum motion manager execution time.

MeanCpuIdlePct	LREAL	0 to 100	Percent	The mean percentage of time the controller CPU is idle.
				• AKD PDMM and PCMM: CPU idle period is measured across all CPU cores.
				  • See Item 3, Channel 4: **AKD PDMM or PCMM - Real Time Margin** in the 5. The five buttons section for more information.
				• PCMM2G: CPU idle period is measured across the CPU cores that run the motion engine and PLC program.
				  • See Item 3, Channel 4: **PCMM2G - PLC Margin** in the 5. The five buttons section for more information.

StdDevCpuIdlePct	LREAL	0 to 100	N/A	• AKD PDMM and PCMM: CPU idle period is measured across all CPU cores.
				  • See Item 3, Channel 4: **AKD PDMM or PCMM - Real Time Margin** in the 5. The five buttons section for more information.
				• PCMM2G: CPU idle period is measured across the CPU cores that run the motion engine and PLC program.
				  • See Item 3, Channel 4: **PCMM2G - PLC Margin** in the 5. The five buttons section for more information.

Output	Data Type	Range	Unit	Description

MinCpuIdlePct	LREAL	0 to 100	N/A	• AKD PDMM and PCMM: CPU idle period is measured across all CPU cores.
				  • See Item 3, Channel 4: **AKD PDMM or PCMM - Real Time Margin** in the 5. The five buttons section for more information.
				• PCMM2G: CPU idle period is measured across the CPU cores that run the motion engine and PLC program.
				  • See Item 3, Channel 4: **PCMM2G - PLC Margin** in the 5. The five buttons section for more information.

MaxCpuIdlePct	LREAL	0 to 100	N/A	• AKD PDMM and PCMM: CPU idle period is measured across all CPU cores.
				  • See Item 3, Channel 4: **AKD PDMM or PCMM - Real Time Margin** in the 5. The five buttons section for more information.
				• PCMM2G: CPU idle period is measured across the CPU cores that run the motion engine and PLC program.
				  • See Item 3, Channel 4: **PCMM2G - PLC Margin** in the 5. The five buttons section for more information.

5.5.4.3 Remarks

Generates a text file with performance statistics of the controller.

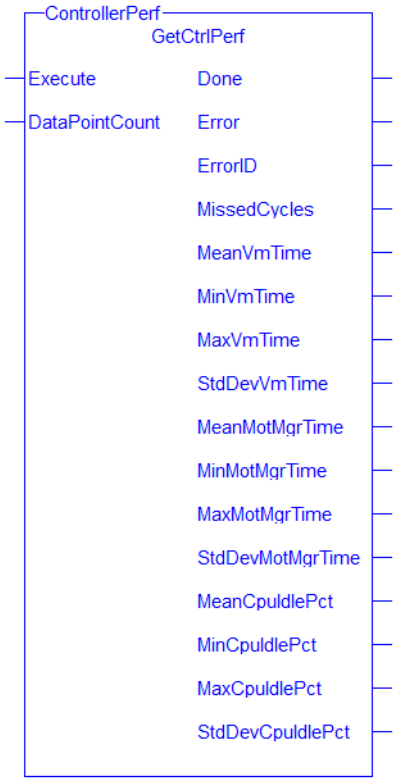
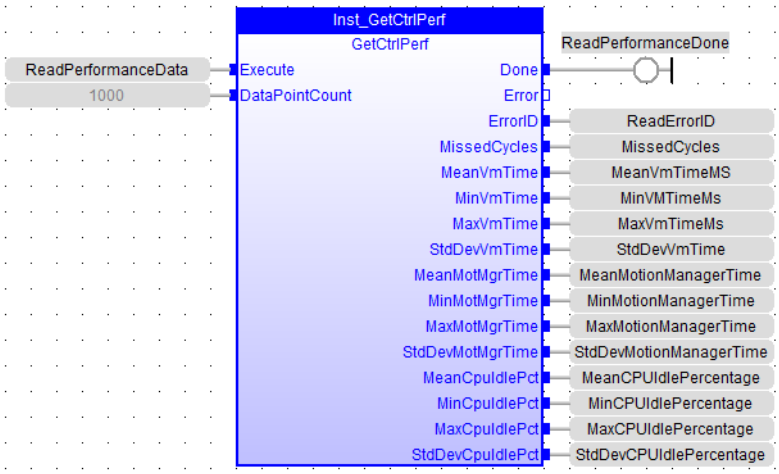
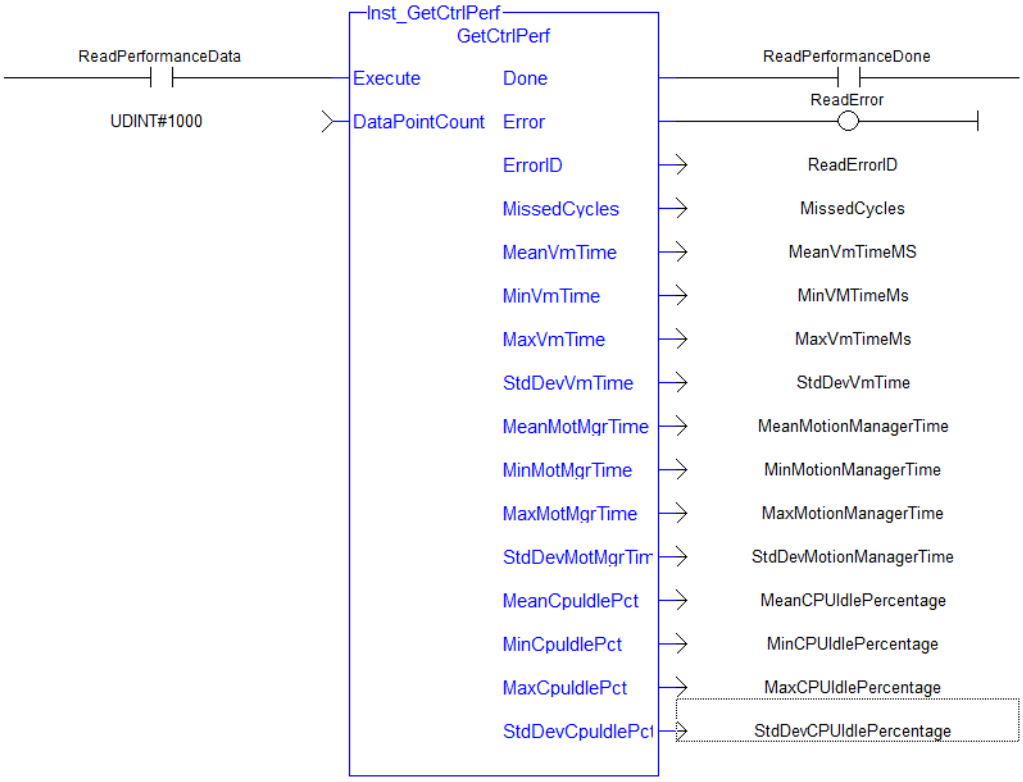


Figure 5-1: GetCtrlPerf

5.5.4.4 FBD Language Example



5.5.4.5 FFLD Language Example



5.5.4.6 IL Language Example

Not available.

5.5.4.7 ST Language Example

```
//Read controller performance data from last 1000 cycles (1 second at T#1ms
update rate)
Inst_GetCtrlPerf( ReadPerformanceData, 1000 );

IF Inst_GetCtrlInfo.Done THEN
   MissedCycles:= Inst_GetCtrlPerf.MissedCycles;
   MaxVmTimeMs:= Inst_GetCtrlPerf.MaxVmTime;
   MaxMotionManagerTime:= Inst_GetCtrlPerf.MaxMotMgrTime;
   MeanCPUIdlePercentage:= Inst_GetCtrlPerf.MeanCpuIdlePct;
   MaxCPUIdlePercentage:= Inst_GetCtrlPerf.MaxCpuIdlePct;
END_IF;
```

See Also

- Call a Function Block
- Differences Between Functions and Function Blocks

5.6 File Tools Function Blocks

This section documents function blocks that can be applied to files.

Name	Description
FileClose	Closes an open file.
FileCopy	Copies a file's contents to a new file.

Name	Description
FileDelete	Removes a file from the file system.
FileEOF	Test if the end of the file is reached in a file that is open for reading.
FileExists	Tests if a file exists.
FileOpenA	Create or open a file in append mode.
FileOpenR	Open a file for reading.
FileOpenW	Create or reset a file and open it for writing.
FileReadBinData	Read binary data from a file.
FileReadLine	Reads a string value from a text file.
FileRename	Renames a file.
FileSeek	Sets the current position in an open file.
FileSize	Gets the size of a file.
FileWriteBinData	Write binary data to a file.
FileWriteLine	Writes a string value to a text file.

5.6.1 FileClose

PLCopen

Function Block - Closes an open file.

5.6.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to perform closing a file.
ID	UDINT	N/A	N/A	No default	The ID of the open file.

5.6.1.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).

5.6.1.3 Remarks

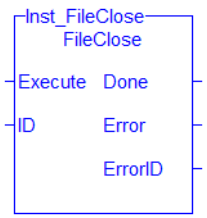


Figure 5-2: FileClose

5.6.1.4 FBD Language Example

Not available.

5.6.1.5 FFLD Language Example

Not available.

5.6.1.6 IL Language Example

Not available.

5.6.1.7 ST Language Example

```
(* FileClose example *)
CASE StepCounter OF
0:
   Inst_FileClose(TRUE, MyOutputFileID);
   StepCounter := StepCounter + 1;
1:
   Inst_FileClose(TRUE, MyOutputFileID);
   IF Inst_FileClose.Done THEN
      Inst_FileClose(FALSE, 0);
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- File Management
- "FileCopy" (→ p. 664)
- "FileDelete" (→ p. 666)
- "FileEOF" (→ p. 668)
- "FileExists" (→ p. 669)
- "FileOpenA" (→ p. 671)
- "FileOpenR" (→ p. 673)
- "FileOpenW" (→ p. 675)
- "FileReadBinData" (→ p. 677)
- "FileReadLine" (→ p. 678)
- "FileRename" (→ p. 680)
- "FileSeek" (→ p. 682)
- "FileSize" (→ p. 683)
- "FileWriteBinData" (→ p. 685)
- "FileWriteLine" (→ p. 686)

5.6.2 FileCopy

PLCopen

Function Block - Copies a file's contents to a new file.

5.6.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to perform copying a file.
Src	STRING	N/A	N/A	No default	The path to the source file.
Dst	STRING	N/A	User units	No default	The destination path to the source file.

5.6.2.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).

5.6.2.3 Remarks

- Large files take a noticeable amount of time to complete.
  - Example: A 1000KB file takes approximately 0.6 seconds.

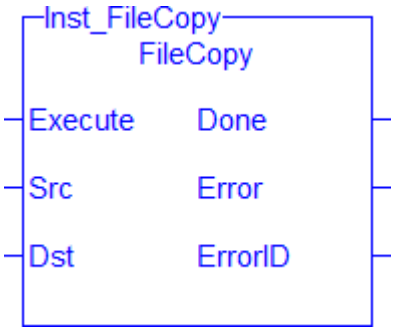


Figure 5-3: FileCopy

5.6.2.4 FBD Language Example

Not available.

5.6.2.5 FFLD Language Example

Not available.

5.6.2.6 IL Language Example

Not available.

5.6.2.7 ST Language Example

```
(* FileCopy example *)
CASE StepCounter OF
0:
   Inst_FileCopy(TRUE, 'Source.txt', 'Dest.txt');
   StepCounter := StepCounter + 1;
```

```
1:
   Inst_FileCopy(TRUE, 'Source.txt', 'Dest.txt');
   IF Inst_FileCopy.Done THEN
      Inst_FileCopy(FALSE, '', '');
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- File Management
- "FileClose" (→ p. 663)
- "FileDelete" (→ p. 666)
- "FileEOF" (→ p. 668)
- "FileExists" (→ p. 669)
- "FileOpenA" (→ p. 671)
- "FileOpenR" (→ p. 673)
- "FileOpenW" (→ p. 675)
- "FileReadBinData" (→ p. 677)
- "FileReadLine" (→ p. 678)
- "FileRename" (→ p. 680)
- "FileSeek" (→ p. 682)
- "FileSize" (→ p. 683)
- "FileWriteBinData" (→ p. 685)
- "FileWriteLine" (→ p. 686)

5.6.3 FileDelete

PLCopen

Function Block - Removes a file from the file system.

