

Kollmorgen Automation Suite

KAS Reference Manual - Motion Library



Document Edition: D

October 2013

Valid for Software Revision 2.7

Valid for AKD PDMM firmware version: 1.10

Part Number: 959716

Keep all manuals as a product component during the life span of the product.
Pass all manuals to future users / owners of the product.

Trademarks and Copyrights

Copyrights

Copyright © 2009-13 Kollmorgen™

Information in this document is subject to change without notice. The software package described in this document is furnished under a license agreement or non-disclosure agreement. The software may be used or copied only in accordance with the terms of those agreements.

This document is the intellectual property of Kollmorgen™ and contains proprietary and confidential information. The reproduction, modification, translation or disclosure to third parties of this document (in whole or in part) is strictly prohibited without the prior written permission of Kollmorgen™.

Trademarks

KAS and AKD are registered trademarks of Kollmorgen™.

SERVOSTAR is a registered trademark of Kollmorgen™.

Kollmorgen™ is part of the Danaher Motion company.

Windows® is a registered trademark of Microsoft Corporation

EnDat is a registered trademark of Dr. Johannes Heidenhain GmbH.

EtherCAT® is registered trademark of Ethercat Technology Group.

PLCopen® is an independent association providing efficiency in industrial automation.

INtime® is a registered trademark of TenAsys® Corporation.

Codemeter is a registered trademark of WIBU-Systems AG.

All product and company names are trademarks™ or registered® trademarks of their respective holders. Use of them does not imply any affiliation with or endorsement by them.

Kollmorgen Automation Suite is based on the work of:

- AdvXMLParser, an XML file parser modified to handle the ALS project (distributed under the terms of the license).
- AjaxFileUpload, software (distributed under the MPL License).
- Apache log4net library for output logging (distributed under the Apache License).
- bsdtar and libarchive2, a utility and library to create and read several different archive formats (distributed under the terms of the BSD License).
- bzip2.dll, a data compression library (distributed under the terms of the BSD License).
- Curl software library
- DockPanel Suite, a docking library for .Net Windows Forms (distributed under the MIT License).
- FileHelpers library to import/export data from fixed length or delimited files.
- GNU gzip¹ (www.gnu.org) is used by the PDMM (distributed under the terms of the GNU General Public License <http://www.gnu.org/licenses/gpl-2.0.html>).
- GNU Tar² (www.gnu.org) is used by the PDMM (distributed under the terms of the GNU General Public License <http://www.gnu.org/licenses/gpl-2.0.html>).
- jQuery.Cookies, a Javascript library for accessing and manipulating HTTP cookies in the web browser (distributed under the MIT License).

¹Copyright (C) 2007 Free Software Foundation, Inc. Copyright (C) 1993 Jean-loup Gailly. This is free software. You may redistribute copies of it under the terms of the GNU General Public License <<http://www.gnu.org/licenses/gpl.html>>. There is NO WARRANTY, to the extent permitted by law. Written by Jean-loup Gailly.

²Copyright (C) 2007 Free Software Foundation, Inc. License GPLv2+: GNU GPL version 2 or later <<http://gnu.org/licenses/gpl.html>> This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law. Written by John Gilmore and Jay Fenlason.

- jQuery File Tree, a file browser plugin (distributed under the MIT License).
- jQueryRotate, a plugin which rotates images (img html objects) by a given angle on web pages (distributed under the MIT License, <http://opensource.org/licenses/mit-license.php>).
- JsonCpp software (distributed under the MIT License –see terms see <http://json-cpp.sourceforge.net/LICENSE> for terms).
- LZMA SDK (<http://www.7-zip.org/sdk.html>), used to compress crash dump information (available as public domain).
- Mongoose v3.7, an embedded web server library (distributed under the MIT License).
- MVVM Light Toolkit components for Model – View –ViewModel patterns with Windows Presentation Foundation (distributed under the MIT License).
- Qwt project (distributed under the terms of the GNU Lesser General Public License).
- U-Boot, a universal boot loader is used by the AKD-PDMM (distributed under the terms of the GNU General Public License, <http://www.gnu.org/licenses/gpl-2.0.html>). The U-Boot source files, copyright notice, and readme are available on the distribution disk that is included with the AKD-PDMM.
- ZedGraph class library, user control, and web control for .NET (distributed under the LGPL License).
- Zlib software library
- Zlib1.dll, a data compression library (distributed under the terms of the BSD License).

All other product and brand names listed in this document may be trademarks or registered trademarks of their respective owners.

Disclaimer

The information in this document (Version 2.7 published on 10/7/2013) 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.

This page intentionally left blank.

Table of Contents

Kollmorgen Automation Suite	1
KAS Reference Manual - Motion Library	1
Trademarks and Copyrights	2
Copyrights	2
Trademarks	2
Disclaimer	3
Table of Contents	5
1 Motion Library	37
1.1 Motion Library / Pipe Network	38
1.1.1 Motion Library	38
1.1.2 Motion Library - Pipe Network	38
1.1.2.1 MLPipeAct	39
Description	39
Arguments	39
Input	39
Output	39
Return Type	40
Related Functions	40
Example	40
Structured Text	40
Ladder Diagram	40
Function Block Diagram	40
1.1.2.2 MLPipeAddBlock	40
Description	40
Arguments	41
Input	41
Output	41
Return Type	41
Related Functions	41
Example	42
Structured Text	42
Ladder Diagram	42
Function Block Diagram	42
1.1.2.3 MLPipeCreate	42

Description	42
Arguments	42
Input	42
Output	43
Related Functions	43
Example	43
Structured Text	43
Ladder Diagram	43
Function Block Diagram	43
1.1.2.4 MLPipeDeact	43
Description	43
Arguments	44
Input	44
Output	44
Return Type	44
Related Functions	44
Example	44
Structured Text	44
Ladder Diagram	45
Function Block Diagram	45
1.1.3 Motion Library - Block	45
1.1.3.1 MLBlkCreate	45
Description	45
Arguments	45
Input	45
Output	46
Related Functions	46
Example	46
Structured Text	46
Ladder Diagram	46
Function Block Diagram	46
1.1.3.2 MLBlkReadOutVal	47
Description	47
Arguments	47
Input	47
Output	47

Related Functions	47
Example	47
Structured Text	47
Ladder Diagram	47
Function Block Diagram	47
1.1.3.3 MLBikReadModPos	48
Description	48
Arguments	48
Input	48
Output	48
Related Functions	48
Example	48
Structured Text	48
Ladder Diagram	49
Function Block Diagram	49
1.1.3.4 MLBikIsReady	49
Description	49
Arguments	49
Input	49
Output	49
Return Type	49
Related Functions	49
Example	50
Structured Text	50
Ladder Diagram	50
Function Block Diagram	50
1.1.3.5 MLBikWriteModPos	50
Description	50
Arguments	50
Input	50
Output	51
Return Type	51
Related Functions	51
Example	51
Structured Text	51
Ladder Diagram	51

Function Block Diagram	51
1.1.4 Motion Library - Adder	52
1.1.4.1 MAddInit	52
Description	52
Arguments	53
Input	53
Output	53
Return Type	53
Related Functions	53
Example	54
Structured Text	54
Ladder Diagram	54
Function Block Diagram	54
1.1.4.2 MAddReadOff1	54
Description	54
Arguments	55
Input	55
Output	55
Related Functions	55
Example	55
Structured Text	55
Ladder Diagram	55
Function Block Diagram	56
1.1.4.3 MAddReadOff2	56
Description	56
Arguments	56
Input	56
Output	56
Related Functions	56
Example	57
Structured Text	57
Ladder Diagram	57
Function Block Diagram	57
1.1.4.4 MAddReadRatio1	57
Description	57
Arguments	57

Input	57
Output	58
Related Functions	58
Example	58
Structured Text	58
Ladder Diagram	58
Function Block Diagram	58
1.1.4.5 MLAddReadRatio2	58
Description	58
Arguments	59
Input	59
Output	59
Related Functions	59
Example	59
Structured Text	59
Ladder Diagram	59
Function Block Diagram	60
1.1.4.6 MLAddWriteInput	60
Description	60
Arguments	60
Input	60
Output	61
Return Type	61
Related Functions	61
Example	61
Structured Text	61
Ladder Diagram	61
Function Block Diagram	62
1.1.4.7 MLAddWriteOff1	62
Description	62
Arguments	62
Input	62
Output	62
Return Type	63
Related Functions	63
Example	63

Structured Text	63
Ladder Diagram	63
Function Block Diagram	63
1.1.4.8 MAddWriteOff2	63
Description	63
Arguments	64
Input	64
Output	64
Return Type	64
Related Functions	64
Example	65
Structured Text	65
Ladder Diagram	65
Function Block Diagram	65
1.1.4.9 MAddWriteRat1	65
Description	65
Arguments	66
Input	66
Output	66
Return Type	66
Related Functions	66
Example	66
Structured Text	66
Ladder Diagram	67
Function Block Diagram	67
1.1.4.10 MAddWriteRat2	67
Description	67
Arguments	67
Input	67
Output	68
Return Type	68
Related Functions	68
Example	68
Structured Text	68
Ladder Diagram	68
Function Block Diagram	68

1.1.5 Motion Library - Axis	69
1.1.6 Motion Library - Cam Profile	71
1.1.7 Motion Library - Comparator	71
1.1.7.1 MLCompCheck	71
Description	71
Arguments	72
Input	72
Output	72
Return Type	72
Related Functions	72
Example	73
Structured Text	73
Ladder Diagram	73
Function Block Diagram	73
1.1.7.2 MLComplnit	73
Description	73
Arguments	73
Input	73
Output	74
Return Type	74
Related Functions	74
Example	74
Structured Text	74
Ladder Diagram	75
Function Block Diagram	75
1.1.7.3 MLCompReadRef	75
Description	75
Arguments	75
Input	75
Output	75
Related Functions	75
Example	76
Structured Text	76
Ladder Diagram	76
Function Block Diagram	76
1.1.7.4 MLCompReset	76

Description	76
Arguments	76
Input	76
Output	76
Return Type	76
Related Functions	76
Example	77
Structured Text	77
Ladder Diagram	77
Function Block Diagram	77
1.1.7.5 MLCompWriteRef	77
Description	77
Arguments	78
Input	78
Output	78
Return Type	79
Related Functions	79
Example	79
Structured Text	79
Ladder Diagram	79
Function Block Diagram	79
1.1.7.6 Usage example of Comparator Functions	79
1.1.8 Motion Library - Convertor	82
1.1.8.1 MLCNVConnect	83
Description	83
Arguments	83
Input	83
Output	84
Return Type	84
Related Functions	84
Example	84
Structured Text	84
Ladder Diagram	84
Function Block Diagram	84
1.1.8.2 MLCNVConnectEx	84
Description	84

Arguments	85
Input	85
Output	86
Return Type	86
Related Functions	87
Example	87
Structured Text	87
Ladder Diagram	87
Function Block Diagram	87
1.1.8.3 MLCNVDisconnect	87
Description	87
Arguments	88
Input	88
Output	88
Return Type	88
Related Functions	88
Example	88
Structured Text	88
Ladder Diagram	88
Function Block Diagram	88
1.1.8.4 MLCNVInit	89
Description	89
Arguments	89
Input	89
Output	89
Return Type	89
Related Functions	89
Example	89
Structured Text	89
Ladder Diagram	90
Function Block Diagram	90
1.1.9 Motion Library - Delay	90
1.1.9.1 MLDelayInit	90
Description	90
Arguments	90
Input	90

Example	90
Structured Text	90
Ladder Diagram	91
Function Block Diagram	91
1.1.10 Motion Library - Derivator	91
1.1.10.1 MLDerInit	91
Description	91
Arguments	92
Input	92
Output	92
Return Type	92
Related Functions	92
Example	92
Structured Text	92
Ladder Diagram	93
Function Block Diagram	93
1.1.10.2 MLDerReadInModPos	93
Description	93
Arguments	94
Input	94
Output	94
Related Functions	94
Example	94
Structured Text	94
Ladder Diagram	94
Function Block Diagram	94
1.1.10.3 MLDerWriteInModPos	95
Description	95
Arguments	95
Input	95
Output	95
Return Type	96
Related Functions	96
Example	96
Structured Text	96
Ladder Diagram	96

Function Block Diagram	96
1.1.11 Motion Library - Gear	96
1.1.11.1 Usage example of Gear Functions	97
1.1.12 Motion Library - Integrator	98
1.1.12.1 MLIntInit	99
Description	99
Arguments	99
Input	99
Output	100
Return Type	100
Related Functions	100
Example	100
Structured Text	100
Ladder Diagram	100
Function Block Diagram	100
1.1.12.2 MLIntWriteOutVal	100
Description	100
Arguments	101
Input	101
Output	101
Return Type	101
Related Functions	101
Example	101
Structured Text	101
Ladder Diagram	101
Function Block Diagram	102
1.1.13 Motion Library - Master	102
1.1.13.1 MLMstAbs	103
Description	103
Arguments	103
Input	103
Output	103
Related Functions	104
Example	104
Structured Text	104
Ladder Diagram	104

Function Block Diagram	104
1.1.13.2 MLMstAdd	104
Description	104
Arguments	104
Input	104
Output	105
Related Functions	105
Example	105
Structured Text	105
Ladder Diagram	105
Function Block Diagram	105
1.1.13.3 MLMstForcePos	106
Description	106
Arguments	106
Input	106
Output	106
Related Functions	106
Example	106
Structured Text	106
Ladder Diagram	107
Function Block Diagram	107
1.1.13.4 MLMstInit	107
Description	107
Arguments	108
Input	108
Output	109
Example	110
Structured Text	110
Ladder Diagram	110
Function Block Diagram	110
1.1.13.5 MLMstReadAccel	110
Description	110
Arguments	111
Input	111
Output	111
Related Functions	111

Example	111
Structured Text	111
Ladder Diagram	111
Function Block Diagram	111
1.1.13.6 MLMstReadDecel	112
Description	112
Arguments	112
Input	112
Output	112
Example	112
Structured Text	112
Ladder Diagram	112
Function Block Diagram	112
1.1.13.7 MLMstReadInitPos	113
Description	113
Arguments	113
Input	113
Output	113
Example	113
Structured Text	113
Ladder Diagram	113
Function Block Diagram	113
1.1.13.8 MLMstReadSpeed	114
Description	114
Arguments	114
Input	114
Output	114
Related Functions	114
Example	114
Structured Text	114
Ladder Diagram	114
Function Block Diagram	115
1.1.13.9 MLMstRel	115
Description	115
Arguments	115
Input	115

Output	115
Related Functions	115
Example	115
Structured Text	115
Ladder Diagram	116
Function Block Diagram	116
1.1.13.10 MLMstRun	116
Description	116
Arguments	116
Input	116
Output	116
Related Functions	117
Example	117
Structured Text	117
Ladder Diagram	117
Function Block Diagram	117
1.1.13.11 MLMstStatus	117
Description	117
Arguments	118
Input	118
Output	118
Example	118
Structured Text	118
Ladder Diagram	119
Function Block Diagram	119
1.1.13.12 MLMstWriteAccel	119
Description	119
Arguments	119
Input	119
Output	119
Related Functions	119
Example	120
Structured Text	120
Ladder Diagram	120
Function Block Diagram	120
1.1.13.13 MLMstWriteDecel	120

Description	120
Arguments	120
Input	120
Output	121
Related Functions	121
Example	121
Structured Text	121
Ladder Diagram	121
Function Block Diagram	121
1.1.13.14 MLMstWriteInitPos	121
Description	121
Arguments	122
Input	122
Output	122
Example	122
Structured Text	122
Ladder Diagram	122
Function Block Diagram	123
1.1.13.15 MLMstWriteSpeed	123
Description	123
Arguments	123
Input	123
Output	123
Related Functions	123
Example	123
Structured Text	123
Ladder Diagram	124
Function Block Diagram	124
1.1.13.16 Usage example of Master Functions	124
1.1.14 Motion Library - Phaser	125
1.1.14.1 Usage example of Phaser Functions	125
1.1.15 Motion Library - PMP	127
1.1.16 Motion Library - Sampler	128
1.1.16.1 MLSmpConnect	128
Description	128
Arguments	128

Input	128
Output	129
Return Type	129
Example	129
Structured Text	129
Ladder Diagram	129
Function Block Diagram	129
1.1.16.2 MLSmpConnectEx	129
Description	129
Arguments	130
Input	130
Output	131
Return Type	131
Example	132
Structured Text	132
Ladder Diagram	132
Function Block Diagram	132
1.1.16.3 MLSmpInit	132
Description	132
Arguments	132
Input	132
Output	133
Example	133
Structured Text	133
Ladder Diagram	134
Function Block Diagram	134
1.1.17 Motion Library - Synchronizer	134
1.1.17.1 MLSynclnit	134
Description	134
Arguments	135
Input	135
Output	135
Related Functions	135
Example	135
Structured Text	135
Ladder Diagram	136

Function Block Diagram	136
1.1.17.2 MLSyncReadDeltaS	136
Description	136
Arguments	137
Input	137
Output	137
Related Functions	137
Example	137
Structured Text	137
Ladder Diagram	137
Function Block Diagram	138
1.1.17.3 MLSyncStart	138
Description	138
Arguments	138
Input	138
Output	138
Example	138
Structured Text	138
Ladder Diagram	139
Function Block Diagram	139
1.1.17.4 MLSyncStop	139
Description	139
Arguments	139
Input	139
Output	139
Example	140
Structured Text	140
Ladder Diagram	140
Function Block Diagram	140
1.1.17.5 MLSyncWriteDeltaS	140
Description	140
Arguments	141
Input	141
Output	141
Example	141
Structured Text	141

Ladder Diagram	142
Function Block Diagram	142
1.1.17.6 Usage example of Synchronizer Functions	142
1.1.18 Motion Library - Trigger	143
1.1.18.1 MLTrigClearFlag	143
Description	143
Arguments	144
Input	144
Output	144
Return Type	144
Related Functions	144
Example	144
Structured Text	144
Ladder Diagram	144
Function Block Diagram	144
1.1.18.2 MLTrigInit	145
Description	145
Arguments	145
Input	145
Output	146
Return Type	146
Related Functions	146
Example	146
Structured Text	146
Ladder Diagram	146
Function Block Diagram	146
1.1.18.3 MLTrigsTriggered	147
Description	147
Arguments	147
Input	147
Output	147
Return Type	147
Related Functions	148
Example	148
Ladder Diagram	148
Function Block Diagram	148

1.1.18.4 MLTrigReadDelay	148
Description	148
Input	148
Output	149
Related Functions	149
1.1.18.5 MLTrigReadPos	149
Description	149
Arguments	150
Input	150
Output	150
Related Functions	150
Previous Function Name	150
Example	150
Structured Text	150
Ladder Diagram	150
Function Block Diagram	150
1.1.18.6 MLTrigReadTime	151
Description	151
Arguments	151
Input	151
Output	151
Related Functions	151
Previous Function Name	152
Example	152
Ladder Diagram	152
Function Block Diagram	152
1.1.18.7 MLTrigSetEdge	152
Description	152
Arguments	152
Input	152
Output	153
Return Type	153
Examples	153
Function Block Diagram	153
Ladder Diagram	153
Structured Text	153

1.1.18.8	MLTrigWriteDelay	153
	Description	153
	Input	153
	Output	154
	Return Type	154
	Related Functions	154
1.1.18.9	Usage example of Trigger Functions	154
1.2	Motion Library / PLCopen	155
	1.2.1 Control	156
	1.2.1.1 MC_ClearFaults (Function)	156
	Description	156
	Arguments	157
	Input	157
	Output	157
	Usage	157
	Related Functions	157
	Example	157
	Structured Text	157
	Ladder Diagram	158
	1.2.1.2 MC_CreateAxis	158
	Description	158
	Arguments	158
	Input	158
	Output	160
	Example	160
	Structured Text	160
	Ladder Diagram	160
	1.2.1.3 MC_EStop	160
	Description	161
	Arguments	161
	Input	161
	Output	161
	Usage	161
	Related Functions	161
	Example	161
	Structured Text	161

Ladder Diagram	162
1.2.1.4 MC_InitAxis (Function)	162
Description	162
Arguments	162
Input	162
Output	163
Example	164
Structured Text	164
Ladder Diagram	164
1.2.1.5 MC_Power	164
Description	164
Arguments	165
Input	165
Output	165
Example	166
Structured Text	166
Ladder Diagram	166
1.2.1.6 MC_ResetError	166
Description	166
Arguments	167
Input	167
Output	167
Example	167
Structured Text	167
Ladder Diagram	168
1.2.1.7 MC_Stop (Function Block)	168
Description	168
Time Diagram	168
Arguments	169
Input	169
Output	170
Example	170
Structured Text	170
Ladder Diagram	171
1.2.2 I/O	171
1.2.2.1 MC_AbortTrigger (Function Block)	171

Description	171
Arguments	171
Input	171
Output	172
Usage	172
Related Functions	173
Example	173
Structured Text	173
Ladder Diagram	173
1.2.2.2 MC_TouchProbe (Function Block)	173
Description	173
Arguments	174
Input	174
Output	176
Usage	176
Related Functions	176
Example	177
Structured Text	177
Ladder Diagram	177
1.2.3 Info	177
1.2.3.1 MC_ReadActPos (Function Block)	177
Description	177
Arguments	178
Input	178
Output	178
Example	178
Structured Text	178
Ladder Diagram	179
1.2.3.2 MC_ReadActVel (Function Block)	179
Description	179
Arguments	179
Input	179
Output	179
Example	180
Structured Text	180
Ladder Diagram	180

1.2.3.3	MC_ReadAxisErr	180
	Description	180
	Arguments	181
	Input	181
	Output	181
	Example	182
	Structured Text	182
	Ladder Diagram	182
1.2.3.4	MC_ReadBoolPar (Function Block)	182
	Description	182
	Arguments	183
	Input	183
	Output	183
	Example	184
	Structured Text	184
	Ladder Diagram	184
1.2.3.5	MC_ReadParam (Function Block)	184
	Description	184
	Arguments	184
	Input	184
	Output	185
	Example	185
	Structured Text	185
	Ladder Diagram	186
1.2.3.6	MC_ReadStatus	186
	Description	186
	Arguments	186
	Input	186
	Output	187
	Example	188
	Structured Text	188
	Ladder Diagram	188
1.2.3.7	MC_WriteBoolPar (Function Block)	188
	Description	188
	Arguments	188
	Input	188

Output	189
Example	189
Structured Text	189
Ladder Diagram	190
1.2.3.8 MC_WriteParam (Function Block)	190
Description	190
Arguments	190
Input	190
Output	191
Example	191
Structured Text	191
Ladder Diagram	191
1.2.4 PLCopenMotion	191
1.2.4.1 MC_Halt (Function Block)	191
Description	191
Time Diagram	192
Arguments	192
Input	192
Output	193
Example	194
Structured Text	194
Ladder Diagram	194
1.2.4.2 MC_MoveAbsolute (Function Block)	194
Description	194
Time Diagram	195
Arguments	196
Input	196
Output	199
Example	200
Structured Text	200
Ladder Diagram	200
1.2.4.3 MC_MoveAdditive (Function Block)	200
Description	200
Time Diagram	201
Arguments	202
Input	202

Output	203
Example	204
Structured Text	204
Ladder Diagram	204
1.2.4.4 MC_MoveRelative	204
Description	204
Time Diagram	204
Arguments	206
Input	206
Output	207
Example	208
Structured Text	208
Ladder Diagram	208
1.2.4.5 MC_MoveSuperimp (Function Block)	208
Description	208
Time Diagram	209
Arguments	210
Input	210
Output	211
Example	211
Structured Text	211
Ladder Diagram	212
1.2.4.6 MC_MoveVelocity (Function Block)	212
Description	212
Time Diagram	212
Arguments	213
Input	213
Output	214
Example	215
Structured Text	215
Ladder Diagram	215
1.2.4.7 MC_SetOverride (Function Block)	215
Description	215
Arguments	215
Input	215
Output	216

Example	216
Structured Text	216
Ladder Diagram	216
1.2.5 Profile	217
1.2.5.1 MC_CamIn	217
Description	217
Arguments	217
Input	217
Output	219
Usage	220
Related Functions	220
Examples	220
Structured Text	220
Ladder Diagram	221
Example 1	221
Example 2	222
Example 3	223
1.2.5.2 MC_CamOut	224
Description	224
Arguments	225
Input	225
Output	225
Usage	226
Related Functions	226
Example	226
Structured Text	226
Ladder Diagram	227
1.2.5.3 MC_CamTbiSelect	227
Description	227
Arguments	227
Input	227
Output	228
Usage	228
Related Functions	229
Example	229
Structured Text	229

Ladder Diagram	229
1.2.5.4 MC_GearIn	229
Description	230
Time Diagram	231
Arguments	231
Input	231
Output	232
Example	233
Structured Text	233
Ladder Diagram	233
1.2.5.5 MC_GearInPos	233
Description	233
Time Diagram	235
Arguments	235
Input	235
Output	236
Example	237
Structured Text	237
Ladder Diagram	238
1.2.5.6 MC_GearOut	238
Description	238
Arguments	239
Input	239
Output	239
Example	240
Structured Text	240
Ladder Diagram	240
1.2.5.7 MC_Phasing	240
Description	240
Arguments	241
Input	241
Output	242
Example	242
Structured Text	242
Ladder Diagram	243
1.2.5.8 MC_SyncSlaves	243