5.6.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to perform deletion of a file.
Path	STRING	N/A	N/A	No default	The path to the file.

5.6.3.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).

5.6.3.3 Remarks

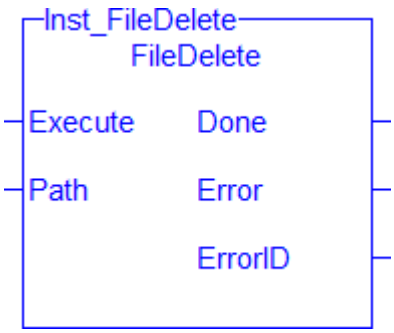


Figure 5-4: FileDelete

5.6.3.4 FBD Language Example

Not available.

5.6.3.5 FFLD Language Example

Not available.

5.6.3.6 IL Language Example

Not available.

5.6.3.7 ST Language Example

```
(* FileDelete example *)
CASE StepCounter OF
0:
   Inst_FileDelete(TRUE, 'Test.txt');
   StepCounter := StepCounter + 1;
1:
   Inst_FileDelete(TRUE, 'Test.txt');
   IF Inst_FileDelete.Done THEN
      Inst_FileDelete(FALSE, '');
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- File Management
- "FileClose" (→ p. 663)
- "FileCopy" (→ p. 664)
- "FileEOF" (→ p. 668)
- "FileExists" (→ p. 669)
- "FileOpenA" (→ p. 671)
- "FileOpenR" (→ p. 673)
- "FileOpenW" (→ p. 675)
- "FileReadBinData" (→ p. 677)
- "FileReadLine" (→ p. 678)
- "FileRename" (→ p. 680)
- "FileSeek" (→ p. 682)
- "FileSize" (→ p. 683)
- "FileWriteBinData" (→ p. 685)
- "FileWriteLine" (→ p. 686)

5.6.4 FileEOF

PLCopen

Function Block - Test if the end of the file is reached in a file that is open for reading.

5.6.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, test if the end of the file is reached.
ID	UDINT	N/A	N/A	No default	The ID of the open file.

5.6.4.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).
EOF	BOOL			TRUE if the end of the file has been encountered.

5.6.4.3 Remarks

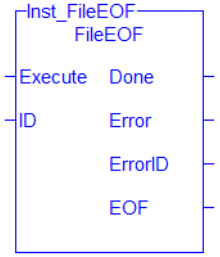


Figure 5-5: FileCopy

5.6.4.4 FBD Language Example

Not available.

5.6.4.5 FFLD Language Example

Not available.

5.6.4.6 IL Language Example

Not available.

5.6.4.7 ST Language Example

```
(* FileEOF example *)
CASE StepCounter OF
0:
   Inst_FileEOF(TRUE, MyInputFileID);
   StepCounter := StepCounter + 1;
1:
   Inst_FileEOF(TRUE, MyInputFileID);
   IF Inst_FileEOF.Done THEN
      EofReached := Inst_FileEOF.EOF;
      Inst_FileEOF(FALSE, 0);
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- File Management
- "FileClose" (→ p. 663)
- "FileCopy" (→ p. 664)
- "FileDelete" (→ p. 666)
- "FileExists" (→ p. 669)
- "FileOpenA" (→ p. 671)
- "FileOpenR" (→ p. 673)
- "FileOpenW" (→ p. 675)
- "FileReadBinData" (→ p. 677)
- "FileReadLine" (→ p. 678)
- "FileRename" (→ p. 680)
- "FileSeek" (→ p. 682)
- "FileSize" (→ p. 683)
- "FileWriteBinData" (→ p. 685)
- "FileWriteLine" (→ p. 686)

5.6.5 FileExists

PLCopen

Function Block - Tests if a file exists.

5.6.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, test to see if a file exists.
Path	STRING	N/A	N/A	No default	The path to the file.

5.6.5.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).
Exists	BOOL			The existence of the file. • TRUE: Indicates the file exists • FALSE: Indicates the file does not exist.

5.6.5.3 Remarks

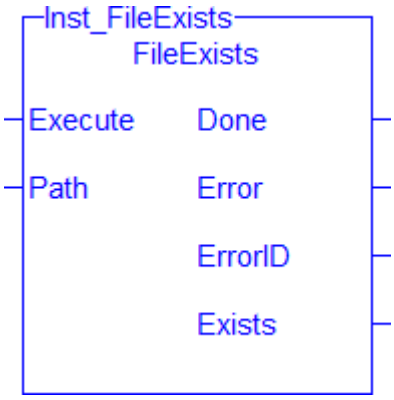


Figure 5-6: FileExists

5.6.5.4 FBD Language Example

Not available.

5.6.5.5 FFLD Language Example

Not available.

5.6.5.6 IL Language Example

Not available.

5.6.5.7 ST Language Example

```
(* FileExists example *)
CASE StepCounter OF
0:
   Inst_FileExists(TRUE, 'Test.txt');
   StepCounter := StepCounter + 1;
1:
   Inst_FileExists(TRUE, 'Test.txt');
```

```
    IF Inst_FileExists.Done THEN
       TestFileExists := Inst_FileExists.Exists;
       Inst_FileExists(FALSE, '');
       StepCounter := StepCounter + 1;
    END_IF;
END_CASE;
```

See Also

- File Management
- "FileClose" (→ p. 663)
- "FileCopy" (→ p. 664)
- "FileDelete" (→ p. 666)
- "FileEOF" (→ p. 668)
- "FileOpenA" (→ p. 671)
- "FileOpenR" (→ p. 673)
- "FileOpenW" (→ p. 675)
- "FileReadBinData" (→ p. 677)
- "FileReadLine" (→ p. 678)
- "FileRename" (→ p. 680)
- "FileSeek" (→ p. 682)
- "FileSize" (→ p. 683)
- "FileWriteBinData" (→ p. 685)
- "FileWriteLine" (→ p. 686)

5.6.6 FileOpenA

Function Block - Create or open a file in append mode.

5.6.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to open a file.
Path	STRING	N/A	N/A	No default	The path to the file.

5.6.6.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).
ID	UDINT			The ID of the open file.

5.6.6.3 Remarks

NOTE

The controller allows only 32 open files at any given time.

Files must be closed using the "FileClose" (→ p. 663) function block.

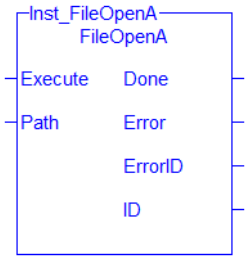


Figure 5-7: FileOpenA

5.6.6.4 FBD Language Example

Not available.

5.6.6.5 FFLD Language Example

Not available.

5.6.6.6 IL Language Example

Not available.

5.6.6.7 ST Language Example

```
(* FileOpenW example *)
(* Replace FileOpenW with FileOpenA or FileOpenR as needed*)
CASE StepCounter OF
0:
   Inst_FileOpenW(TRUE, 'Results.txt');
   StepCounter := StepCounter + 1;
1:
   Inst_FileOpenW(TRUE, 'Results.txt');
   IF Inst_FileOpenW.Done THEN
      ResultsFileID := Inst_FileOpenW.ID;
      Inst_FileOpenW(FALSE, '');
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- File Management
- "FileClose" (→ p. 663)
- "FileCopy" (→ p. 664)
- "FileDelete" (→ p. 666)
- "FileEOF" (→ p. 668)
- "FileExists" (→ p. 669)
- "FileOpenR" (→ p. 673)
- "FileOpenW" (→ p. 675)
- "FileReadBinData" (→ p. 677)
- "FileReadLine" (→ p. 678)
- "FileRename" (→ p. 680)
- "FileSeek" (→ p. 682)
- "FileSize" (→ p. 683)
- "FileWriteBinData" (→ p. 685)
- "FileWriteLine" (→ p. 686)

5.6.7 FileOpenR

Function Block - Open a file for reading.

5.6.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to open a file.
Path	STRING	N/A	N/A	No default	The path to the file.

5.6.7.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).
ID	UDINT			The ID of the open file.

5.6.7.3 Remarks

NOTE

The controller allows only 32 open files at any given time.

Files must be closed using the "FileClose" (→ p. 663) function block.

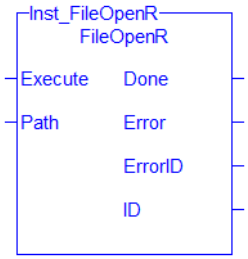


Figure 5-8: FileOpenR

5.6.7.4 FBD Language Example

Not available.

5.6.7.5 FFLD Language Example

Not available.

5.6.7.6 IL Language Example

Not available.

5.6.7.7 ST Language Example

```
(* FileOpenW example *)
(* Replace FileOpenW with FileOpenA or FileOpenR as needed*)
CASE StepCounter OF
0:
   Inst_FileOpenW(TRUE, 'Results.txt');
   StepCounter := StepCounter + 1;
1:
   Inst_FileOpenW(TRUE, 'Results.txt');
   IF Inst_FileOpenW.Done THEN
      ResultsFileID := Inst_FileOpenW.ID;
      Inst_FileOpenW(FALSE, '');
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- File Management
- "FileClose" (→ p. 663)
- "FileCopy" (→ p. 664)
- "FileDelete" (→ p. 666)
- "FileEOF" (→ p. 668)
- "FileExists" (→ p. 669)
- "FileOpenA" (→ p. 671)
- "FileOpenW" (→ p. 675)
- "FileReadBinData" (→ p. 677)
- "FileReadLine" (→ p. 678)
- "FileRename" (→ p. 680)
- "FileSeek" (→ p. 682)
- "FileSize" (→ p. 683)
- "FileWriteBinData" (→ p. 685)
- "FileWriteLine" (→ p. 686)

5.6.8 FileOpenW

Function Block - Create or reset a file and open it for writing.

5.6.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to open a file.
Path	STRING	N/A	N/A	No default	The path to the file.

5.6.8.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).
ID	UDINT			The ID of the open file.

5.6.8.3 Remarks

NOTE

The controller allows only 32 open files at any given time.

- If a file already exists, it is overwritten.
- Files must be closed using the "FileClose" (→ p. 663) function block.

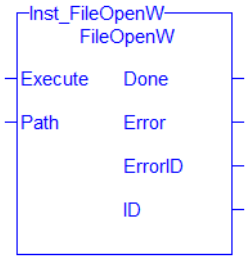


Figure 5-9: FileOpenW

5.6.8.4 FBD Language Example

Not available.

5.6.8.5 FFLD Language Example

Not available.

5.6.8.6 IL Language Example

Not available.

5.6.8.7 ST Language Example

```
(* FileOpenW example *)
(* Replace FileOpenW with FileOpenA or FileOpenR as needed*)
CASE StepCounter OF
0:
   Inst_FileOpenW(TRUE, 'Results.txt');
   StepCounter := StepCounter + 1;
1:
   Inst_FileOpenW(TRUE, 'Results.txt');
   IF Inst_FileOpenW.Done THEN
      ResultsFileID := Inst_FileOpenW.ID;
      Inst_FileOpenW(FALSE, '');
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- File Management
- "FileClose" (→ p. 663)
- "FileCopy" (→ p. 664)
- "FileDelete" (→ p. 666)
- "FileEOF" (→ p. 668)
- "FileExists" (→ p. 669)
- "FileOpenA" (→ p. 671)
- "FileOpenR" (→ p. 673)
- "FileReadBinData" (→ p. 677)
- "FileReadLine" (→ p. 678)
- "FileRename" (→ p. 680)
- "FileSeek" (→ p. 682)
- "FileSize" (→ p. 683)
- "FileWriteBinData" (→ p. 685)
- "FileWriteLine" (→ p. 686)

5.6.9 FileReadBinData

PLCopen

Function Block - Read binary data from a file.

5.6.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, read the size of a file.
ID	UDINT	N/A	N/A	No default	The ID of the open file.
Frame	USINT[ ]	N/A	N/A	No default	Where the binary data is stored.
FrameSize	DINT	N/A	N/A	No default	Number of bytes to store in the **Frame** array.

5.6.9.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).
NbRead	STRING			The number of bytes read from the file.

5.6.9.3 Remarks

- **FileReadBinData** stops reading from the file if it fills the passed **Frame** array, reads **FrameSize** bytes, or encounters the end of file, whichever comes first.
- After a successful call to **FileReadBinData**, use SerializeIn to extract variable data from the binary data read from a file.

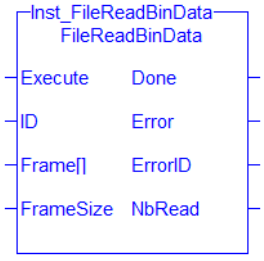


Figure 5-10: FileReadBinData

5.6.9.4 FBD Language Example

Not available.

5.6.9.5 FFLD Language Example

Not available.

5.6.9.6 IL Language Example

Not available.

5.6.9.7 ST Language Example

```
(* FileReadBinData example *)
CASE StepCounter OF
0:
   Inst_FileReadBinData(TRUE, MyInputFileID, InputFrame, 128);
   StepCounter := StepCounter + 1;
1:
   Inst_FileReadBinData(TRUE, MyInputFileID, InputFrame, 128);
   IF Inst_FileReadBinData.Done THEN
      BytesRead := Inst_FileReadBinData.NbRead;
      Inst_FileReadBinData(FALSE, 0, InputFrame, 0);
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- File Management
- "FileClose" (→ p. 663)
- "FileCopy" (→ p. 664)
- "FileDelete" (→ p. 666)
- "FileEOF" (→ p. 668)
- "FileExists" (→ p. 669)
- "FileOpenA" (→ p. 671)
- "FileOpenR" (→ p. 673)
- "FileOpenW" (→ p. 675)
- "FileReadLine" (→ p. 678)
- "FileRename" (→ p. 680)
- "FileSeek" (→ p. 682)
- "FileSize" (→ p. 683)
- "FileWriteBinData" (→ p. 685)
- "FileWriteLine" (→ p. 686)

5.6.10 FileReadLine

PLCopen

Function Block - Reads a string value from a text file.