Description	243
Arguments	243
Input	243
Output	244
Usage	244
Related Functions	244
Example	245
Structured Text	245
Ladder Diagram	245
1.2.6 Reference	245
1.2.6.1 MC_Reference (Function Block)	245
Description	245
Arguments	246
Input	246
Output	248
Usage	248
Related Functions	249
Example	249
Structured Text	249
Ladder Diagram	250
1.2.6.2 MC_SetPosition (Function)	250
Description	250
Arguments	250
Input	250
Output	251
Example	251
Structured Text	251
Ladder Diagram	252
2 Fieldbus Library	253
2.1 EtherCAT Library	254
2.1.1 EtherCAT Library - Drive	256
Execution Time	256
Reason	256
Result	256
Solution	256
2.1.1.1 DriveParamRead (Function Block)	256

Description	256
Arguments	257
Input	257
Output	258
Usage	260
Related Functions	260
Example	260
Structured Text	260
2.1.1.2 DriveParamWrite (Function Block)	260
Description	260
Arguments	261
Input	261
Output	262
Usage	263
Related Functions	263
Example	263
Structured Text	263
2.1.2 EtherCAT Library - SDO	263
2.1.2.1 ECATReadSdo (Function Block)	263
Description	263
State Diagram	264
Arguments	265
Input	265
Output	266
Related Functions	267
Example	267
Structured Text	267
2.1.2.2 ECATWriteSdo (Function Block)	267
Description	267
State Diagram	268
Arguments	269
Input	269
Output	270
Related Functions	271
Example	271
Structured Text	271

Ladder Diagram	271
2.1.3 EtherCAT Library - Debug	271
2.1.3.1 ECATReadData (Function)	271
Description	271
Arguments	272
Input	272
Output	272
Related Functions	272
Example	272
Structured Text	272
2.1.3.2 ECATWriteData (Function)	272
Description	272
Arguments	273
Input	273
Output	273
Related Functions	273
2.1.3.3 ECATGetObjVal (Function)	273
2.1.4 EtherCAT Library - Status	273
2.1.4.1 ECATGetStatus (Function)	273
Description	274
Arguments	274
Input	274
Output	274
Related Functions	274
Example	274
Structured Text	274
2.1.4.2 ECATSetControl (Function)	274
Description	274
Arguments	275
Input	275
Output	275
Related Functions	275
3 System Library	277
3.1 PrintMessage (Function)	278
3.1.1 Description	278

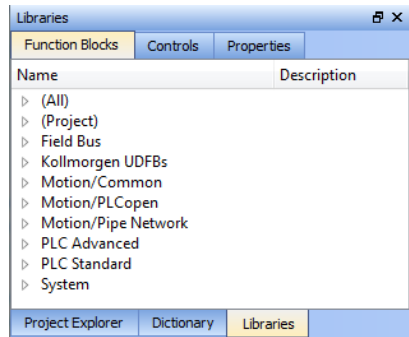
3.1.1.1	About the Source	278
3.1.1.2	About the Level	278
3.1.2	Arguments	278
3.1.2.1	Input	278
3.1.2.2	Output	279
3.1.3	Usage	279
3.1.4	Example	279
3.1.4.1	Structured Text	279
3.1.4.2	Function Block Diagram	280
Index	281
Global Support Contacts	285

This page intentionally left blank.

1 Motion Library

1.1	Motion Library / Pipe Network	38
1.2	Motion Library / PLCopen	155

This chapter covers the Motion Library (for **Pipe Network** and **PLCopen**) in the function blocks tab of the Library toolbox.



KAS function library contains ML function blocks that are used to integrate motion in a PLC program. ML function blocks can be used in 4 of the IEC 61131-3 languages: ST, FBD, FFLD and IL.

Regarding SFC/SFC programs, ML function blocks (like any other function blocks from the library) are used as part of a step/step or transition/transition which are defined with ST, FBD, FFLD or IL languages.

1.1 Motion Library / Pipe Network

The KAS IDE function library contains ML function blocks that are used to integrate motion from a Pipe Network in a PLC program. ML Function blocks are of the following types:

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

Table 1-1: List of Pipe Network FB

! IMPORTANT Pipe Network code is generated automatically by the compiler, you should not try to modify it.

Hover the mouse above a Pipe Block in the figure below to display its description at the bottom....



1.1.1 Motion Library

Name	Description	Return type
MLMotionInit	Initializes the motion library. Must be called before any other Motion Library function. Returns TRUE if the function succeeded. BasePeriod is the duration of one motion cycle in microseconds.	BOOL
MLMotionStart	Starts the motion engine, motion bus driver, and initializes EtherCAT network to operational mode. Applicable to PLCopen and PipesNetwork motion engines. Returns TRUE if the function succeeded.	BOOL
MLMotionStatus	Returns the status of the motion engine 0: Not initialized 1: Running 2: Stopped 3: Error	None
MLMotionStop	Stops the motion bus driver, motion engine, and EtherCAT network operation, resulting in the EtherCAT network transitioning to the Init state. Returns TRUE if the function succeeded.	BOOL
MLMotionSysTime	Prints the system time to the log	BOOL
MLMotionCycleTime		

1.1.2 Motion Library - Pipe Network

Name	Description	Return type
MLPipeAct	Activates a pipe	BOOL

Name	Description	Return type
MLPipeAddBlock	Adds a Pipe Block to a pipe	BOOL
MLPipeCreate	Creates a new pipe object	None
MLPipeDeact	Deactivates a pipe	BOOL

1.1.2.1 MLPipeAct

Description

Activates a pipe. 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 figure below 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 pipe ends with the second converter, and has a CAM Pipe Block to modify the input values from the Master.

Once a Pipe is activated then 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.

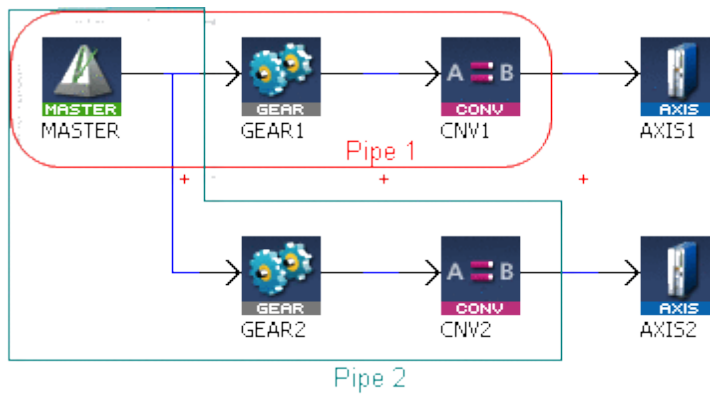


Figure 1-1: MLPipeAct

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. Therefore, in a multi-pipe program only one command can be used to activate Pipes instead of writing code for each Pipe separately.

Arguments

Input

PipeID	Description
	ID number of a created Pipe object
	Data type: DINT
	Range: [-2147483648, 2147483648]
	Unit: n/a
	Default: —

Output

Default (.Q)	Description
	Returns TRUE if the Pipe is activated
	Data type: BOOL
	Unit: n/a

Return Type

BOOL

Related Functions

MLPipeDeact

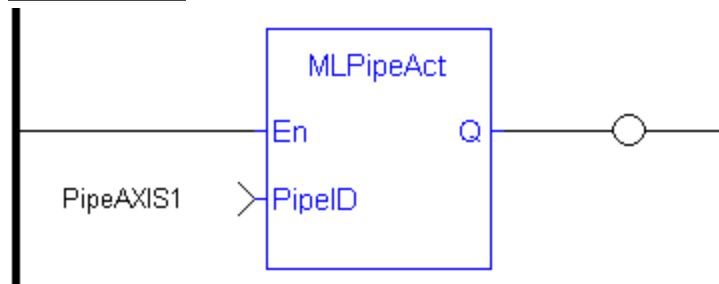
MLCNVConnect

PipeNetwork(MLPN_ACTIVATE)

MLPipeAddBlock

ExampleStructured Text

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

Ladder DiagramFunction Block Diagram**1.1.2.2 MLPipeAddBlock**Description

Add a Pipe Block to a pipe. 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 figure below shows two Pipes, both with the same Master Input Pipe Block. If a user were to create the Pipe 1 below without using the Graphical Engine, they would use the following commands once a Pipe and the Pipe Blocks have been created.

```
MLPipeAddBlock( PipeAXIS1, MASTER);
```

```
MLPipeAddBlock( PipeAXIS1, MyGear);
```

```
MLPipeAddBlock( PipeAXIS1, CNV1);
```

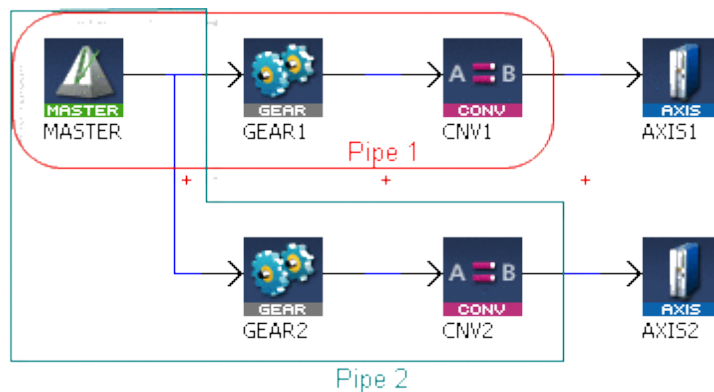



Figure 1-2: MLPipeAddBlock

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). Therefore, when using the Pipe Network graphical engine to create Pipe Blocks the user does not have to manually add MLPipeAddBlock commands to the Project.

Arguments

Input

PipeID	Description	ID number of a created Pipe
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
BlockID	Description	ID number of a created Pipe object to add to the selected Pipe
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Default (.Q)	Description	Returns TRUE if the Pipe Block is added to the Pipe
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

Related Functions

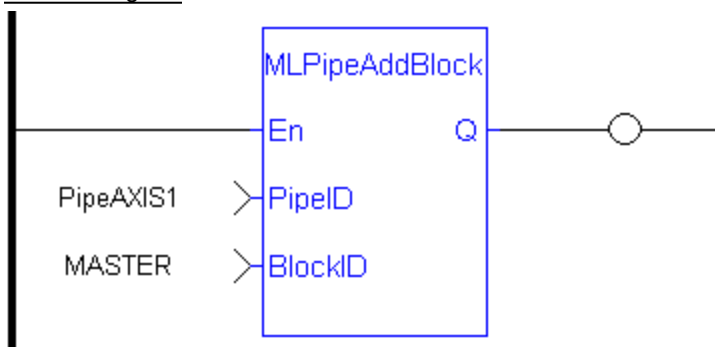
- PipeNetwork(MLPN_CREATE_OBJECTS)
- MLPipeAct
- MLPipeCreate
- MLPipeDeact

Example

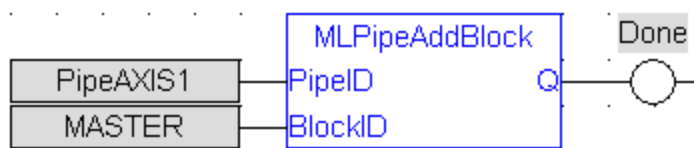
Structured Text

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

Ladder Diagram



Function Block Diagram



1.1.2.3 MLPipeCreate

Description

Create a new pipe object. 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 figure below shows two Pipes, both with the same Master Input Pipe Block.

NOTE

Pipes are normally created in the Pipe Network using the graphical engine. Then you do not have to add MLPipeCreate function blocks to their programs. Pipes are created graphically, and the code with MLPipeCreate commands are automatically generated and called in a program with PipeNetwork(MLPN_CREATE_OBJECTS).

Arguments

Input

Name	Description	Desired name for the newly created Pipe
	Data type	String
	Range	—
	Unit	n/a
	Default	—

Output

ID	Description	Assigned ID number of the created Pipe
	Data type	DINT
	Unit	n/a
	Default	—

Related Functions

PipeNetwork(MLPN_CREATE_OBJECTS)

MLPipeAddBlock

MLPipeAct

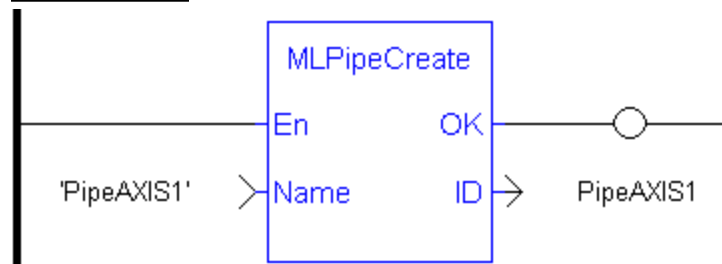
MLPipeDeact

Example

Structured Text

```
//Create a new pipe
PipeAXIS1 := MLPipeCreate( 'PipeAXIS1' );
```

Ladder Diagram



Function Block Diagram



1.1.2.4 MLPipeDeact

Description

Deactivates a pipe. 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 figure below 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 pipe ends with the second converter, and has a CAM Pipe Block to modify the input values from the Master.

Once a Pipe is activated then 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.