5.6.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, read the size of a file.
ID	UDINT	N/A	N/A	No default	The ID of the open file.

5.6.10.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.

Output	Data Type	Range	Unit	Description
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).
Value	STRING			The string value read from the file.

5.6.10.3 Remarks

FileReadLine stops reading from the file if either:

- 255 characters are read (the maximum length of the STRING type).
- A new line is encountered.

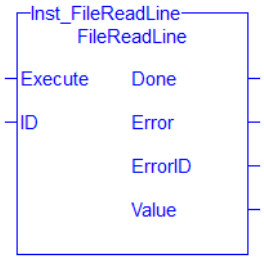


Figure 5-11: FileReadLine

5.6.10.4 FBD Language Example

Not available.

5.6.10.5 FFLD Language Example

Not available.

5.6.10.6 IL Language Example

Not available.

5.6.10.7 ST Language Example

```
(* FileReadLine example *)
CASE StepCounter OF
0:
   Inst_FileReadLine(TRUE, MyInputFileID);
   StepCounter := StepCounter + 1;
1:
   Inst_FileReadLine(TRUE, MyInputFileID);
   IF Inst_FileReadLine.Done THEN
      lineText := Inst_FileReadLine.Value;
      Inst_FileReadLine(FALSE, 0);
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- File Management
- "FileClose" (→ p. 663)
- "FileCopy" (→ p. 664)
- "FileDelete" (→ p. 666)
- "FileEOF" (→ p. 668)
- "FileExists" (→ p. 669)
- "FileOpenA" (→ p. 671)
- "FileOpenR" (→ p. 673)
- "FileOpenW" (→ p. 675)
- "FileReadBinData" (→ p. 677)
- "FileRename" (→ p. 680)
- "FileSeek" (→ p. 682)
- "FileSize" (→ p. 683)
- "FileWriteBinData" (→ p. 685)
- "FileWriteLine" (→ p. 686)

5.6.11 FileRename

Function Block - Renames a file.

5.6.11.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to rename a file.
Path	STRING	N/A	N/A	No default	The path to the file.
NewName	STRING	N/A	User units	No default	The new name of the file.

5.6.11.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).

5.6.11.3 Remarks

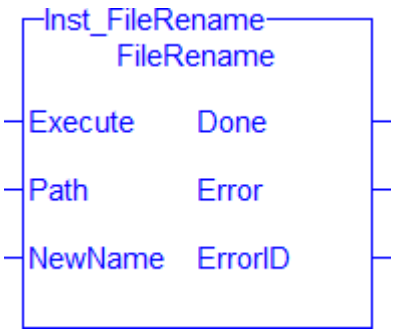


Figure 5-12: FileRename

5.6.11.4 FBD Language Example

Not available.

5.6.11.5 FFLD Language Example

Not available.

5.6.11.6 IL Language Example

Not available.

5.6.11.7 ST Language Example

```
(* FileRename example *)
CASE StepCounter OF
0:
   Inst_FileRename(TRUE, 'Original.txt', 'Renamed.txt');
   StepCounter := StepCounter + 1;
1:
   Inst_FileRename(TRUE, 'Original.txt', 'Renamed.txt');
   IF Inst_FileRename.Done THEN
      Inst_FileRename(FALSE, '', '');
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- File Management
- "FileClose" (→ p. 663)
- "FileCopy" (→ p. 664)
- "FileDelete" (→ p. 666)
- "FileEOF" (→ p. 668)
- "FileExists" (→ p. 669)
- "FileOpenA" (→ p. 671)
- "FileOpenR" (→ p. 673)
- "FileOpenW" (→ p. 675)
- "FileReadBinData" (→ p. 677)
- "FileReadLine" (→ p. 678)
- "FileSeek" (→ p. 682)
- "FileSize" (→ p. 683)
- "FileWriteBinData" (→ p. 685)
- "FileWriteLine" (→ p. 686)

5.6.12 FileSeek

PLCopen

Function Block - Sets the current position in an open file.

5.6.12.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, test if the end of the file is reached in a file that is open for reading.
ID	UDINT	N/A	N/A	No default	The ID of the open file.
POS	DINT	N/A	N/A	No default	Number of bytes to offset from ORG. • If ORG = SEEK_SET, then POS should be ≥ 0 (zero). • If ORG = SEEK_END, then POS should be ≤ 0 (zero). • If ORG = SEEK_CUR, then POS can be positive or negative.
ORG	DINT	• SEEK_SET • SEEK_CUR • SEEK_END	N/A	No default	Origin of the move. • SEEK_SET = Beginning of the file. • SEEK_CUR = Current position. • SEEK_END = End of the file.

5.6.12.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).

5.6.12.3 Remarks

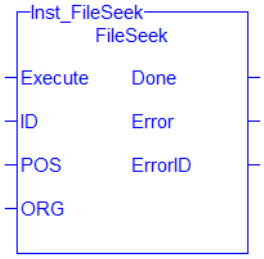


Figure 5-13: FileCopy

5.6.12.4 FBD Language Example

Not available.

5.6.12.5 FFLD Language Example

Not available.

5.6.12.6 IL Language Example

Not available.

5.6.12.7 ST Language Example

```
(* FileSeek example *)
CASE StepCounter OF
0:
   (* Move to beginning of the file *)
   Inst_FileSeek(TRUE, MyInputFileID, 0, SEEK_SET);
   StepCounter := StepCounter + 1;
1:
   Inst_FileSeek(TRUE, MyInputFileID, 0, SEEK_SET);
   IF Inst_FileSeek.Done THEN
      Inst_FileSeek(FALSE, 0, 0, SEEK_SET);
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- File Management
- "FileClose" (→ p. 663)
- "FileCopy" (→ p. 664)
- "FileDelete" (→ p. 666)
- "FileEOF" (→ p. 668)
- "FileExists" (→ p. 669)
- "FileOpenA" (→ p. 671)
- "FileOpenR" (→ p. 673)
- "FileOpenW" (→ p. 675)
- "FileReadBinData" (→ p. 677)
- "FileReadLine" (→ p. 678)
- "FileRename" (→ p. 680)
- "FileSize" (→ p. 683)
- "FileWriteBinData" (→ p. 685)
- "FileWriteLine" (→ p. 686)

5.6.13 FileSize

PLCopen

Function Block - Gets the size of a file.

5.6.13.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, read the size of a file.
Path	STRING	N/A	N/A	No default	The path to the file.

5.6.13.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).
Size	DINT			The size of the file in bytes.

5.6.13.3 Remarks

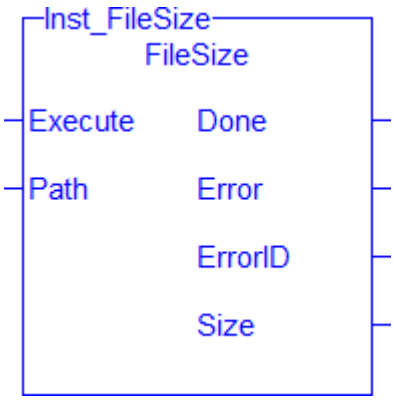


Figure 5-14: FileSize

5.6.13.4 FBD Language Example

Not available.

5.6.13.5 FFLD Language Example

Not available.

5.6.13.6 IL Language Example

Not available.

5.6.13.7 ST Language Example

```
(* FileSize example *)
CASE StepCounter OF
0:
   Inst_FileSize(TRUE, 'Test.txt');
   StepCounter := StepCounter + 1;
1:
   Inst_FileSize(TRUE, 'Test.txt');
   IF Inst_FileSize.Done THEN
      TestFileSize := Inst_FileSize.Size;
      Inst_FileSize(FALSE, '');
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- File Management
- "FileClose" (→ p. 663)
- "FileCopy" (→ p. 664)
- "FileDelete" (→ p. 666)
- "FileEOF" (→ p. 668)
- "FileExists" (→ p. 669)
- "FileOpenA" (→ p. 671)
- "FileOpenR" (→ p. 673)
- "FileOpenW" (→ p. 675)
- "FileReadBinData" (→ p. 677)
- "FileReadLine" (→ p. 678)
- "FileRename" (→ p. 680)
- "FileSeek" (→ p. 682)
- "FileWriteBinData" (→ p. 685)
- "FileWriteLine" (→ p. 686)

5.6.14 FileWriteBinData

PLCopen

Function Block - Write binary data to a file.

5.6.14.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, write binary data to a file.
ID	UDINT	N/A	N/A	No default	The ID of the open file.
Frame	USINT[ ]	N/A	N/A	No default	The array of binary data to write.
FrameSize	DINT	N/A	N/A	No default	Number of bytes to write from the **Frame** array.

5.6.14.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).

5.6.14.3 Remarks

Before using **FileWriteBinData**, use SerializeOut to copy variable data to a binary frame.

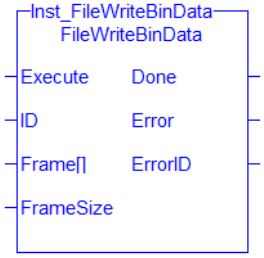


Figure 5-15: FileWriteLine

5.6.14.4 FBD Language Example

Not available.

5.6.14.5 FFLD Language Example

Not available.

5.6.14.6 IL Language Example

Not available.

5.6.14.7 ST Language Example

```
(* FileWriteBinData example *)
CASE StepCounter OF
0:
   Inst_FileWriteBinData(TRUE, MyOutputFileID, OutputFrame, 128);
   StepCounter := StepCounter + 1;
1:
   Inst_FileWriteBinData(TRUE, MyOutputFileID, OutputFrame, 128);
   IF Inst_FileWriteBinData.Done THEN
      Inst_FileWriteBinData(FALSE, 0, OutputFrame, 0);
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- File Management
- "FileClose" (→ p. 663)
- "FileCopy" (→ p. 664)
- "FileDelete" (→ p. 666)
- "FileEOF" (→ p. 668)
- "FileExists" (→ p. 669)
- "FileOpenA" (→ p. 671)
- "FileOpenR" (→ p. 673)
- "FileOpenW" (→ p. 675)
- "FileReadBinData" (→ p. 677)
- "FileReadLine" (→ p. 678)
- "FileRename" (→ p. 680)
- "FileSeek" (→ p. 682)
- "FileSize" (→ p. 683)
- "FileWriteLine" (→ p. 686)

5.6.15 FileWriteLine

PLCopen

Function Block - Writes a string value to a text file.

5.6.15.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, write a string value to a file.
ID	UDINT	N/A	N/A	No default	The ID of the open file.
IN	STRING	N/A	N/A	No default	The string value to be written.

5.6.15.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).

5.6.15.3 Remarks

An end-of-line character is systematically written after the string value.

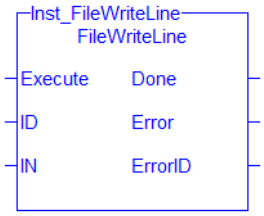


Figure 5-16: FileWriteLine

5.6.15.3.1 String Escape Sequences

For greater formatting control over the STRING output, escape the STRING by pre-pending a $ and use a pre-defined sequence. This is called a string escape sequence.

Escape Sequence	Result
$'	'
$$	$
$L	linefeed
$N	newline
$P	page (form feed)
$R	return
$T	tab
$xx	hex value

Example: The is how STRING escape sequences are used.