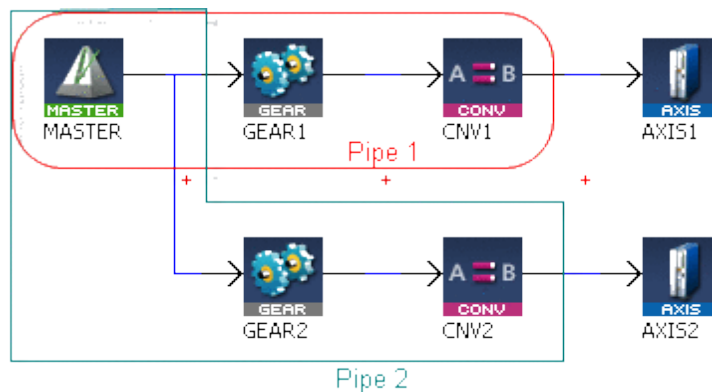


Figure 1-3: MLPipeDeact

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. Therefore, in a multi-pipe program only one command can be used to deactivate Pipes instead of writing code for each Pipe separately.

Arguments

Input

PipeID	Description	ID number of a created Pipe object
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Default (.Q)	Description	Returns TRUE if the Pipe is deactivated
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

Related Functions

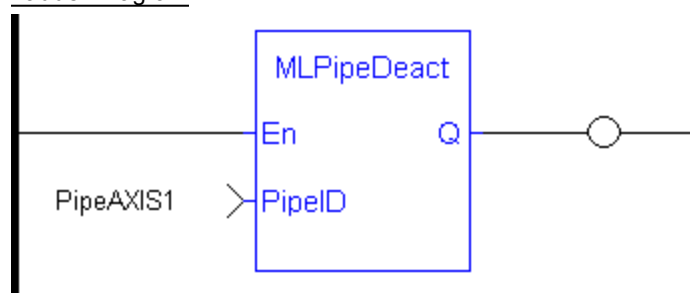
- MLPipeAct
- MLCNVDisconnect
- PipeNetwork(MLPN_DEACTIVATE)
- MLPipeAddBlock

Example

Structured Text

```
//Deactivate a Pipe
MLPipeDeact ( PipeAXIS1 );
```

Ladder Diagram



Function Block Diagram



1.1.3 Motion Library - Block

Name	Description	Return type
"MLBikCreate" (see page 45)	Creates a new Pipe Block object	None
"MLBikIsReady" (see page 49)	Checks if a Pipe Block currently has a function running	BOOL
"MLBikReadModPos" (see page 48)	Gets the value of the period of a block in user units	None
"MLBikReadOutVal" (see page 47)	Gets the output value of a selected Pipe Block	None
"MLBikWriteModPos" (see page 50)	Sets the value of the period of a block in user units	BOOL

1.1.3.1 MLBikCreate

Description

Creates a new Pipe Block object. Before a Pipe Block is Initialized the block needs to be created and assigned an ID number. MLBikCreate function block 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. Then you do not have to add MLBikCreate function blocks to their programs. Blocks are created graphically, and the code with MLBikCreate commands are automatically generated and called in a program with Pipe Network(MLPN_CREATE_OBJECTS).

TIP

This function must be called or executed before MLMotionStart is called.

Arguments

Input

Name	Description	Desired name for the newly created Pipe Block
Data type		String
Range		—
Unit		n/a

	Default	—
Type	Description	Type of Pipe Block to create (ex. MASTER, GEAR, PHASER, etc.)
	Data type	String
	Range	—
	Unit	n/a
	Default	—

Output

ID	Description	Assigned ID number of the created Block
	Data type	DINT
	Unit	n/a
	Default	—

Related Functions

PipeNetwork(MLPN_CREATE_OBJECTS)

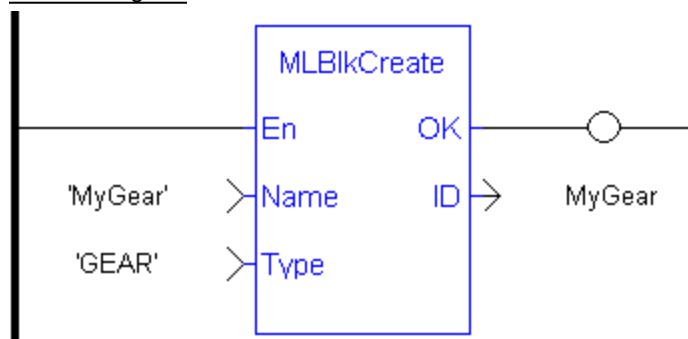
MLAxisInit

Example

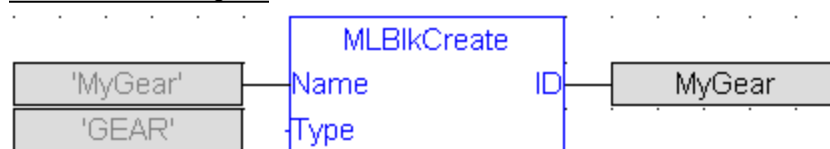
Structured Text

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

Ladder Diagram



Function Block Diagram



1.1.3.2 MLBlkReadOutVal

Description

Get the output value a selected Pipe Block.

Arguments

Input

ID	Description	ID number of a created Pipe Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Value	Description	Current output value of the selected Pipe Block
	Data type	LREAL
	Unit	n/a
	Default	—

Related Functions

MLBlkReadModPos

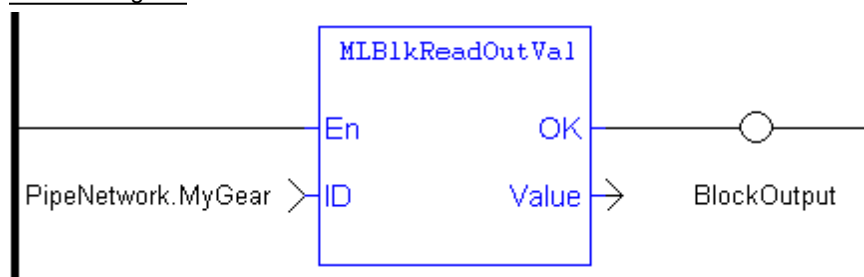
MLBlkCreate

Example

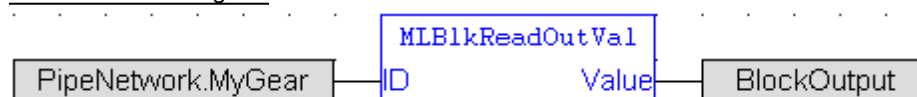
Structured Text

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

Ladder Diagram



Function Block Diagram



1.1.3.3 MLBlkReadModPos

Description

Get the value of the period of a block in user units. The output value of a block is reset each time it reaches its period value.

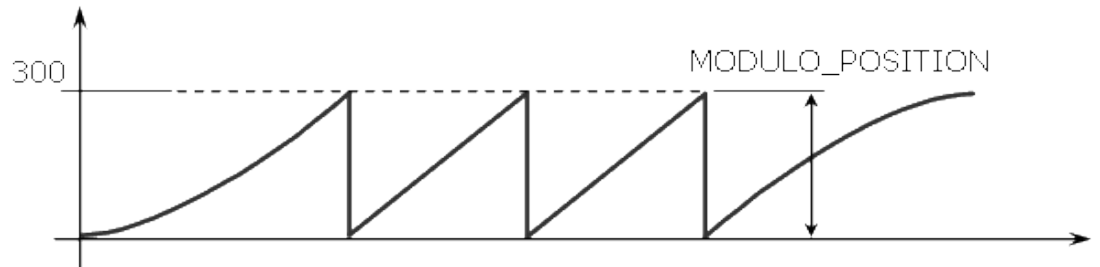


Figure 1-4: MLBlkReadModPos

Arguments

Input

ID	Description	ID number of a created Pipe Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

ModuloPosition	Description	Current Period Value for selected Pipe Block
	Data type	LREAL
	Unit	User unit
	Default	—

Related Functions

MLBlkWriteModPos

MLBlkCreate

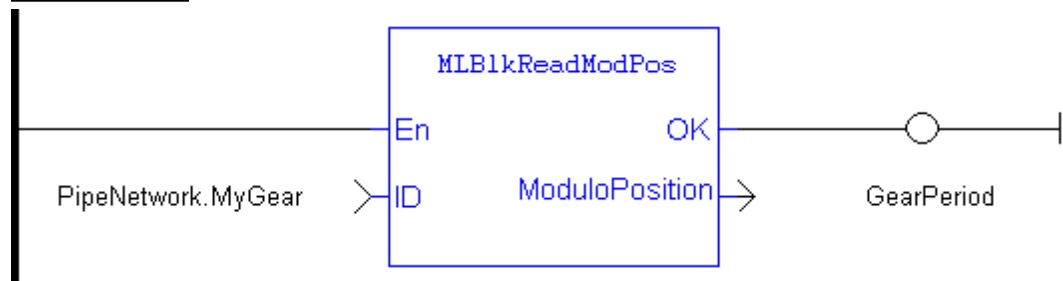
MLBlkReadOutVal

Example

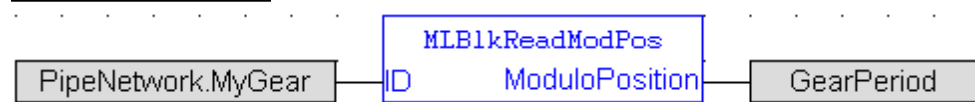
Structured Text

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


Ladder Diagram



Function Block Diagram



1.1.3.4 MLBlkIsReady

Description

Check if a block is ready. Returns FALSE if the selected Pipe Block has a function running. Returns TRUE if no function of a specified Pipe Block is running.

NOTE Same return value as the .Q output of a specific function itself

Arguments

Input

ID	Description	ID number of a created Pipe Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Default (.Q)	Description	Returns TRUE if no function of a specified Pipe Block is running.
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

Related Functions

MLBlkReadOutVal

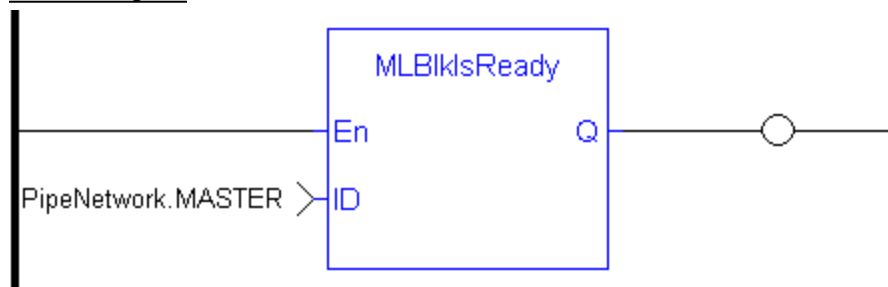
MLBlkReadModPos

Example

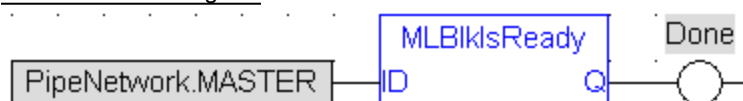
Structured Text

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

Ladder Diagram



Function Block Diagram



1.1.3.5 MLBlkWriteModPos

Description

Set the value of the period of a block in user units. The output value of a block is reset each time it reaches its period value.

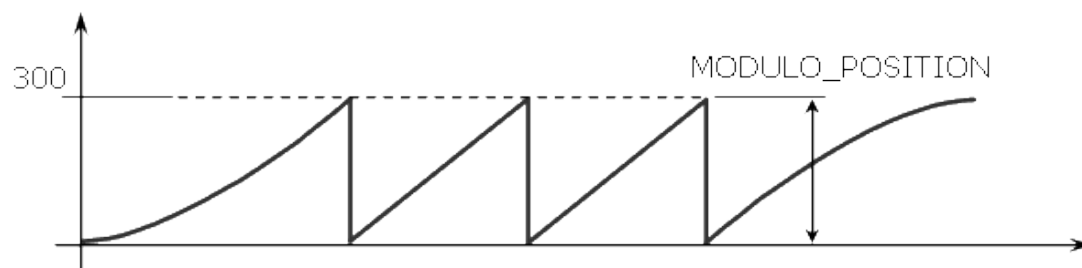


Figure 1-5: MLBlkReadModPos

Arguments

Input

Input	Description
ID	ID number of a created Pipe Block
	Data type DINT
	Range [-2147483648, 2147483648]
	Unit n/a
	Default —

ModuloPosition	Description	Desired new Period Value for selected Pipe Block
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

Default (.Q)	Description	Returns TRUE if the function block executes
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

Related Functions

MLBlkReadModPos

MLBlkCreate

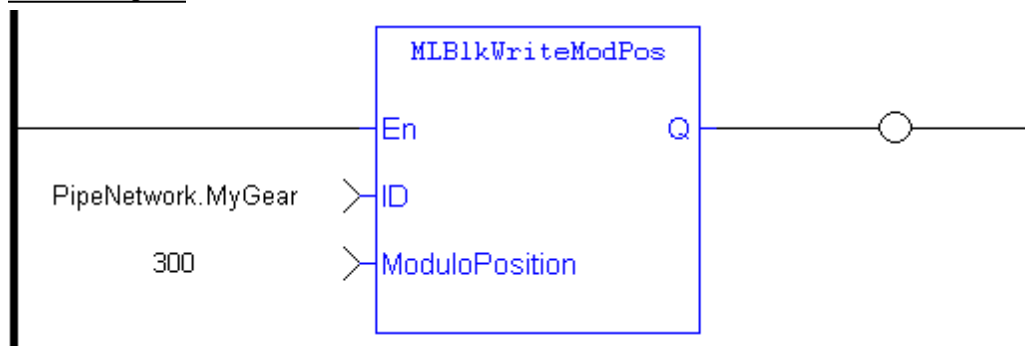
MLBlkReadOutVal

Example

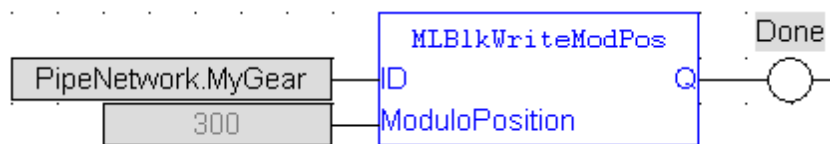
Structured Text

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

Ladder Diagram



Function Block Diagram



1.1.4 Motion Library - Adder

Name	Description	Return type
MLAddInit	Initializes an Adder Pipe Block with user-defined settings	BOOL
MLAddReadOff1	Returns the offset value of the first entry of an Adder block	None
MLAddReadOff2	Returns the offset value of the second entry of an Adder block	None
MLAddReadRatio1	Returns the ratio value of the first entry of an Adder block	None
MLAddReadRatio2	Returns the ratio value of the second entry of an Adder block	None
MLAddWriteInput	Sets the source of an input of an adder Pipe Block	BOOL
MLAddWriteOff1	Sets the offset value of the first entry of the Adder block	BOOL
MLAddWriteOff2	Sets the offset value of the second entry of the Adder block	BOOL
MLAddWriteRat1	Sets the ratio value of the first entry of the Adder block	BOOL
MLAddWriteRat2	Sets the ratio value of the second entry of the Adder block	BOOL

1.1.4.1 MLAddInit

Description

Initializes an Adder Pipe Block for use in a PLC Program. Function block is automatically called if an Adder Block is added to the Pipe Network, with user-defined settings 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 still need to be selected using the MLAddWriteInput function block or graphically using the Pipe Network.

$$\text{Adder Block Output} = \text{Ratio1} * \text{Input1} + \text{Offset1} + \text{Ratio2} * \text{Input2} + \text{Offset2}$$

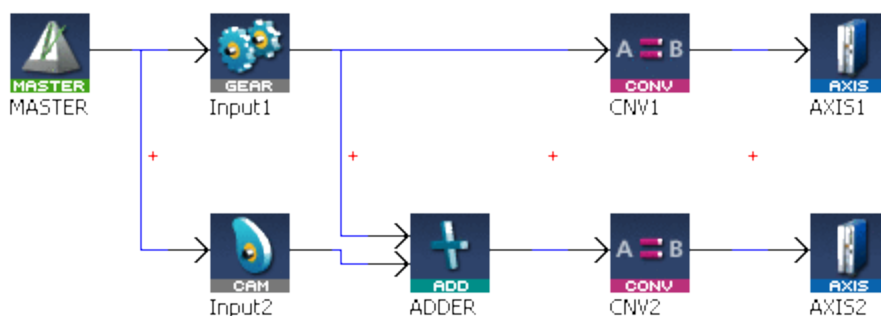


Figure 1-6: MLAddInit

NOTE Adder objects are normally created in the Pipe Network using the graphical engine. Then you do not have to add MLAddInit function blocks to their

NOTE

programs. Parameters are entered directly in pop-up windows, and the code is then automatically added to the current project.

Arguments

	<u>Input</u>		
BlockID	Description	ID number of a created Pipe Block	
	Data type	DINT	
	Range	[-2147483648, 2147483648]	
	Unit	n/a	
	Default	—	
Ratio1	Description	Sets the Ratio value of the first entry of an Adder object	
	Data type	LREAL	
	Range	—	
	Unit	n/a	
	Default	—	
Offset1	Description	Sets the Offset value of the first entry of an Adder object	
	Data type	LREAL	
	Range	—	
	Unit	n/a	
	Default	—	
Ratio2	Description	Sets the Ratio value of the second entry of an Adder object	
	Data type	LREAL	
	Range	—	
	Unit	n/a	
	Default	—	
Offset2	Description	Sets the Offset value of the second entry of an Adder object	
	Data type	LREAL	
	Range	—	
	Unit	n/a	
	Default	—	
Default (.Q)	<u>Output</u>		
	Description	Returns TRUE if the Adder Pipe Block is initialized	
	Data type	BOOL	
	Unit	n/a	

Return Type

BOOL

Related Functions

MLBikCreate

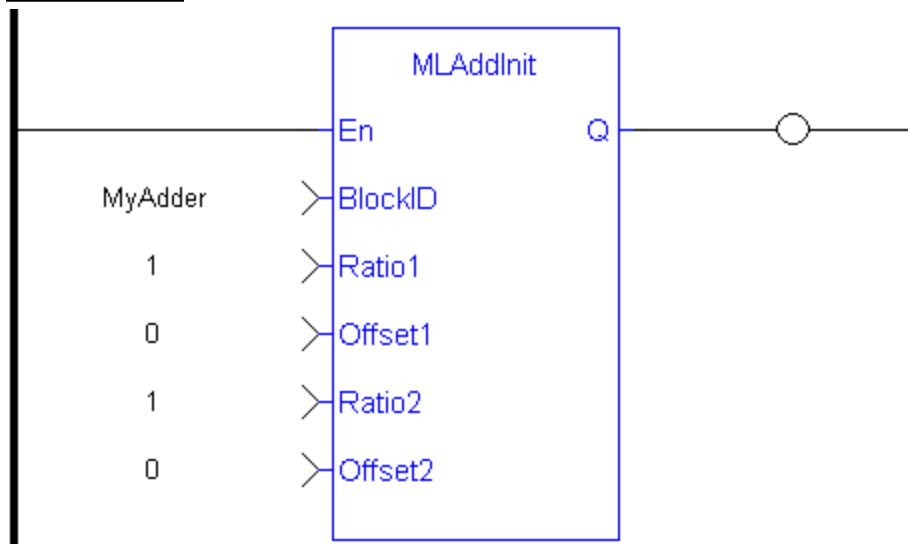
MAddWriteInput
 MAddReadOff1
 MAddReadRatio1

Example

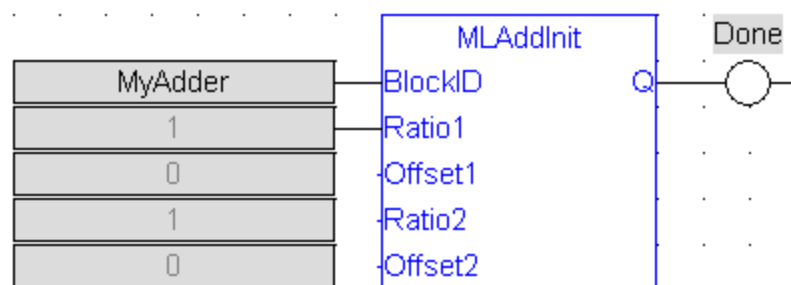
Structured Text

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

Ladder Diagram



Function Block Diagram



1.1.4.2 MAddReadOff1

Description

Returns the offset value of the first entry of an Adder block. Can change the offset value with MAddWriteOff1 function block. Offset1 shifts the value of the first input to the block before its added to the second input.

$$\text{Adder Block Output} = \text{Ratio1} * \text{Input1} + \text{Offset1} + \text{Ratio2} * \text{Input2} + \text{Offset2}$$

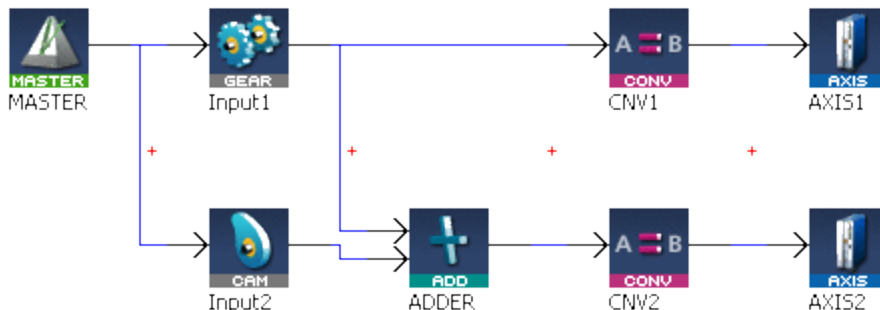


Figure 1-7: MLAddReadOff1

Arguments

	Input	Description	
BlockID		ID number of an initiated Adder object	
		Data type	DINT
		Range	[-2147483648, 2147483648]
		Unit	n/a
		Default	—
Offset		Description	Returns the offset value of the first entry of an Adder object
		Data type	LREAL
		Unit	n/a

Related Functions

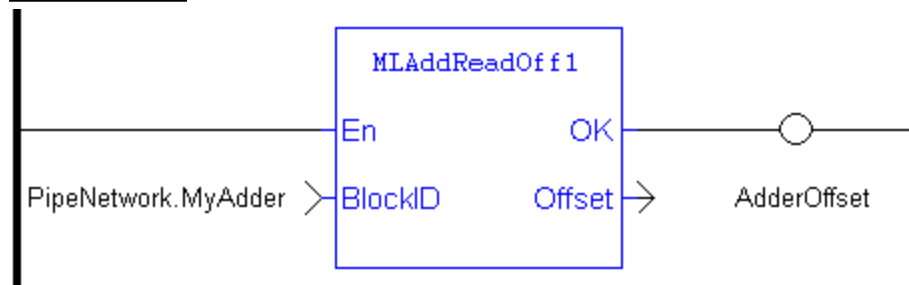
- MLAddWriteOff1
- MLAddReadOff2
- MLAddReadRatio1
- MLAddWriteRat1

Example

Structured Text

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

Ladder Diagram



Function Block Diagram



1.1.4.3 MAddReadOff2

Description

Returns the offset value of the second entry of an Adder block. Can change the offset value with MAddWriteOff2 function block. Offset2 shifts the value of the second input to the block before its added to the first input.

$$\text{Adder Block Output} = \text{Ratio1} * \text{Input1} + \text{Offset1} + \text{Ratio2} * \text{Input2} + \text{Offset2}$$

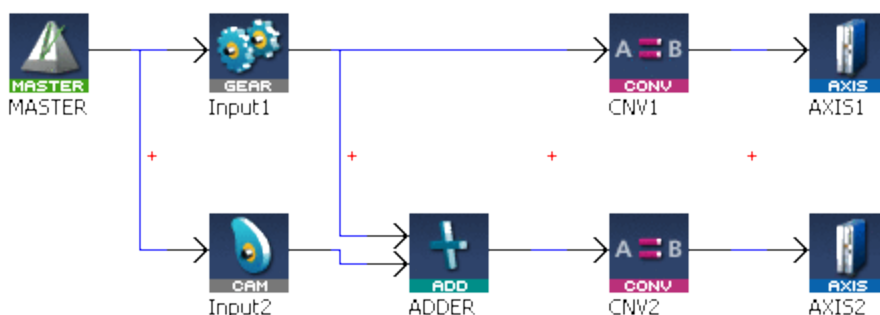


Figure 1-8: MAddReadOff2

Arguments

	Input		
BlockID		Description	ID number of an initiated Adder object
		Data type	DINT
		Range	[-2147483648, 2147483648]
		Unit	n/a
		Default	—
Offset		Description	Returns the offset value of the second entry of an Adder object
		Data type	LREAL
		Unit	n/a

Related Functions

- MLAddWriteOff2
- MLAddReadOff1
- MLAddReadRatio2
- MLAddWriteRat2

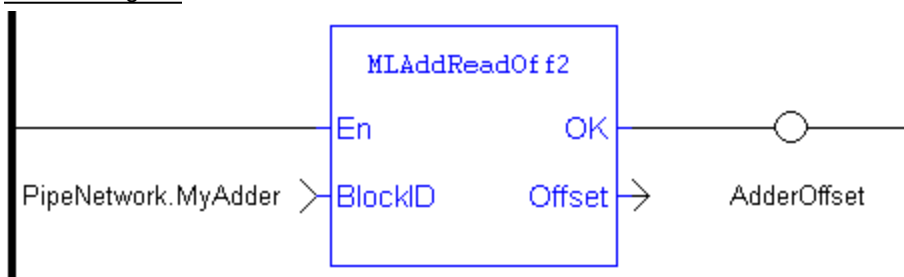
Example

Structured Text

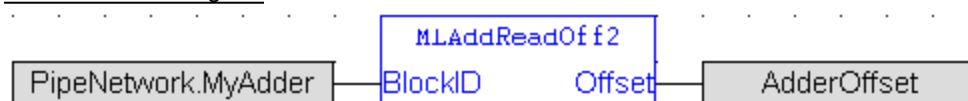
```
//Save the offset value of second entry to the Adder
block

AdderOffset := MLAddReadOff2( PipeNetwork.MyAdder );
```