```
ID:=FileOpenW('c:\ myfile.txt');
WOK:=FileWriteLine(ID,'123456$N');
//WOK:=FileWriteLine(ID,'$N');
WOK:=FileWriteLine(ID,'abcd$N');
WOK:=FileWriteLine(ID,'the end');
WOK:=FileClose(ID);
```

The example outputs a file which reads:

```
123456
abcd
the end
```

5.6.15.4 FBD Language Example

Not available.

5.6.15.5 FFLD Language Example

Not available.

5.6.15.6 IL Language Example

Not available.

5.6.15.7 ST Language Example

```
(* FileWriteLine example *)
CASE StepCounter OF
0:
   Inst_FileWriteLine(TRUE, MyOutputFileID, 'Hello, world.');
   StepCounter := StepCounter + 1;
1:
   Inst_FileWriteLine(TRUE, MyOutputFileID, 'Hello, world.');
   IF Inst_FileWriteLine.Done THEN
      Inst_FileWriteLine(FALSE, 0, '');
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- File Management
- "FileClose" (→ p. 663)
- "FileCopy" (→ p. 664)
- "FileDelete" (→ p. 666)
- "FileEOF" (→ p. 668)
- "FileExists" (→ p. 669)
- "FileOpenA" (→ p. 671)
- "FileOpenR" (→ p. 673)
- "FileOpenW" (→ p. 675)
- "FileReadBinData" (→ p. 677)
- "FileReadLine" (→ p. 678)
- "FileRename" (→ p. 680)
- "FileSeek" (→ p. 682)
- "FileSize" (→ p. 683)
- "FileWriteBinData" (→ p. 685)

5.7 TCP/IP Function Blocks

This table lists the function blocks used to communicate via TCP/IP.

Name	Description
TcpAccept	Performs the accept operation.
TcpBinReceive	Receives characters over a socket connection.
TcpBinSend	Sends characters over a socket.
TcpClose	Closes and releases a socket.
TcpConnect	Creates a new socket and performs the connect operation.
TcpIsConnected	Tests if a client socket is connected.
TcpIsValid	Tests if a socket is valid.
TcpListen	Creates a new socket by performing the bind and listen operations.
TcpReceive	Receives characters over a socket connection.
TcpSend	Sends characters over a socket.

5.7.1 TcpAccept

PLCopen

Function Block - Performs the accept operation.

5.7.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, accept a new socket connection.
ListenID	UDINT	N/A	N/A	No default	The ID of a server socket returned by the **TcpListen** function block.

5.7.1.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).
ID	UDINT			Either: • ID of a new client socket accepted. • Invalid ID if no new connection.

5.7.1.3 Remarks

TIP

It is possible that the socket becomes invalid if an error occurs in the TCP connection after this function block is called.
Use the "TcpIsValid" (→ p. 699) function block after "TcpSend" (→ p. 702).
If the socket is no longer valid, close it using the "TcpClose" (→ p. 694) function block.

- This function block performs the **accept** operation using default TCP settings.
- Use the "TcpClose" (→ p. 694) function block to release the socket returned by **TcpAccept**.

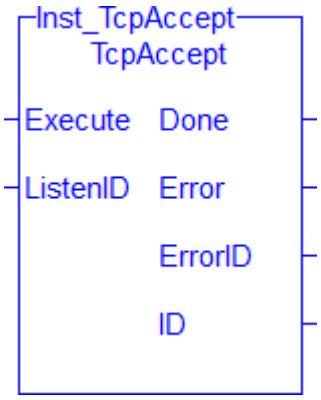


Figure 5-17: TcpAccept

5.7.1.4 FBD Language Example

Not available.

5.7.1.5 FFLD Language Example

Not available.

5.7.1.6 IL Language Example

Not available.

5.7.1.7 ST Language Example

```
(* TcpAccept example *)
CASE StepCounter OF
0:
   Inst_TcpAccept(TRUE, MyListenID);
   StepCounter := StepCounter + 1;
1:
   Inst_TcpAccept(TRUE, MyListenID);
   IF Inst_TcpAccept.Done THEN
      MySocketID := Inst_TcpAccept.ID;
      Inst_TcpAccept(FALSE, 0);
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- "TcpBinReceive" (→ p. 690)
- "TcpBinSend" (→ p. 692)
- "TcpClose" (→ p. 694)
- "TcpConnect" (→ p. 695)
- "TcpIsConnected" (→ p. 697)
- "TcpIsValid" (→ p. 699)
- "TcpListen" (→ p. 700)
- TcpReceive
- "TcpSend" (→ p. 702)

5.7.2 TcpBinReceive

PLCopen

Function Block - Receives characters over a socket connection.

5.7.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to perform copying a file.
ID	UDINT	N/A	N/A	No default	The ID of the client socket.
Data	USINT[ ]	N/A	N/A	No default	The array where the received data is stored.
MaxChar	DINT	N/A	N/A	No default	The maximum number of bytes to receive.

5.7.2.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).
NbRcv	DINT			The number of characters actually received.

5.7.2.3 Remarks

TIP

It is possible that the socket becomes invalid if an error occurs in the TCP connection after this function block is called.
Use the "TcpIsValid" (→ p. 699) function block after "TcpSend" (→ p. 702).
If the socket is no longer valid, close it using the "TcpClose" (→ p. 694) function block.

- It is possible that the number of characters actually received is less than the number expected.
  - In this case, call this function again on the next cycle to receive the pending characters.

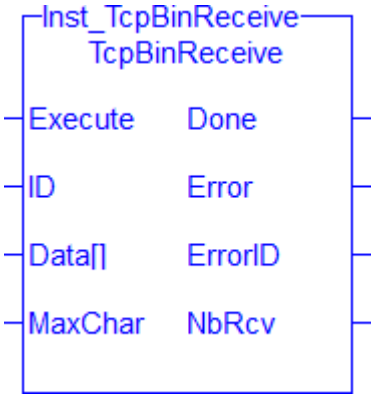


Figure 5-18: TcpBinReceive

5.7.2.4 FBD Language Example

Not available.

5.7.2.5 FFLD Language Example

Not available.

5.7.2.6 IL Language Example

Not available.

5.7.2.7 ST Language Example

```
(* TcpBinReceive example *)
CASE StepCounter OF
0:
   Inst_TcpBinReceive(TRUE, MySocketID, MyDataArray, 256);
   StepCounter := StepCounter + 1;
1:
   Inst_TcpBinReceive(TRUE, MySocketID, MyDataArray, 256);
   IF Inst_TcpBinReceive.Done THEN
      BytesReceived := Inst_TcpBinReceive.NbRcv;
      Inst_TcpBinReceive(FALSE, 0, MyDataArray, 0);
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- "TcpAccept" (→ p. 689)
- "TcpBinSend" (→ p. 692)
- "TcpClose" (→ p. 694)
- "TcpConnect" (→ p. 695)
- "TcpIsConnected" (→ p. 697)
- "TcpIsValid" (→ p. 699)
- "TcpListen" (→ p. 700)
- TcpReceive
- "TcpSend" (→ p. 702)

5.7.3 TcpBinSend

Function Block - Sends characters over a socket.

5.7.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, send a string over a socket connection.
ID	UDINT	N/A	N/A	No default	The ID of the client socket.
Data	USINT[ ]	N/A	N/A	No default	The data array to send.
DataSize	DINT	N/A	N/A	No default	Number of bytes in the Data array to send.

5.7.3.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).
NbSent	DINT			The number of characters actually sent.

5.7.3.3 Remarks

TIP

It is possible that the socket becomes invalid if an error occurs in the TCP connection after this function block is called.
Use the "TcpIsValid" (→ p. 699) function block after "TcpSend" (→ p. 702).
If the socket is no longer valid, close it using the "TcpClose" (→ p. 694) function block.

- It is possible the number of characters actually sent is less than the number expected.
  - In this case, use this function again to send the pending characters.

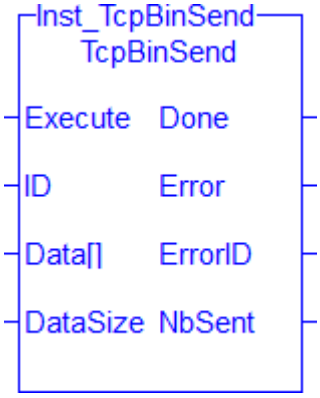


Figure 5-19: TcpBinSendf

5.7.3.4 FBD Language Example

Not available.

5.7.3.5 FFLD Language Example

Not available.

5.7.3.6 IL Language Example

Not available.

5.7.3.7 ST Language Example

```
(* TcpBinSend example *)
CASE StepCounter OF
0:
   Inst_TcpBinSend(TRUE, MySocketID, MyDataArray, 256);
   StepCounter := StepCounter + 1;
1:
   Inst_TcpBinSend(TRUE, MySocketID, MyDataArray, 256);
```

```
    IF Inst_TcpBinSend.Done THEN
       BytesSent := Inst_TcpBinSend.NbSent;
       Inst_TcpBinSend(FALSE, 0, MyDataArray, 0);
       StepCounter := StepCounter + 1;
    END_IF;
END_CASE;
```

See Also

- "TcpAccept" (→ p. 689)
- "TcpBinReceive" (→ p. 690)
- "TcpClose" (→ p. 694)
- "TcpConnect" (→ p. 695)
- "TcpIsConnected" (→ p. 697)
- "TcpIsValid" (→ p. 699)
- "TcpListen" (→ p. 700)
- TcpReceive
- "TcpSend" (→ p. 702)

5.7.4 TcpClose

Function Block - Closes and releases a socket.

5.7.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, close a socket.
ID	UDINT	N/A	N/A	No default	The ID of the client socket.

5.7.4.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).

5.7.4.3 Remarks

You are responsible for closing any socket created by the "TcpListen" (→ p. 700), "TcpAccept" (→ p. 689), or "TcpConnect" (→ p. 695) function blocks, even if they have become invalid.

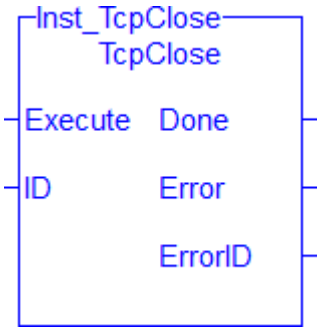


Figure 5-20: TcpClose

5.7.4.4 FBD Language Example

Not available.

5.7.4.5 FFLD Language Example

Not available.

5.7.4.6 IL Language Example

Not available.