Ladder Diagram



Function Block Diagram



1.1.4.4 MLAddReadRatio1

Description

Returns the ratio value of the first entry of an Adder block. Can change the ratio value with MLAddWriteRat1 function block. Ratio1 amplifies the value of the first input to the block before its added to the second input.

$$\text{Adder Block Output} = \text{Ratio1} * \text{Input1} + \text{Offset1} + \text{Ratio2} * \text{Input2} + \text{Offset2}$$

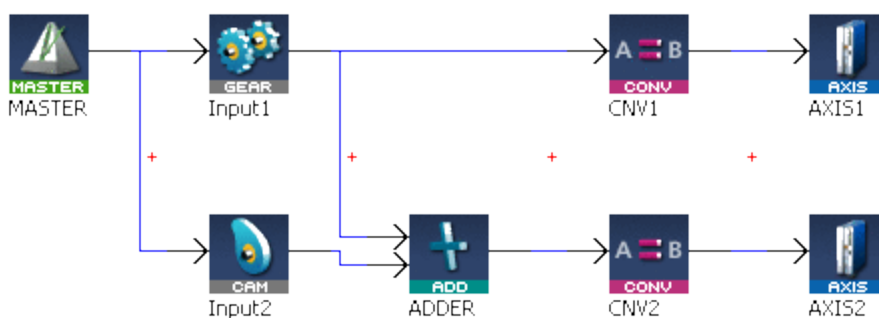


Figure 1-9: MLAddReadRatio1

Arguments

BlockID	Description	ID number of an initiated Adder object
<u>Input</u>	Data type	DINT
	Range	[-2147483648, 2147483648]

	Unit	n/a
	Default	—
Ratio	<u>Output</u>	
	Description	Returns the Ratio value of the first entry of an Adder object
	Data type	LREAL
	Unit	n/a

Related Functions

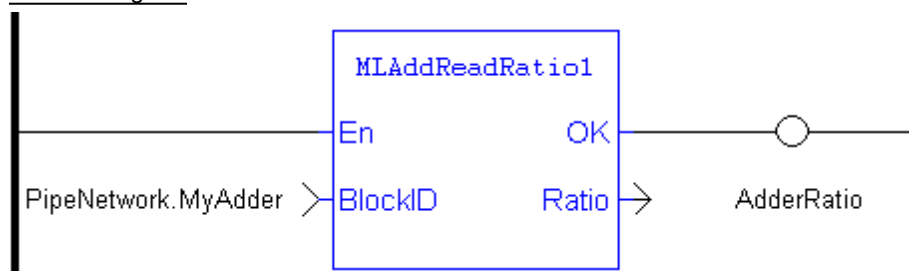
- MAddWriteRat1
- MAddReadRatio2
- MAddReadOff1
- MAddReadOff2

Example

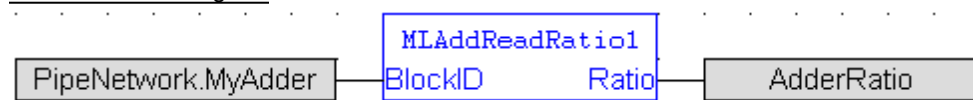
Structured Text

```
//Save the ratio value of first entry to the Adder block
AdderRatio := MAddReadRatio1( PipeNetwork.MyAdder );
```

Ladder Diagram



Function Block Diagram



1.1.4.5 MAddReadRatio2

Description

Returns the ratio value of the second entry of an Adder block. Can change the ratio value with MAddWriteRat2 function block. Ratio2 amplifies the value of the second input to the block before its added to the first input.

$$\text{Adder Block Output} = \text{Ratio1} * \text{Input1} + \text{Offset1} + \text{Ratio2} * \text{Input2} + \text{Offset2}$$

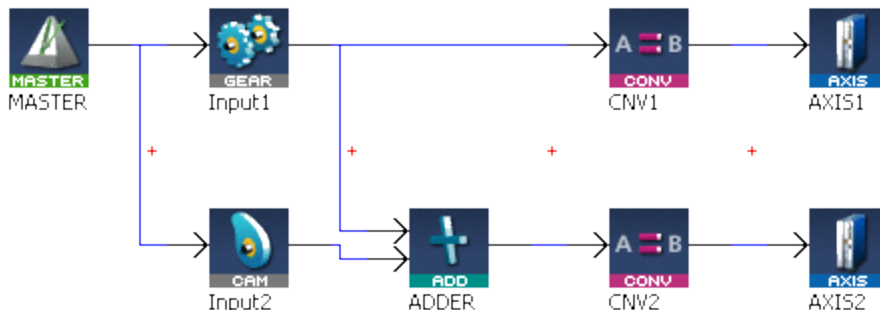


Figure 1-10: MLAddReadRatio2

Arguments

		<u>Input</u>	
BlockID	Description	ID number of an initiated Adder object	
	Data type	DINT	
	Range	[-2147483648, 2147483648]	
	Unit	n/a	
	Default	—	
		<u>Output</u>	
Ratio	Description	Returns the Ratio value of the second entry of an Adder object	
	Data type	LREAL	
	Unit	n/a	

Related Functions

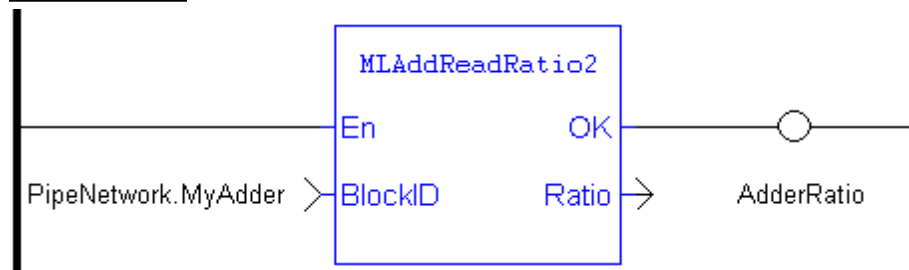
- MLAddWriteRat2
- MLAddReadRatio1
- MLAddReadOff1
- MLAddReadOff2

Example

Structured Text

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

Ladder Diagram



Function Block Diagram



1.1.4.6 MAddWriteInput

Description

Sets the source of an input of an adder Pipe Block. Function block is automatically called if an Adder Block is connected to other blocks in the Pipe Network.

$$\text{Adder Block Output} = \text{Ratio1} * \text{Input1} + \text{Offset1} + \text{Ratio2} * \text{Input2} + \text{Offset2}$$

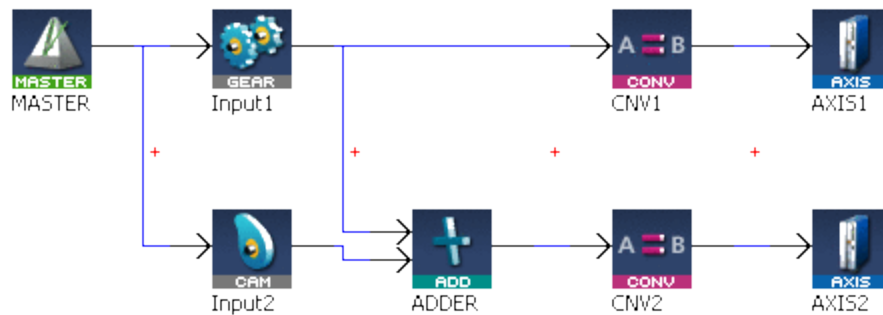


Figure 1-11: MAddWriteInput

NOTE Adder objects are normally created in the Pipe Network using the graphical engine. Then you do not have to add MAddWriteInput function blocks to their programs. Blocks are connected with lines in the Pipe Network, and the code is then automatically added to the current project.

Arguments

	Input		
BlockID	Description	ID number of an initiated Adder object	
	Data type	DINT	
	Range	[-2147483648, 2147483648]	
	Unit	n/a	
	Default	—	
InputID	Description	Select first or second input to the Adder object	
	Data type	DINT	
	Range	[1, 2]	
	Unit	n/a	
	Default	—	
InputBlockID	Description	ID number of an initiated Pipe Block which is an input to the Adder object	
	Data type	DINT	
	Range	[-2147483648, 2147483648]	

	Unit	n/a
	Default	—
Default (.Q)	<u>Output</u>	
	Description	Returns TRUE if the input to the Adder object is set
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

Related Functions

MLBkCreate

MLAddInit

MLAddReadOff1

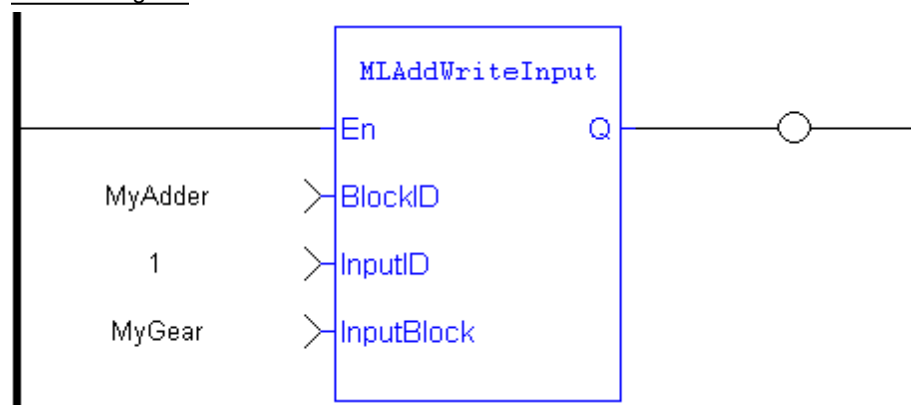
MLAddReadRatio1

Example

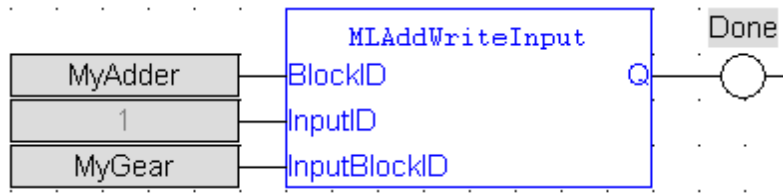
Structured Text

```
//Set the inputs to an Adder object
MLAddWriteInput( MyAdder, 1, GEAR );
DoneGEAR :=TRUE;
MLAddWriteInput( MyAdder, 2, CAM );
DoneCAM :=TRUE;
```

Ladder Diagram



Function Block Diagram



1.1.4.7 MAddWriteOff1

Description

Set the offset value of the first entry of the Adder block. Offset1 shifts the value of the first input to the block before its added to the second input.

$$\text{Adder Block Output} = \text{Ratio1} * \text{Input1} + \text{Offset1} + \text{Ratio2} * \text{Input2} + \text{Offset2}$$

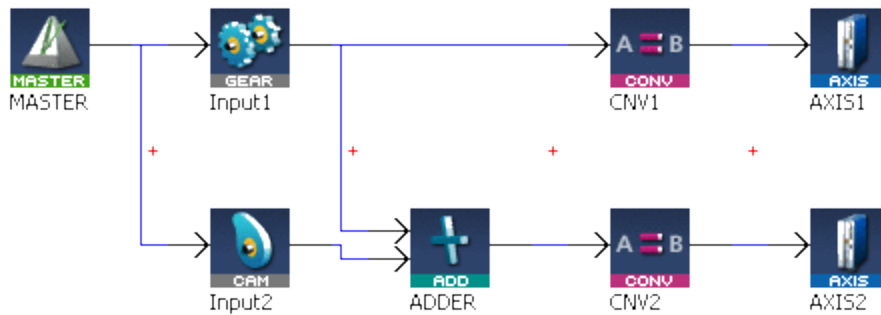


Figure 1-12: MAddWriteOff1

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

Arguments

		Input	
BlockID	Description	ID number of an initiated Adder object	
	Data type	DINT	
	Range	[-2147483648, 2147483648]	
	Unit	n/a	
	Default	—	
Offset	Description	Desired new value for the Adder Object's Offset1	
	Data type	LREAL	
	Range	—	
	Unit	n/a	
	Default	—	
		Output	
Default (.Q)	Description	Returns TRUE if the Offset value for input one is set	
	Data type	BOOL	
	Unit	n/a	