5.7.4.7 ST Language Example

```
(* TcpClose example *)
CASE StepCounter OF
0:
   Inst_TcpClose(TRUE, MySocketID);
   StepCounter := StepCounter + 1;
1:
   Inst_TcpClose(TRUE, MySocketID);
   IF Inst_TcpClose.Done THEN
      Inst_TcpClose(FALSE, 0);
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- "TcpAccept" (→ p. 689)
- "TcpBinReceive" (→ p. 690)
- "TcpBinSend" (→ p. 692)
- "TcpConnect" (→ p. 695)
- "TcpIsConnected" (→ p. 697)
- "TcpIsValid" (→ p. 699)
- "TcpListen" (→ p. 700)
- TcpReceive
- "TcpSend" (→ p. 702)

5.7.5 TcpConnect

Function Block - Creates a new socket and performs the connect operation.

5.7.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to connect a socket to a server.
Address	STRING	N/A	N/A	No default	The IP Address of the remote server.
Port	DINT	N/A	User units	No default	The network port to use.

5.7.5.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).
ID	UDINT			The ID of the client socket.

5.7.5.3 Remarks

TIP

It is possible that the function returns a valid socket ID even if the connection to the server is not yet actually performed.
After calling this function, use the "TcpIsConnected" (→ p. 697) function block to know if the connection is ready.

- Use the "TcpClose" (→ p. 694) function block to release the socket returned by **TcpConnect**.
- This function block uses the default TCP settings and a specified server address and port.

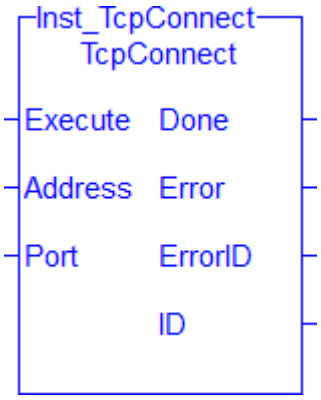


Figure 5-21: TcpConnect

5.7.5.4 FBD Language Example

Not available.

5.7.5.5 FFLD Language Example

Not available.

5.7.5.6 IL Language Example

Not available.

5.7.5.7 ST Language Example

```
(* TcpConnect example *)
CASE StepCounter OF
0:
   Inst_TcpConnect(TRUE, '192.168.1.1', 1234);
   StepCounter := StepCounter + 1;
1:
   Inst_TcpConnect(TRUE, '192.168.1.1', 1234);
   IF Inst_TcpConnect.Done THEN
      MySocketID := Inst_TcpConnect.ID;
      Inst_TcpConnect(FALSE, '', 0);
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- "TcpAccept" (→ p. 689)
- "TcpBinReceive" (→ p. 690)
- "TcpBinSend" (→ p. 692)
- "TcpClose" (→ p. 694)
- "TcpIsConnected" (→ p. 697)
- "TcpIsValid" (→ p. 699)
- "TcpListen" (→ p. 700)
- TcpReceive
- "TcpSend" (→ p. 702)

5.7.6 TcpIsConnected

PLCopen

Function Block - Tests if a client socket is connected.

5.7.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, test whether a socket is connected.
ID	UDINT	N/A	N/A	No default	The ID of the client socket.

5.7.6.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).
Connected	BOOL			TRUE if a connection is correctly established.

5.7.6.3 Remarks

TIP

It is possible that the socket becomes invalid if an error occurs in the TCP connection after this function block is called.
Use the "TcpIsValid" (→ p. 699) function block after "TcpSend" (→ p. 702).
If the socket is no longer valid, close it using the "TcpClose" (→ p. 694) function block.

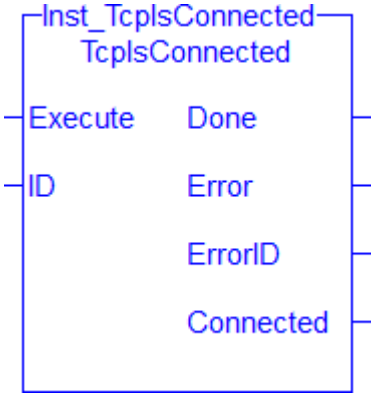


Figure 5-22: TcpIsConnected

5.7.6.4 FBD Language Example

Not available.

5.7.6.5 FFLD Language Example

Not available.

5.7.6.6 IL Language Example

Not available.

5.7.6.7 ST Language Example

```
(* TcpIsConnected example *)
CASE StepCounter OF
0:
   Inst_TcpIsConnected(TRUE, MySocketID);
   StepCounter := StepCounter + 1;
1:
   Inst_TcpIsConnected(TRUE, MySocketID);
   IF Inst_TcpIsConnected.Done THEN
      MyTcpIsConnected := Inst_TcpIsConnected.Connected;
      Inst_TcpIsConnected(FALSE, 0);
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- "TcpAccept" (→ p. 689)
- "TcpBinReceive" (→ p. 690)
- "TcpBinSend" (→ p. 692)
- "TcpClose" (→ p. 694)
- "TcpConnect" (→ p. 695)

- "TcpIsValid" (→ p. 699)
- "TcpListen" (→ p. 700)
- TcpReceive
- "TcpSend" (→ p. 702)

5.7.7 TcpIsValid

PLCopen

Function Block - Tests if a socket is valid.

5.7.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, request to perform copying a file.
ID	UDINT	N/A	N/A	No default	The ID of the client socket.

5.7.7.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).
Valid	BOOL			TRUE if the specified socket is still valid.

5.7.7.3 Remarks

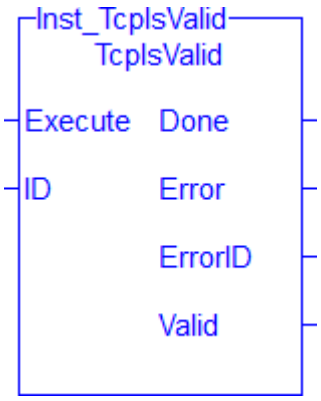


Figure 5-23: TcpIsValid

5.7.7.4 FBD Language Example

Not available.

5.7.7.5 FFLD Language Example

Not available.

5.7.7.6 IL Language Example

Not available.

5.7.7.7 ST Language Example

```
(* TcpIsValid example *)
CASE StepCounter OF
0:
   Inst_TcpIsValid(TRUE, MySocketID);
   StepCounter := StepCounter + 1;
1:
   Inst_TcpIsValid(TRUE, MySocketID);
   IF Inst_TcpIsValid.Done THEN
      MyTcpIsValid := Inst_TcpIsValid.Valid;
      Inst_TcpIsValid(FALSE, 0);
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- "TcpAccept" (→ p. 689)
- "TcpBinReceive" (→ p. 690)
- "TcpBinSend" (→ p. 692)
- "TcpClose" (→ p. 694)
- "TcpConnect" (→ p. 695)
- "TcpIsConnected" (→ p. 697)
- "TcpListen" (→ p. 700)
- TcpReceive
- "TcpSend" (→ p. 702)

5.7.8 TcpListen

Function Block - Creates a new socket by performing the bind and listen operations.

5.7.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, listen for socket connections.
Port	DINT	N/A	User units	No default	The network port to use.
Backlog	DINT	N/A	N/A	No default	The size of the queue for pending connections. If more than **Backlog** number of connection attempts are made prior to a "TcpAccept" (→ p. 689) call, the controller may refuse the connections.

5.7.8.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).
ListenID	UDINT			The ID of the new listen socket.

5.7.8.3 Remarks

- This function block uses the default TCP settings.
- Use the "TcpClose" (→ p. 694) function block to release the socket returned by **TcpListen**.

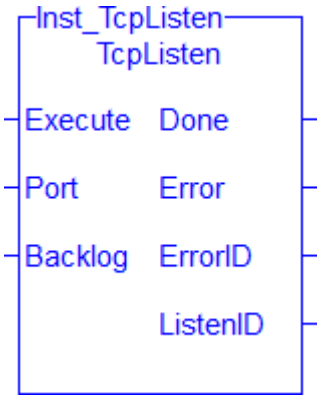


Figure 5-24: TcpListen

5.7.8.4 FBD Language Example

Not available.

5.7.8.5 FFLD Language Example

Not available.

5.7.8.6 IL Language Example

Not available.

5.7.8.7 ST Language Example

```
(* TcpListen example *)
CASE StepCounter OF
0:
   Inst_TcpListen(TRUE, 1234, 2);
   StepCounter := StepCounter + 1;
1:
   Inst_TcpListen(TRUE, 1234, 2);
   IF Inst_TcpListen.Done THEN
      MyListenID := Inst_TcpListen.ListenID;
      Inst_TcpListen(FALSE, 0, 0);
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- "TcpAccept" (→ p. 689)
- "TcpBinReceive" (→ p. 690)
- "TcpBinSend" (→ p. 692)
- "TcpClose" (→ p. 694)
- "TcpConnect" (→ p. 695)
- "TcpIsConnected" (→ p. 697)
- "TcpIsValid" (→ p. 699)
- TcpReceive
- "TcpSend" (→ p. 702)

5.7.9 TcpSend

Function Block - Sends characters over a socket.

5.7.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	0, 1	N/A	No default	On the rising edge, send a string over a socket connection.
ID	UDINT	N/A	N/A	No default	The ID of the client socket.
NbChar	DINT	N/A	N/A	No default	The number of characters to send.
Data	STRING	N/A	N/A	No default	The IP Address of the remote server.

5.7.9.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL			If TRUE, the command completed successfully.
Error	BOOL			If TRUE, an error has occurred.
ErrorID	DINT			Indicates the error if **Error** output is TRUE. See the table in "File and TCP/IP Function Block ErrorIDs" (→ p. 717).
NbSent	DINT			The number of characters actually sent.

5.7.9.3 Remarks

TIP

It is possible that the socket becomes invalid if an error occurs in the TCP connection after this function block is called.
Use the "TcpIsValid" (→ p. 699) function block after "TcpSend" (→ p. 702).
If the socket is no longer valid, close it using the "TcpClose" (→ p. 694) function block.

- It is possible the number of characters actually sent is less than the number expected.
  - In this case, use this function again to send the pending characters.

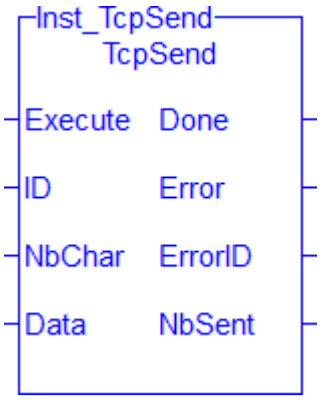


Figure 5-25: TcpSend

5.7.9.4 FBD Language Example

Not available.

5.7.9.5 FFLD Language Example

Not available.

5.7.9.6 IL Language Example

Not available.

5.7.9.7 ST Language Example

```
(* TcpSend example *)
CASE StepCounter OF
0:
   Inst_TcpSend(TRUE, MySocketID, 5, 'Hello');
   StepCounter := StepCounter + 1;
1:
   Inst_TcpSend(TRUE, MySocketID, 5, 'Hello');
   IF Inst_TcpSend.Done THEN
      BytesSent := Inst_TcpSend.NbSent;
      Inst_TcpSend(FALSE, 0, 0, '');
      StepCounter := StepCounter + 1;
   END_IF;
END_CASE;
```

See Also

- "TcpAccept" (→ p. 689)
- "TcpBinReceive" (→ p. 690)
- "TcpBinSend" (→ p. 692)
- "TcpClose" (→ p. 694)
- "TcpConnect" (→ p. 695)
- "TcpIsConnected" (→ p. 697)
- "TcpIsValid" (→ p. 699)
- "TcpListen" (→ p. 700)
- TcpReceive

5.8 UDP Functions for Controller and Simulator

UDP is a communications protocol which allows computers to exchange messages across an IP network.

When a UDP packet is sent to a broadcast address (e.g., 255.255.255.255), the controller automatically converts the given broadcast address to the broadcast address of its Ethernet interface.

Example

If the controller's IP address is `192.168.1.10` and the subnet mask is `255.255.255.0`, the controller's Ethernet interface broadcast address is `192.168.1.255`.

5.8.1 UDP Functions

This list of UDP functions allows a controller to communicate with a remote PC or another KAS controller over an Ethernet network.

Name	Description
udpAddrMake	Build an address buffer for UDP functions.
udpClose	Closes a socket.
udpCreate	Creates a UDP socket.
udpIsValid	Tests if a socket is valid.
udpRcvFrom	Receives a UDP telegram.
udpRcvFromArray	Receives a byte array through UDP.
udpRcvFromVar	Receives the contents of a variable from another controller through UDP.
udpSendTo	Sends a UDP telegram.
udpSendToArray	Sends a byte array through UDP.
udpSendToVar	Sends the contents of a local variable to another controller through UDP.

TIP

See this Wikipedia page about User Datagram Protocol for more information.

5.8.2 udpAddrMake

PLCopen

Function - Build an address buffer for UDP functions.

5.8.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Execute the function.
IPaddr	STRING	0.0.0.0, 255.255.255.255	N/A	No default	IP address in the form `XXX.XXX.XXX.XXX`.
port	DINT	0, 65535	N/A	No default	IP port number.
add[ ]	USINT	0, 32	N/A	No default	Buffer containing the UDP address (filled on output).

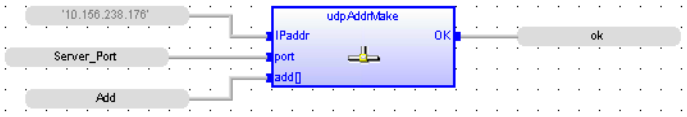
5.8.2.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL		N/A	Returns TRUE when the function successfully executes.

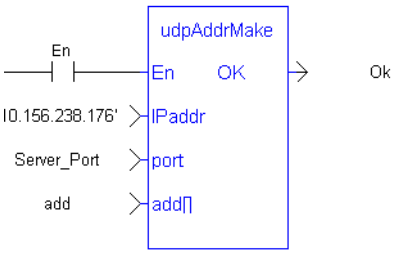
5.8.2.3 Remarks

- This function:
  - Builds an address buffer for UDP functions.
  - Is required for building an internal UDP address to be passed to the "udpSendTo" (→ p. 712) function in case of UDP client processing.

5.8.2.4 FBD Language Example



5.8.2.5 FFLD Language Example



5.8.2.6 IL Language Example

Not available.

5.8.2.7 ST Language Example

```
bAddrMake := udpAddrMake('10.156.238.176',Server_Port,add); //server details
```

5.8.3 udpClose

PLCopen

Function - Closes a socket.

5.8.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Execute the function.
sock	DINT	0, 65535	N/A	No default	IP of the socket.

5.8.3.2 Outputs

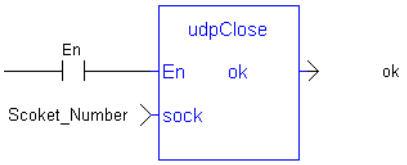
Output	Data Type	Range	Unit	Description
OK	BOOL		N/A	Returns TRUE when the function successfully executes.

5.8.3.3 Remarks

None

5.8.3.4 FBD Language Example

5.8.3.5 FFLD Language Example



5.8.3.6 IL Language Example

Not available.

5.8.3.7 ST Language Example

```
udpClose(Socket_Number); //Close socket
```

5.8.4 udpCreate

PLCopen

Function - Creates a UDP socket.

5.8.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Execute the function.
port	DINT	0, 65535	N/A	No default	UDP port number attached to the server socket or 0 for a client socket.

5.8.4.2 Outputs

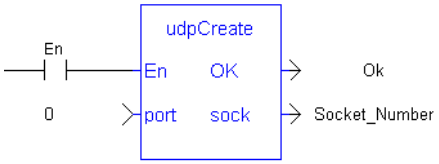
Output	Data Type	Range	Unit	Description
OK	BOOL		N/A	Returns TRUE when the function successfully executes.
sock	DINT		N/A	IP of the socket.

5.8.4.3 Remarks

None

5.8.4.4 FBD Language Example

5.8.4.5 FFLD Language Example



5.8.4.6 IL Language Example

Not available.

5.8.4.7 ST Language Example

```
Socket_Number := udpCreate(Client_Port); //create a socket
```

5.8.5 udpIsValid

PLCopen

Function - Tests if a socket is valid.

5.8.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Execute the function.
sock	DINT	0, 65535	N/A	No default	IP of the socket.

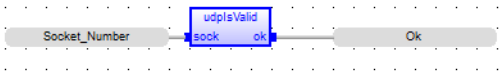
5.8.5.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL		N/A	Returns TRUE when the function successfully executes.

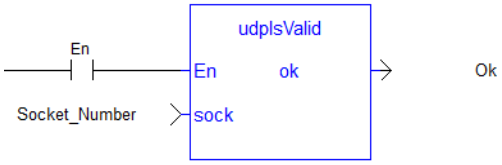
5.8.5.3 Remarks

None

5.8.5.4 FBD Language Example



5.8.5.5 FFLD Language Example



5.8.5.6 IL Language Example

Not available.

5.8.5.7 ST Language Example

```
bIsValid := udpIsValid(Socket_Number); //Valid socket?
```

5.8.6 udpRcvFrom

PLCopen

Function - Receives a UDP telegram.

5.8.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Execute the function.
sock	DINT	0, 65535	N/A	No default	IP of the socket.
nb	DINT	0, 65535	N/A	No default	Maximum number of characters received.
add[ ]	USINT	0, 32	N/A	No default	Buffer containing the UDP address (filled on output).
data	STRING	0, 255	N/A	No default	Buffer to store received characters.

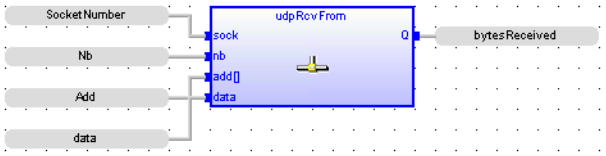
5.8.6.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL		N/A	Returns TRUE when the function successfully executes.
Q	DINT		N/A	Actual number of received characters.

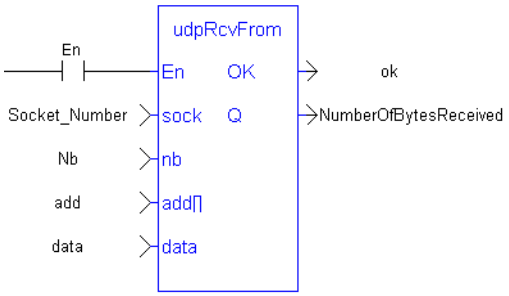
5.8.6.3 Remarks

- If the characters are received, the function fills the ADD argument with the internal UDP of the sender.
- This buffer can be passed to "udpSendTo" (→ p. 712) to send the answer.

5.8.6.4 FBD Language Example



5.8.6.5 FFLD Language Example



5.8.6.6 IL Language Example

Not available.

5.8.6.7 ST Language Example

```
ReceivedBytes := udpRcvFrom(Socket_Number,5,add,data); //Read the position
```

5.8.7 udpRcvFromArray

PLCopen

Function - Receives a byte array through UDP.

5.8.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Execute the function.
sock	DINT	0, 65535	N/A	No default	Socket number, return value from "udpCreate" (→ p. 706).
nb	DINT	0, 65535	N/A	No default	Number of bytes to be transferred.
add[ ]	USINT	0, 32	N/A	No default	Array which contains information about the server.
data[ ]	USINT	0, 65535	N/A	No default	Array of bytes to be transferred.

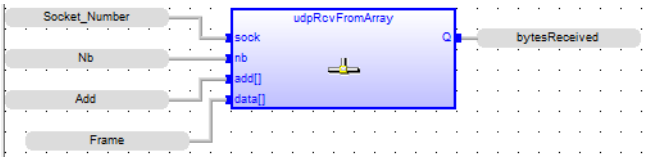
5.8.7.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL		N/A	Returns TRUE when the function successfully executes.
Q	DINT		N/A	Number of bytes received.

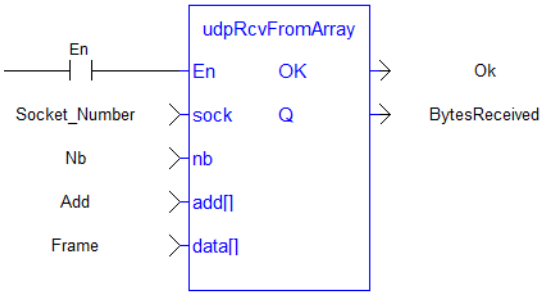
5.8.7.3 Remarks

None

5.8.7.4 FBD Language Example



5.8.7.5 FFLD Language Example



5.8.7.6 IL Language Example

Not available.

5.8.7.7 ST Language Example

```
BytesReceived := udpRcvFromArray(Socket_Number,nb,add,Frame);
```

5.8.8 udpRcvFromVar

PLCopen

Function - Receives the contents of a variable from another controller through UDP.

5.8.8.1 Inputs

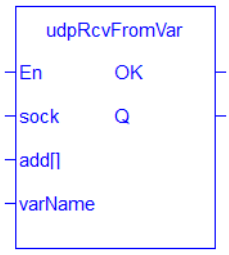
Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Execute the function.
sock	DINT	0, 65535	N/A	No default	Socket number, return value from "udpCreate" (→ p. 706).
add[ ]	USINT	0, 32	N/A	No default	Array which contains information about the server. This includes the sender's IP address.
varName	STRING		N/A	No default	The name of a PxMM variable (or array or structure) that stores data from the sender. The variable should be the same type as what is being sent. See "udpSendToVar" (→ p. 714) for more information.

5.8.8.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL		N/A	Returns TRUE when the function successfully executes.
Q	DINT		N/A	Number of bytes received.

5.8.8.3 Remarks

- Receives the contents a variable sent from another controller and saves it to a local variable.
- This allows for the exchange of data across controllers.



TIP

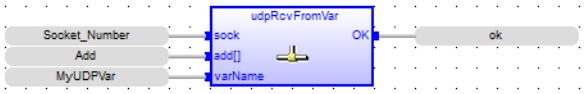
Limitations

- Function block instance variable types are not supported.
- These types of variables cannot be sent or received:
  - Variables defined with a UDFB.
  - The Input and Output variables defined for a sub-program.
- "udpSendToVar" (→ p. 714) and "udpRcvFromVar" (→ p. 710) do **not** automatically swap bytes for big vs. little endian systems.
- Send / receive functionality is:

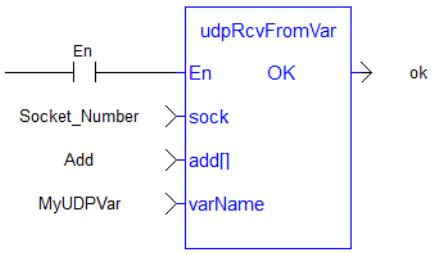
	PCMM / AKD PDMM	PCMM2G	Simulator
PCMM / AKD PDMM	✓	✗	✗
PCMM2G	✗	✓	✓
Simulator	✗	✓	✓

  Compatibility is based on the endianness of the controller's information.
  PCMM and AKD PDMM are big-endian.
  PCMM2G and Simulator are little-endian.
- 3rd party stand-alone programs on x86 platforms are responsible for endian conversions for UDP telegrams from a AKD PDMM / PCMM.

5.8.8.4 FBD Language Example



5.8.8.5 FFLD Language Example



5.8.8.6 IL Language Example

Not available.

5.8.8.7 ST Language Example

```
udpRcvFromVar( Socket_Number, Add, MyUDPVar )
```

5.8.9 udpSendTo

PLCopen

Function - Sends a UDP telegram.

5.8.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Execute the function.
sock	DINT	0, 65535	N/A	No default	ID of the client socket.
nb	DINT	0, 65535	N/A	No default	Number of bytes of data to send.
add[ ]	USINT	0, 32	N/A	No default	Buffer containing the UDP address (filled on output).
data	STRING	0, 255	N/A	No default	The characters to send.

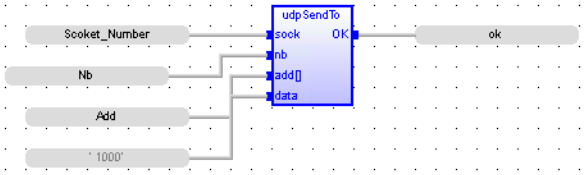
5.8.9.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL		N/A	Returns TRUE when the function successfully executes.

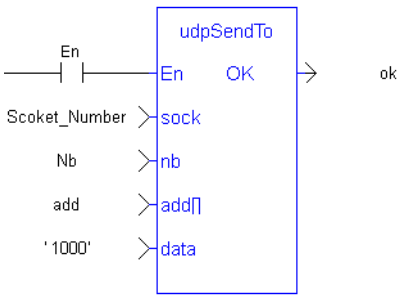
5.8.9.3 Remarks

None

5.8.9.4 FBD Language Example



5.8.9.5 FFLD Language Example



5.8.9.6 IL Language Example

Not available.

5.8.9.7 ST Language Example

```
bUdpSendTo := udpSendTo(Socket_Number,5,add,'1000');
```

5.8.10 udpSendToArray

PLCopen

Function - Sends a byte array through UDP.

5.8.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Execute the function.
sock	DINT	0, 65535	N/A	No default	Socket number, return value from "udpCreate" (→ p. 706).
nb	DINT	0, 65535	N/A	No default	Number of bytes to be transferred.
add[ ]	USINT	0, 32	N/A	No default	Array which contains information about the server.
data[ ]	USINT	0, 65535	N/A	No default	Array of bytes to be transferred.

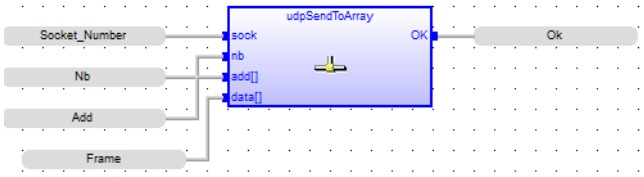
5.8.10.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL		N/A	Returns TRUE when the function successfully executes.

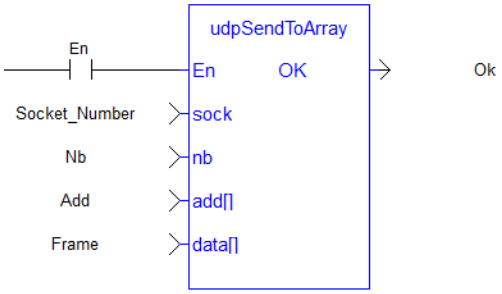
5.8.10.3 Remarks

None

5.8.10.4 FBD Language Example



5.8.10.5 FFLD Language Example



5.8.10.6 IL Language Example

Not available.

5.8.10.7 ST Language Example