Return Type

BOOL

Related Functions

MAddReadOff1

MAddWriteOff2

MAddReadRatio1

MAddWriteRat1

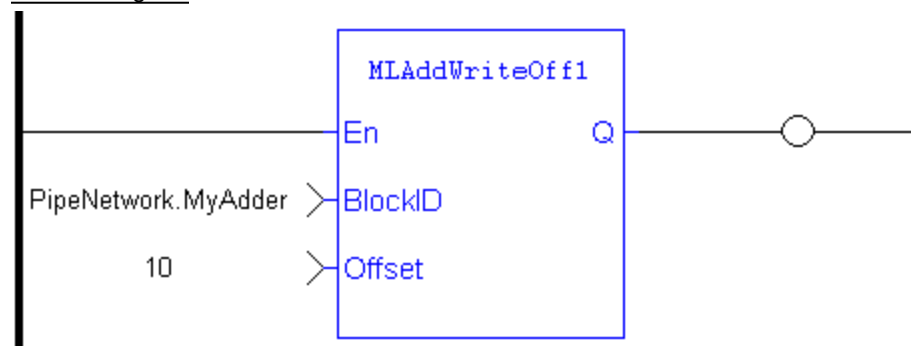
Example

Structured Text

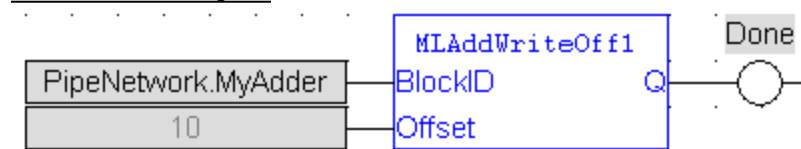
```
//Change the offset value of first entry to the Adder
block to 10

MAddWriteOff1( PipeNetwork.MyAdder, 10 );
```

Ladder Diagram



Function Block Diagram



1.1.4.8 MAddWriteOff2

Description

Set the offset value of the second entry of the Adder block. Offset2 shifts the value of the second input to the block before its added to the first input.

$$\text{Adder Block Output} = \text{Ratio1} * \text{Input1} + \text{Offset1} + \text{Ratio2} * \text{Input2} + \text{Offset2}$$

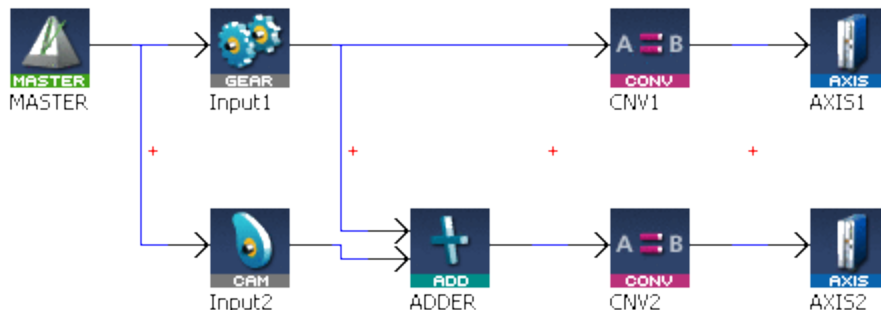


Figure 1-13: MAddWriteOff2

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

Arguments

	<u>Input</u>		
BlockID	Description	ID number of an initiated Adder object	
	Data type	DINT	
	Range	[-2147483648, 2147483648]	
	Unit	n/a	
	Default	—	
Offset	Description	Desired new value for the Adder Object's Offset2	
	Data type	LREAL	
	Range	—	
	Unit	n/a	
	Default	—	
	<u>Output</u>		
Default (.Q)	Description	Returns TRUE if the Offset value for input two is set	
	Data type	BOOL	
	Unit	n/a	

Return Type
 BOOL

Related Functions

- MAddReadOff2
- MAddWriteOff1
- MAddReadRatio2
- MAddWriteRat2

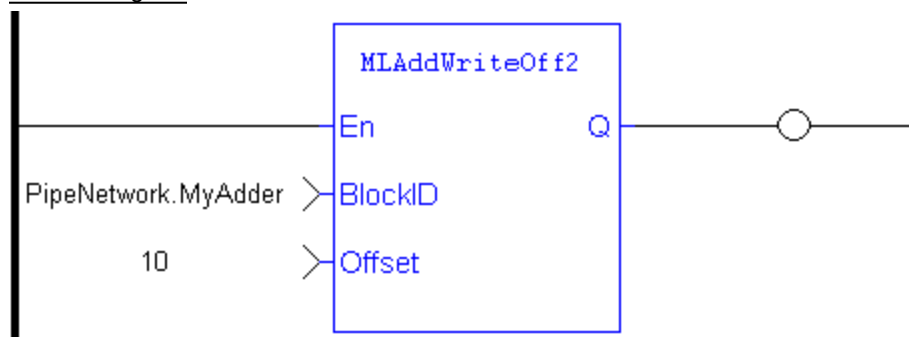
Example

Structured Text

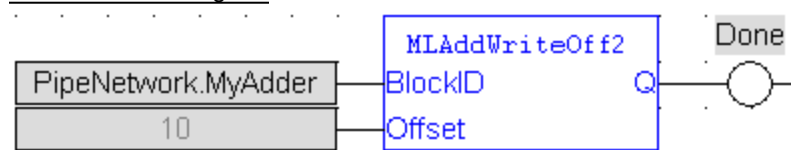
```
//Change the offset value of second entry to the Adder
block to 10

MLAddWriteOff2( PipeNetwork.MyAdder, 10 );
```

Ladder Diagram



Function Block Diagram



1.1.4.9 MAddWriteRat1

Description

Set the ratio value of the first entry of the Adder block. Ratio1 amplifies the value of the first input to the block before its added to the second input.

$$\text{Adder Block Output} = \text{Ratio1} * \text{Input1} + \text{Offset1} + \text{Ratio2} * \text{Input2} + \text{Offset2}$$

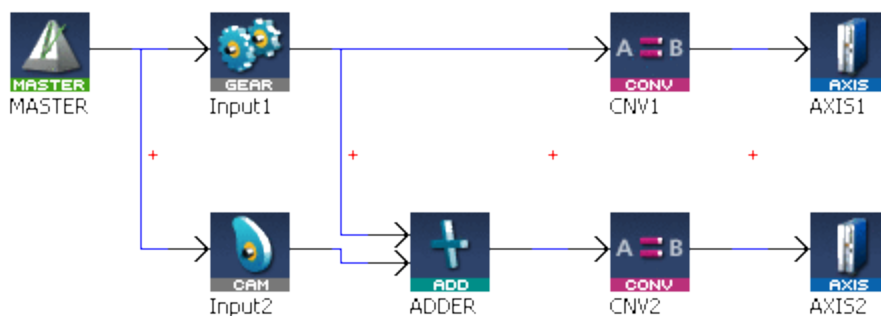


Figure 1-14: MAddWriteRat1

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

Arguments

	<u>Input</u>		
BlockID		Description	ID number of an initiated Adder object
		Data type	DINT
		Range	[-2147483648, 2147483648]
		Unit	n/a
		Default	—
Ratio		Description	Desired new value for the Adder Object's Ratio1
		Data type	LREAL
		Range	—
		Unit	n/a
		Default	—
	<u>Output</u>		
Default (.Q)		Description	Returns TRUE if the Ratio value for input one is set
		Data type	BOOL
		Unit	n/a

Return Type

BOOL

Related Functions

MLAddReadRatio1

MLAddWriteRat2

MLAddReadOff1

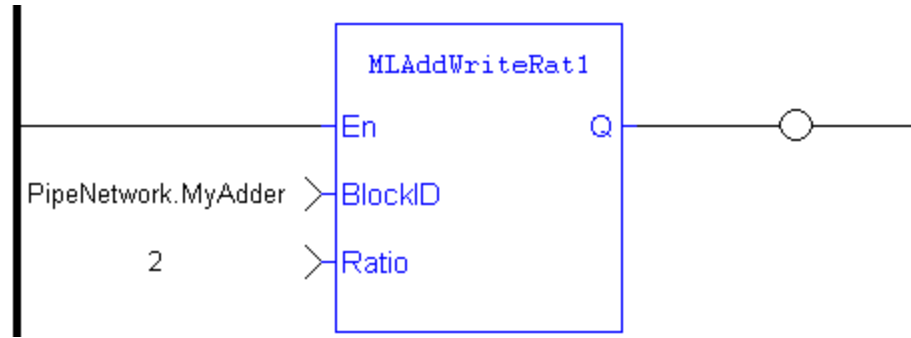
MLAddWriteOff1

ExampleStructured Text

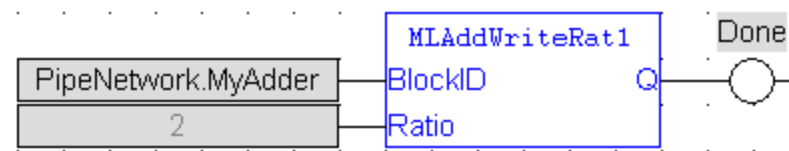
```
//Change the ratio value of first entry to the Adder
block to 2
```

```
MLAddWriteRat1( PipeNetwork.MyAdder, 2 );
```

Ladder Diagram



Function Block Diagram



1.1.4.10 MLAddWriteRat2

Description

Set the ratio value of the second entry of the Adder block. Ratio2 amplifies the value of the second input to the block before its added to the first input.

$$\text{Adder Block Output} = \text{Ratio1} * \text{Input1} + \text{Offset1} + \text{Ratio2} * \text{Input2} + \text{Offset2}$$

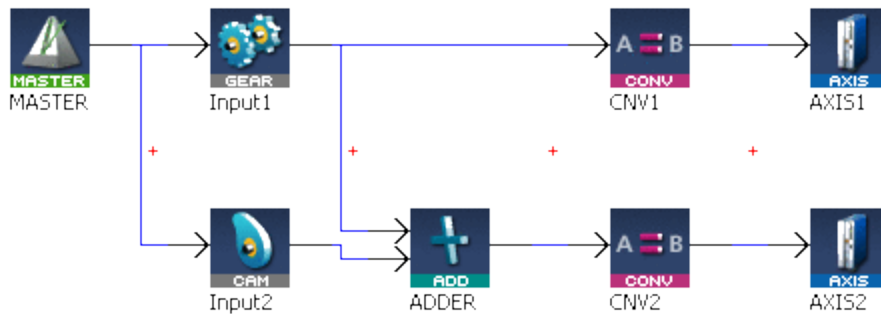


Figure 1-15: MLAddWriteRat2\

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

Arguments

		<u>Input</u>	
BlockID	Description	ID number of an initiated Adder object	
	Data type	DINT	
	Range	[-2147483648, 2147483648]	
	Unit	n/a	
	Default	—	
Ratio	Description	Desired new value for the Adder Object's Ratio2	

Data type	LREAL
Range	—
Unit	n/a
Default	—

Default (.Q) Output

Description	Returns TRUE if the Ratio value for input two is set
Data type	BOOL
Unit	n/a

Return Type
 BOOL

Related Functions

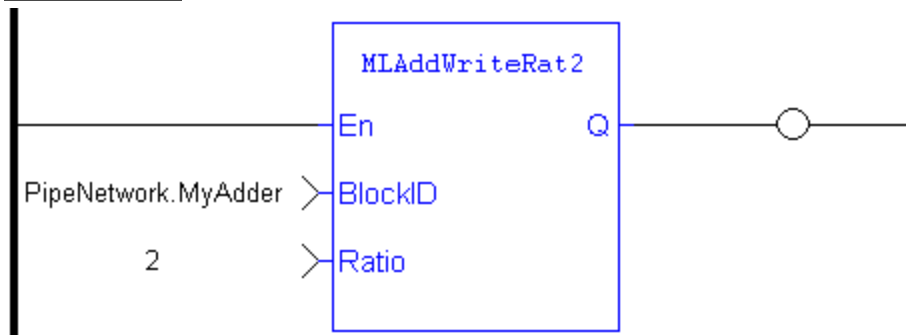
- MLAddReadRatio2
- MLAddWriteRat1
- MLAddReadOff2
- MLAddWriteOff2

Example

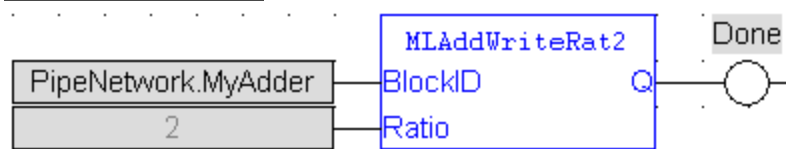
Structured Text

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

Ladder Diagram



Function Block Diagram



1.1.5 Motion Library - Axis


TIP

For usage example about Axis Functions, click here...