```
Success:= udpSendToArray(Socket_Number,nb,add,Frame);
```

5.8.11 udpSendToVar

PLCopen

Function - Sends the contents of a local variable to another controller through UDP.

5.8.11.1 Inputs

Input	Data Type	Range	Unit	Default	Description
En	BOOL	0, 1	N/A	No default	Execute the function.
sock	DINT	0, 65535	N/A	No default	Socket number, return value from "udpCreate" (→ p. 706).
add[ ]	USINT	0, 32	N/A	No default	Array which contains information about the server. This includes the sender's IP address.
varName			N/A	No default	The name of a variable (or array or structure) to send to the receiver. The variable should be the same type as what is expected by the receiver. See "udpRcvFromVar" (→ p. 710) for more information.

5.8.11.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL		N/A	Returns TRUE when the function successfully executes.

5.8.11.3 Remarks

- Sends the contents of a local variable to another controller.
- This allows for the exchange of data across controllers.

TIP

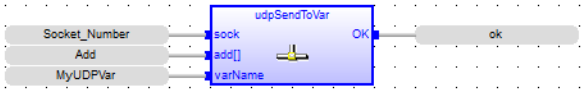
Limitations

- Function block instance variable types are not supported.
- These types of variables cannot be sent or received:
  - Variables defined with a UDFB.
  - The Input and Output variables defined for a sub-program.
- "udpSendToVar" (→ p. 714) and "udpRcvFromVar" (→ p. 710) do **not** automatically swap bytes for big vs. little endian systems.
- Send / receive functionality is:

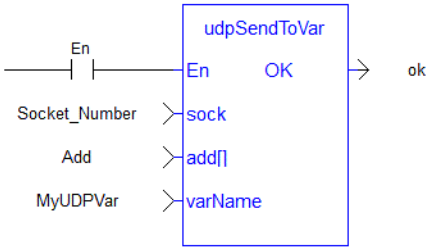
	PCMM / AKD PDMM	PCMM2G	Simulator
PCMM / AKD PDMM	✓	✗	✗
PCMM2G	✗	✓	✓
Simulator	✗	✓	✓

  Compatibility is based on the endianness of the controller's information.
  PCMM and AKD PDMM are big-endian.
  PCMM2G and Simulator are little-endian.
- 3rd party stand-alone programs on x86 platforms are responsible for endian conversions for UDP telegrams from a AKD PDMM / PCMM.

5.8.11.4 FBD Language Example



5.8.11.5 FFLD Language Example



5.8.11.6 IL Language Example

Not available.

5.8.11.7 ST Language Example

```
udpSendToVar( Socket_Number, Add, MyUDPVar )
```

5.9 PrintMessage

PLCopen ✓ Pipe Network ✓

Function - Generates a log message with any wanted strings in the Log Messages window.

5.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Level	DINT	0, 4	N/A	No default	Level of the logged message (i.e., it's importance). • By default, not all messages are shown in the log windows. • Only Error and Critical messages are shown. • Change the log settings to display a lower level. • PrintMessage logs SYSTEM messages. The Range defines are: • 0 = LEVEL_DEBUG • 1 = LEVEL_INFO • 2 = LEVEL_WARNING • 3 = LEVEL_ERROR • 4 = LEVEL_CRITICAL
Message	STRING	1 - 255			Content of the message. The string is a maximum of 255 characters.

5.9.2 Outputs

Output	Data Type	Range	Unit	Description
Default (.Q)	BOOL		N/A	Returns TRUE when the function successfully executes.

5.9.3 Remarks

5.9.3.1 Source

PrintMessage uses the SYSTEM message type.

To view all messages generated by PrintMessage, go to the log configuration and select the specified level for the SYSTEM source.

5.9.3.2 Level

The message could be sent with a logging level from 0 to 4 that qualifies its importance.

- The highest level, 4, logs critical messages.
  - Available levels are: Debug, Informational, Warning, Error, and Critical.
- Only Error and Critical messages a generated by default.
  - To force the system to generate every message level, use the Configuration Settings to change the settings to the desired level.

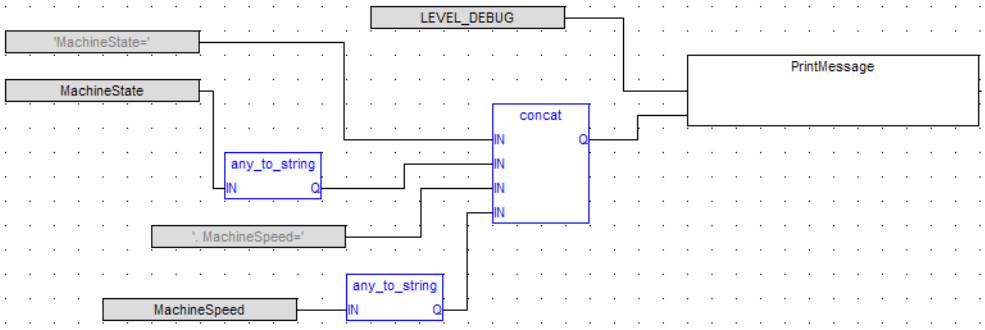
IMPORTANT

Enabling all messages can slow down the application's execution.
To avoid locking up communications between the IDE and Runtime, you must never include a print statement in your program that prints to the log every update cycle.

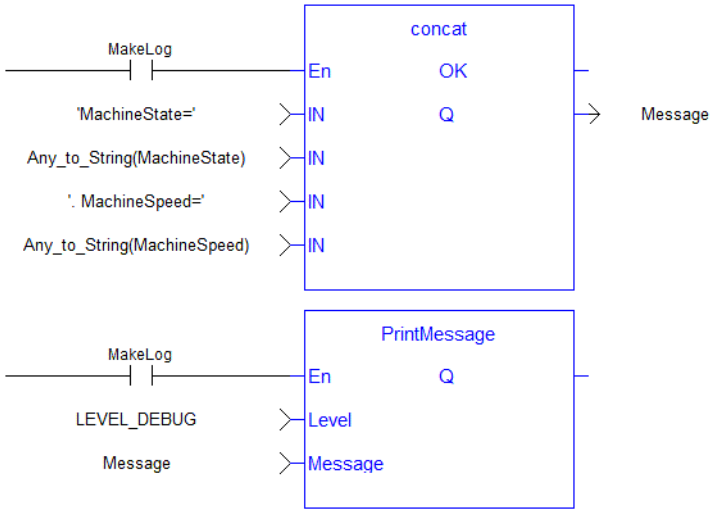
5.9.3.3 Usage

```
PrintMessage(  LEVEL_DEBUG, 'Message string to be logged' );
```

5.9.4 FBD Language Example



5.9.5 FFLD Language Example



5.9.6 IL Language Example

Not available.

5.9.7 ST Language Example

```
            // It's possible to create a temporary variable with the message.

MESSAGE := CONCAT( 'MachineState=', ANY_TO_STRING(MachineState), '.
MachineSpeed=', ANY_TO_STRING(MachineSpeed) );

// Then print the message to the log window
PrintMessage( LEVEL_INFO,     MESSAGE );
PrintMessage( LEVEL_WARNING,  MESSAGE );
PrintMessage( LEVEL_ERROR,    MESSAGE );

// Or to create the string directly in the function call:
PrintMessage( LEVEL_CRITICAL, CONCAT( 'MachineState=', ANY_TO_STRING
(MachineState), '. MachineSpeed=', ANY_TO_STRING(MachineSpeed) ) );
```

5.10 File and TCP/IP Function Block ErrorIDs

This is the list of possible errors that could be returned at the **ErrorID** output of the "File Tools Function Blocks" (→ p. 662) and "TCP/IP Function Blocks" (→ p. 688).

5.10.1 File ErrorIDs

ErrorID	Description
16000	Error opening file.
16001	All PLC file handles used.
16002	File ID is invalid.
16003	File ID has been closed.
16004	Error in file stream.
16005	End of file encountered.
16006	Internal file FB error.
16007	Unknown file error.
16008	No such file or directory.
16009	Bad file descriptor.
16010	File already exists.
16011	Too many open files in the system.
16012	Text file is busy.
16013	File is too large.
16014	File system is read only.
16015	File locking deadlock.
16016	Filename is too long.
16017	File positioning error.
16018	Bad or corrupted file system detected .

5.10.2 TCP/IP ErrorIDs

ErrorID	Description
16100	Connection error.
16101	TCP ID has been closed.
16102	All PLC TCP handles used.
16103	TCP ID is invalid.
16104	Failed to allocate TCP/IP socket.
16105	TCP operation internal error.

Appendix A: Index

A

Abort Camming **479**
Abort Gearing **493, 500**
actual position
pipe network 275
actual velocity **437**
Adapter, E/IP **69**
AKDFaultLookup
PLCopen 135-136
AKDFltRpt **78**
ASCII **25**
AXIS_GROUP_REF **560, 562, 564, 566-567, 569, 571-572, 574, 576, 578, 580, 582, 584, 591, 594, 597, 599, 602, 605-606, 614, 616-617, 624, 632, 634, 637, 642, 648**
AxisPlsPosModulo **81**
AxisPlsPosNoModulo **83**

C

Cam
Aborting 479
Coordinated Motion
Group Control items 559
Info items 591
List of Function Blocks 557
Motion items 609
Reference items 647
copyrights **726**
curve
synchronizer 388
cycle
missed 25

D

debug
EtherCAT 53
delay compensation **402**
disclaimer **727**
DriveFault
PipeNetwork 90
PLCopen 137
DriveParamRead **26**
DriveParamStrRead **31**
DriveParamWrite **34**

E

ECAT Restart
PLCopen 140
ECATGetObjVal **51**
ECATReadData **53**
ECATWCStatus **58**
ECATWriteData **59**
ECATWriteSdo **54, 61**

ECERR_DEVICE_ERROR **40**
ECERR_INVALID_ARRAY_SIZE **40**
EIP **69**
endian **711, 715**
EtherCAT
image 53, 60
timeout 29, 33, 37, 67
EtherCAT library **23**
EtherCAT library function
DriveParamRead 26
DriveParamStrRead 31
DriveParamWrite 34
ECATGetObjVal 51
ECATReadData 53
ECATWCStatus 58
ECATWriteData 59
ECATWriteSdo 54, 61

F

falling edge **399**
fast input **224, 233, 403**
fbCylinder **85**
feed-forward
torque- 305
feedback position **224, 236, 275**
FSoE **66**
FSoEParamsInit **65**

G

Gear
Aborting 493, 500
generator position **275**
GetCtrlPerf **658**

H

HomeFindHomeFastInput **94**
HomeFindHomeFastInputModulo **98**
HomeFindHomeInput **110**
HomeFindHomeInputThenZeroAngle **113**
HomeFindLimitFastInput **103, 106**
HomeFindLimitInput **115**
HomeFindLimitInputThenZeroAngle **117**
HomeFindZeroAngle **119**
HomeMoveUntilPosErrExceeded **121**
HomeMoveUntilPosErrExceededThenZeroAngle **123**
HomeUsingCurrenctPosition **126**
homing **512**

I

image EtherCAT **53, 60**

J

Jog
Pipe Network 127
PLCopen 170

K

Kollmorgen UDFB **179-180, 185**
AKDFaultLookup 135-136
AKDFltRpt 78
AxisPlsPosModulo 81
AxisPlsPosNoModulo 83
DriveFault 90, 137
ECATRestart 93, 140
fbCylinder 85
HomeFindHomeFastInput 94
HomeFindHomeFastInputModulo 98
HomeFindHomeInput 110
HomeFindHomeInputThenZeroAngle 113
HomeFindLimitFastInput 103, 106
HomeFindLimitInput 115
HomeFindLimitInputThenZeroAngle 117
HomeFindZeroAngle 119
HomeMoveUntilPosErrExceeded 121
HomeMoveUntilPosErrExceededThenZeroAngle 123
HomeUsingCurrenctPosition 126
Jog 127, 170
PlsPosFw 129
PlsPosFwBw 130
PlsTimeFw 132
S700FltRpt 87
ScaleInput 188
ScaleOutput 190
StepAbsolute 151
StepAbsSwitch 153
StepAbsSwitchFastInput 141
StepBlock 158
StepLimitSwitch 162
StepLimitSwitchFastInput 146
StepRefPulse 166
TemperaturePID 192

L

latency **408**

M

Mark-to-Machine **516**
Mark-to-Mark **516**
MC_AbortTrigger **410, 426**
MC_AddSuperAxis **447**
MC_AxisSetDefaults **610**
MC_CamIn **476**
MC_CamOut **484**
MC_CamResumePos **486**
MC_CamStartPos **488**
MC_CamTblSelect **490**
MC_ClearFaults **411**
MC_CreateAxesGrp **562**
MC_CreatePLCAxis **413**
MC_ErrorDescription **533**
MC_EStop **416**
MC_GearIn **492**
MC_GearInPos **495**
MC_GearOut **503**

MC_GrpDisable 564
MC_GrpEnable 565
MC_GrpHalt 614
MC_GrpReadActAcc 591
MC_GrpReadActPos 594
MC_GrpReadActVel 596
MC_GrpReadBoolPar 567
MC_GrpReadCmdPos 599
MC_GrpReadCmdVel 602
MC_GrpReadError 605
MC_GrpReadStatus 606
MC_GrpReset 570
MC_GrpSetOverride 616
MC_GrpSetPos 647
MC_GrpStop 572
MC_GrpWriteBoolPar 574
MC_Halt 449
MC_InitAxesGrp 577
MC_InitAxis 417
MC_InitAxisFeedback 419
MC_MachRegist 516
MC_MarkRegist 525
MC_MoveAdditive 456
MC_MoveCircAbs 617
MC_MoveCircRel 623
MC_MoveDirAbs 631
MC_MoveDirRel 634
MC_MoveLinAbs 636
MC_MoveLinRel 642
MC_MoveRelative 451, 464
MC_MoveSuperimp 467
MC_MoveVelocity 470
MC_Phasing 505
MC_Power 420
MC_ReadActPos 436
MC_ReadActVel 437
MC_ReadAxisErr 438
MC_ReadBoolPar 440
MC_ReadParam 441
MC_ReadStatus 442
MC_Reference 509
MC_RemSuperAxis 473
MC_ResetError 422
MC_SetKinTra 581
MC_SetOverride 475
MC_SetPos 514
MC_SetPosition 516
MC_Stop 423
MC_StopRegist 531
MC_SyncSlaves 507
MC_TouchProbe 428
MC_UngroupAllAxes 583
MC_WriteBoolPar 444
MC_WriteParam 445
missing PLC cycles 25
MLAxisAbs 226
MLAxisActualPos 255
MLAxisAdd 229
MLAxisClrErrors 266
MLAxisGenStatus 258
MLAxisPowerOff 252

MLAxisPowerOn **252**
MLAxisRefPos **234**
MLAxisRun **250**
MLAxisSetZero **275**
MLCNVConECAT **302**
MLPNAxisCreate **277**
MLPrfGetIRatio **284**
MLPrfGetORatio **286**
MLPrfSetIRatio **289**
MLPrfSetORatio **292**
MLProfileRelease **545**
MLSmpConPLCAxis **382**
MLSmpConPNAxis **383**
MLTrigGetPos **404**
MLTrigGetTime **406**
Modulo calculation, MLTrigReadPos **404**
motion library **199**
- adder 199
- Axis 223
- Block 212
- CAM Profile 279
- Comparator 293
- Convertor 301
- Delay 309
- Derivator 310
- Gear 314
- Integrator 328
- Master 331
- Phaser 24, 26, 352
- Pipe 217
- PMP 360
- Sampler 378
- State Machine 548
- Synchronizer 387
- Trigger 395

motion library function
- MC_AbortTrigger 410, 426
- MC_CamIn 476
- MC_CamOut 484
- MC_CamTblSelect 490
- MC_ClearFaults 411
- MC_CreatePLCAxis 413
- MC_EStop 416
- MC_GearIn 492
- MC_GearInPos 495
- MC_GearOut 503
- MC_Halt 449
- MC_InitAxis 417
- MC_InitAxisFeedback 419
- MC_MachRegist 516
- MC_MarkRegist 525
- MC_MoveAdditive 456
- MC_MoveRelative 451, 464
- MC_MoveSuperimp 467
- MC_MoveVelocity 470
- MC_Phasing 505
- MC_Power 420
- MC_ReadActPos 436
- MC_ReadActVel 437
- MC_ReadAxisErr 438
- MC_ReadBoolPar 440

MC_ReadParam 441
MC_ReadStatus 442
MC_Reference 509
MC_ResetError 422
MC_SetKinTra 581
MC_SetOverride 475
MC_SetPos 514
MC_SetPosition 516
MC_Stop 423
MC_StopRegist 531
MC_SyncSlaves 507
MC_TouchProbe 428
MC_WriteBoolPar 444
MC_WriteParam 445
MLAxisAbs 226
MLAxisAdd 229

P

phasing
synchronizer pipe block 387, 390
pipe position **276**
PlsPosFw **129**
PlsPosFwBw **130**
PlsTimeFw **132**
position
actual position 275
feedback position 224, 236, 275
generator position 275
pipe position 276
reference position 225, 263, 276, 293, 298
Position
Zero Offset 276

R

reference position **225, 263, 276, 293, 298**
registration **423, 516, 525**
rising edge **399**
robot **582, 585**

S

S-curve **371**
S700FltRpt **87**
ScaleInput **188**
ScaleOutput **190**
servo axis **409, 411, 417**
state machine
motion 548
stats **25-26**
StepAbsolute **151**
StepAbsSwitch **153**
StepAbsSwitchFastInput **141**
StepBlock **158**
StepLimitSwitch **162**
StepLimitSwitchFastInput **146**
StepRefPulse **166**
SuperimposedCmdPos **447, 473**

T

TCP/IP **688**
TCP/IP, functions **688**
TemperaturePID **192**
timeout
EtherCAT 29, 33, 37, 67
torque feed-forward **305**
trademarks **726**

U

UDFB
INOUT 78
out 78
UDP Functions;udpAddrMake **704**
UDP Functions;udpClose **705**
UDP Functions;udpCreate **706**
UDP Functions;udpIsValid **707**
UDP Functions;udpRcvFrom **708**
UDP Functions;udpRcvFromArray **709**
UDP Functions;udpSendTo **712**
UDP Functions;udpSendToArray **713**

V

Velocity, current **485, 504**

Z

Zero Offset
Pipe Network 276

6 Copyrights, Licenses, and Trademarks

"Copyrights" (→ p. 726) "Trademarks" (→ p. 726)
"PCMM2G" (→ p. 727) "Disclaimer" (→ p. 727)

6.1 Copyrights

Copyright © 2009-2023 Regal Rexnord Corporation, All Rights Reserved.

6.2 Trademarks

Regal Rexnord and Kollmorgen are trademarks of Regal Rexnord Corporation or one of its affiliated companies.

- KAS and AKD are registered trademarks of Kollmorgen.
- Kollmorgen is part of the Altra Industrial Motion Company.
- Codemeter is a registered trademark of WIBU-Systems AG.
- EnDat is a registered trademark of Dr. Johannes Heidenhain GmbH.
- EtherCAT is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH.
- Ethernet/IP Communication Stack: copyright (c) 2009, Rockwell Automation.
- Ethernet/IP is a registered trademark of ODVA, Inc.
- Ghostscript is a registered trademark of Artifex Software, Inc. and is distributed under the AGPL license.
- HIPERFACE is a registered trademark of Max Stegmann GmbH.
- PLCopen is an independent association providing efficiency in industrial automation.
- PROFINET is a registered trademark of PROFIBUS and PROFINET International (PI).
- SIMATIC is a registered trademark of SIEMENS AG.
- SyCon® is a registered trademark of Hilscher GmbH.
- Windows® is a registered trademark of Microsoft Corporation.

Kollmorgen Automation Suite is based on the work of:

- 7-zip (distributed under the terms of the LGPL and the BSD 3-clause licenses - see terms)
- curl software library
- JsonCpp software (distributed under the MIT License – see terms)
- Mongoose software (distributed under the MIT License - see terms)
- Qt cross-platform SDK (distributed under the terms of the LGPL3; Qt source is available on KDN)
- Qwt project (distributed under the terms of the Qwt License)
- The C++ Mathematical Expression Library (distributed under the MIT License)
- U-Boot, a universal boot loader is used by the AKD PDMM and PCMM (distributed under the terms of the GNU General Public License). The U-Boot source files, copyright notice, and readme are available on the distribution disk that is included with the AKD PDMM and PCMM.
- Zlib software library

All other product and brand names listed in this document may be trademarks or registered trademarks of their respective owners.

6.3 PCMM2G

The PCMM2G's Operating System (OS) and bootloader (U-Boot) are based on free and open source software, distributed under version 3 of the GNU General Public License as published by the Free Software Foundation. For more information about your rights and obligations with GPL-3.0, see the GNU website: https://www.gnu.org/licenses/gpl-3.0.html.

The individual copyright notices, license texts, and disclaimers of warranty for the software components are located in the controller, under the directory: /usr/share/common-licenses/.

For more information about accessing the PCMM2G's files, see SSH Login to a Controller.

The OS, bootloader, and their software component's source codes including modifications, copyright notices, license texts, disclaimers of warranty, and the compilation scripts to build the OS image are available from the Kollmorgen web-site: https://www.kollmorgen.com/en-us/developer-network/downloads.

The OS image and its corresponding sources file is identified by an "OS-Sources" designator, followed by its version number: OS-Sources-x.xx.x.xxxxx.

The compilation scripts and sources file used to build the OS image is identified by the "OS-Build-Sources" designator, followed by its version number: OS-Build-Sources-x.xx.x.xxxxx.

See PCMM2G - File Naming Conventions in the KAS online help.

6.4 Disclaimer

The information in this document is believed to be accurate and reliable at the time of its release. Notwithstanding the foregoing, Kollmorgen assumes no responsibility for any damage or loss resulting from the use of this help, and expressly disclaims any liability or damages for loss of data, loss of use, and property damage of any kind, direct, incidental or consequential, in regard to or arising out of the performance or form of the materials presented herein or in any software programs that accompany this document.

All timing diagrams, whether produced by Kollmorgen or included by courtesy of the PLCopen organization, are provided with accuracy on a best-effort basis with no warranty, explicit or implied, by Kollmorgen. The user releases Kollmorgen from any liability arising out of the use of these timing diagrams.

7 Support and Services

About KOLLMORGEN

Kollmorgen is a leading provider of motion systems and components for machine builders.
Through world-class knowledge in motion, industry-leading quality and deep expertise in linking and integrating standard and custom products,
Kollmorgen delivers breakthrough solutions that are unmatched in performance, reliability and ease-of-use, giving machine builders an irrefutable marketplace advantage.

Join the Kollmorgen Support (Developer) Network for product support.
Ask the community questions, search the knowledge base for answers, get downloads, and suggest improvements.

North America

KOLLMORGEN
201 West Rock Road
Radford, VA 24141, USA

Web:	www.kollmorgen.com
Mail:	support@kollmorgen.com
Tel.:	+1 - 540 - 633 - 3545
Fax:	+1 - 540 - 639 - 4162

Europe

KOLLMORGEN Europe GmbH
Pempelfurtstr. 1
40880 Ratingen, Germany

Web:	www.kollmorgen.com
Mail:	technik@kollmorgen.com
Tel.:	+49 - 2102 - 9394 - 0
Fax:	+49 - 2102 - 9394 - 3155

South America

Altra Industrial Motion do Brasil
Equipamentos Industriais LTDA.
Avenida João Paulo Ablas, 2970
Jardim da Glória, Cotia - SP
CEP 06711-250, Brazil

Web:	www.kollmorgen.com
Mail:	contato@kollmorgen.com
Tel.:	+55 11 4615-6300

China and SEA

KOLLMORGEN
Room 302, Building 5, Lihpao Plaza,
88 Shenbin Road, Minhang District,
Shanghai, China.

Web:	www.kollmorgen.cn
Mail:	sales.china@kollmorgen.com
Tel.:	+86 - 400 668 2802
Fax:	+86 - 21 6248 5367