Function sorted by types:

Power Stage	Motion Control	Inquiry Functions	Position setting
MLAxisPower	MLAxisAbs	MLAxisGenPos	
MLAxisPowerDOff	MLAxisAdd	MLAxisPipePos	MLAxisReAlign
	MLAxisMoveVel	MLAxisCmdPos	
	MLAxisRel	MLAxisReadActPos	
	MLAxisStop	MLAxisFBackPos	
		MLAxisStatus	
		MLAxisReadGenStatus	
		MLAxisGenIsRdy	
		MLAxisTimeStamp	

Functions sorted in alphabetical order:

Name	Description	Return type
MLAxisAbs	Performs a move to an absolute position	BOOL
MLAxisAdd	Performs an additive move relative for a specified distance from the endpoint of the previous move	BOOL
MLAxisAddress	Returns the motion bus address of the axis	DINT
MLAxisAddTq	Sets additive torque	BOOL
MLAxisCfgFastIn	Initializes the Fast Input capability for the axis	BOOL
MLAxisCmdPos	Returns the reference position of the axis	None
MLAxisCreate	Creates a new axis object	None
MLAxisFBackPos	Returns the feedback position of the axis	None
MLAxisGenEN	Enables or disables the internal TMP generator of the axis	BOOL
MLAxisGenIsEN	Checks if the internal TMP generator of the axis is enabled	BOOL
MLAxisGenIsRdy	Checks if an axis is ready	BOOL
MLAxisGenPos	Returns the generator position of the axis	None
MLAxisGenReadAcc	Gets the acceleration of the internal generator of an axis	None
MLAxisGenReadDec	Gets the deceleration of the internal generator of an axis	None
MLAxisGenReadSpd	Gets the speed of the internal generator of an axis	None
MLAxisGenWriteAcc	Sets the acceleration of the internal generator of an axis	BOOL
MLAxisGenWriteDec	Sets the deceleration of the internal generator of an axis	BOOL

Name	Description	Return type
MLAxisGenWriteSpd	Sets the speed of the internal generator of an axis	BOOL
MLAxisInit	Initializes an axis object	BOOL
MLAxisIsCnctd	Checks if a pipe is currently connected to the axis	BOOL
MLAxisIsTriggered	Checks if the axis got a trigger event	BOOL
MLAxisMoveVel	Jogs at the specified speed	BOOL
MLAxisPipePos	Returns the pipe position of the axis	None
MLAxisPower	Powers up the axis. Enables Axis Servo Drive.	BOOL
MLAxisPowerDOff	Returns the adjustment of position done by the last power on to avoid bumps	None
MLAxisRatedTq	Sets rated motor torque	BOOL
MLAxisRead2ndFB	Read secondary feedback	None
MLAxisReadActPos	Returns the actual position of the axis	None
MLAxisReadFBUnit	Gets the feedback units per revolution value of the axis	None
MLAxisReadFEUU	Read following error in user units	None
MLAxisReadGenStatus	Returns the status of the internal generator of the axis	DINT
MLAxisReadModPos	Get the value period of the axis	None
MLAxisReadTq	Read actual torque	None
MLAxisReadUUnits	Get the user units per revolution value of the axis	None
MLAxisReadVel	Read actual velocity	None
MLAxisReAlignRdy	Checks if an axis is ready. Returns TRUE if the internal realignment axis is ready.	BOOL
MLAxisReAlign	Realigns the actual position with the reference position by moving the axis by the specified delta position	BOOL
MLAxisRel	Performs a relative move for a specified distance from the current position	BOOL
MLAxisResetErrors	Clears errors of the specified axis	BOOL
MLAxisRstFastIn	Resets the Fast Input	BOOL
MLAxisStatus	Returns the status of the axis	DINT
MLAxisStop	Stop with the specified deceleration	None
MLAxisTimeStamp	Returns the timestamp of the triggered axis	DINT
MLAxisWriteModPos	Sets the value period of the axis	BOOL
MLAxisWritePipPos	Forces the pipe position internal value. This function is working only when no pipe is connected.	BOOL
MLAxisWritePos	Sets the logical zero position of an axis	BOOL
MLAxisWriteUUnits	Sets the user units per revolution value of the axis	BOOL

1.1.6 Motion Library - Cam Profile

Name	Description	Return type
MLCamInit	Initializes a cam Pipe Block with user-defined settings	BOOL
MLCamSwitch	Switches profiles of the selected cam object	BOOL
MLPrfReadIOffset	Returns the Input Offset value of a selected cam profile	None
MLPrfReadIScale	Returns the Input Ratio value of a selected cam profile	None
MLPrfReadOOffset	Returns the Output Offset value of a selected cam profile	None
MLPrfReadOScale	Returns the Output Ratio value of a selected cam profile	None
MLPrfWriteIOffset	Sets the Input Offset value of a selected cam profile	BOOL
MLPrfWriteIScale	Sets the Input Ratio value of a selected cam profile	BOOL
MLPrfWriteOOffset	Sets the Output Offset value of a selected cam profile	BOOL
MLPrfWriteOScale	Sets the Output Ratio value of a selected cam profile	BOOL
MLProfileBuild	Builds a cam profile from application data	See Output
MLProfileCreate	Creates a new cam profile object	None
MLProfileInit	Initializes a previously created cam profile object	BOOL
MLProfileRelease	Removes a Profile so the Profile ID may be used by a different or new Profile.	See Output

1.1.7 Motion Library - Comparator


TIP

For usage example about Comparator Functions, see page 79

Name	Description	Return type
MLCompCheck	Checks if the reference of a comparator Pipe Block has been crossed. Returns TRUE if the reference has been crossed	BOOL
MLCompInit	Initializes a comparator Pipe Block with user-defined settings	BOOL
MLCompReadRef	Returns the reference position of a comparator block	None
MLCompReset	Clears the Transition Flag of a comparator Pipe Block	BOOL
MLCompWriteRef	Sets the reference position of a comparator block	BOOL

1.1.7.1 MLCompCheck

Description

Check if the reference of a comparator Pipe Block has been crossed. 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.

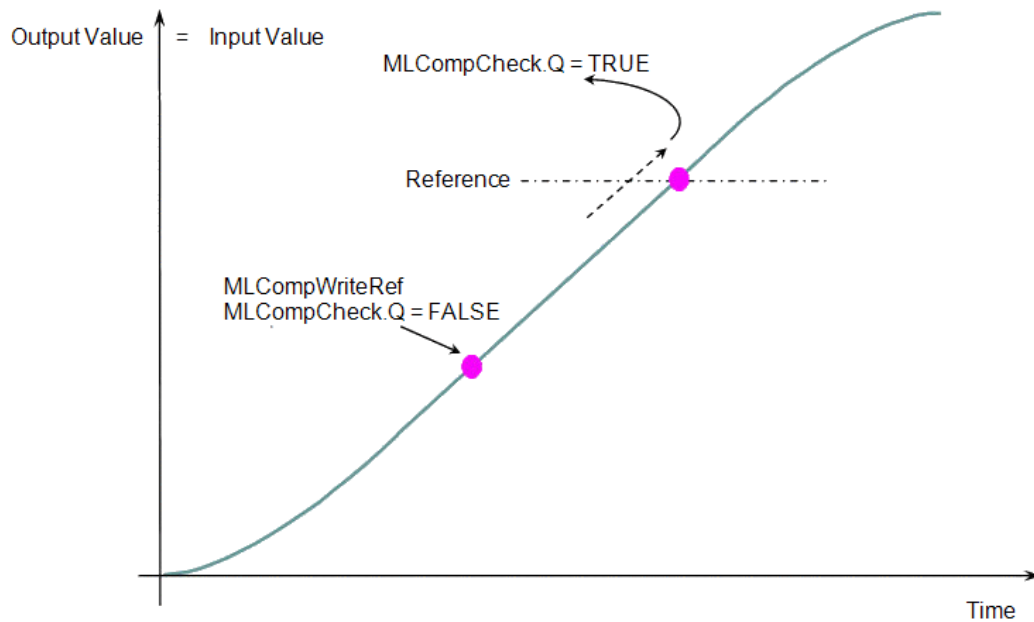


Figure 1-16: MLCompCheck

Arguments

Input

BlockID	Description	ID number of an initiated Comparator object
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Default (.Q)	Description	Returns TRUE if reference position of the Comparator object has been crossed
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

Related Functions

MLCompReset

MLCompWriteRef

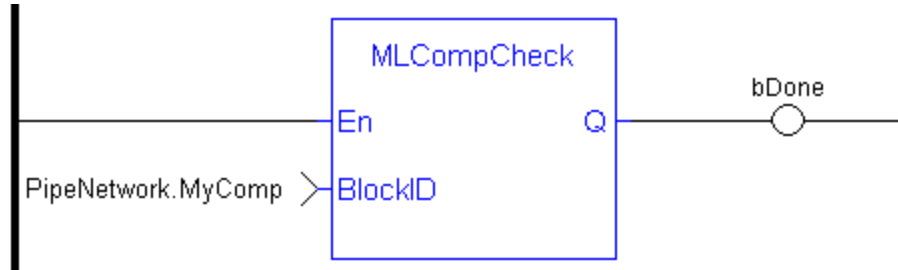
MLCompReadRef

Example

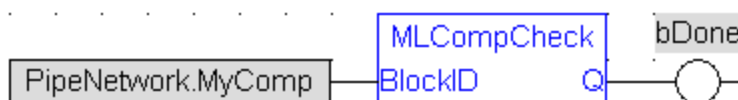
Structured Text

```
//Check if Comparator Reference has been reached
bCrossed := MLCompCheck( PipeNetwork.MyComp );
```

Ladder Diagram



Function Block Diagram



1.1.7.2 MLComplnit

Description

Initializes a comparator Pipe Block for use in a PLC Program. Function block is automatically called if a Comparator Block is added to the Pipe Network, with user-defined settings 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, 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 or equal to the Reference value.

NOTE Comparator objects are normally created in the Pipe Network using the graphical engine. Then you do not have to add MLComplnit function blocks to their programs. Parameters are entered directly in pop-up windows, and the code is then automatically added to the current project.

Arguments

Input

BlockID	Description	ID number of a created Comparator Pipe Block
	Data type	DINT
	Range	[-2147483648, 2147483648]

	Unit	n/a
	Default	—
ModuloPosition	Description	Value of the period of a cyclic system
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
ThroughZero	Description	When TRUE, system must cross zero and then the reference position before the Transition Flag is set. If FALSE, Transition Flag is set immediately if the input pipe position is greater then or equal to the Reference value.
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Reference	Description	Set the reference position in the new Comparator object
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
<u>Output</u>		
Default (.Q)	Description	Returns TRUE when function starts to execute
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

Related Functions

MLBlkCreate

MLCompCheck

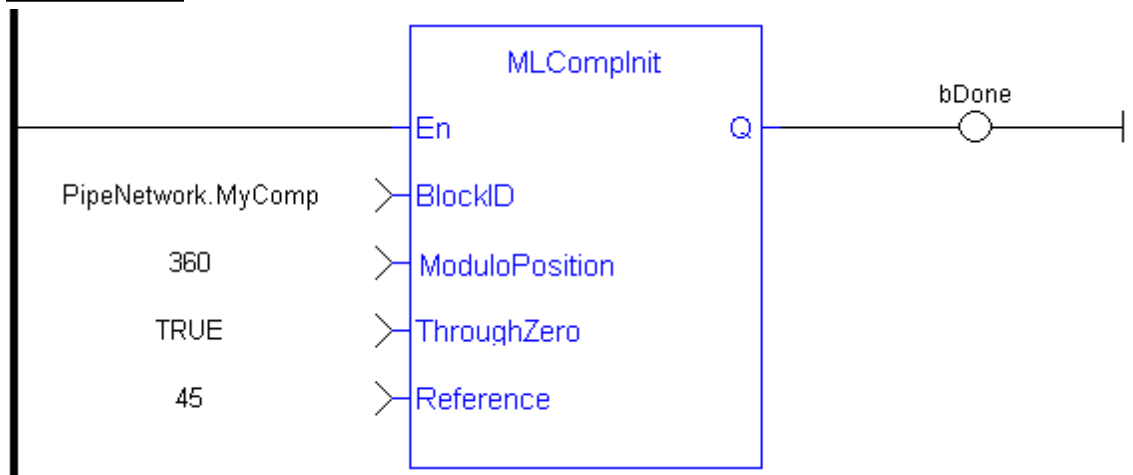
MLCompReset

MLCompWriteRef

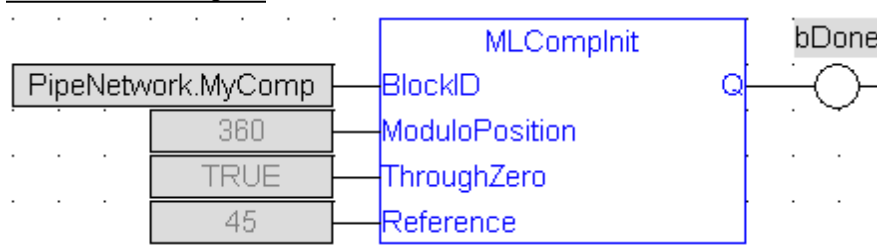
ExampleStructured Text

```
//Create and Initiate a Trigger object
MyComp := MLBlkCreate( 'MyComp', 'COMPARATOR' );
MLCompInit( MyComp, 360.0, TRUE, 45.0 );
```

Ladder Diagram



Function Block Diagram



1.1.7.3 MLCompReadRef

Description

Returns the reference position of a comparator block. 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.

Arguments

Input

BlockID	Description	ID number of an initiated Comparator object
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Reference	Description	Returns the current reference position of the Comparator object
	Data type	LREAL
	Unit	User unit

Related Functions

MLCompWriteRef

MLCompReset

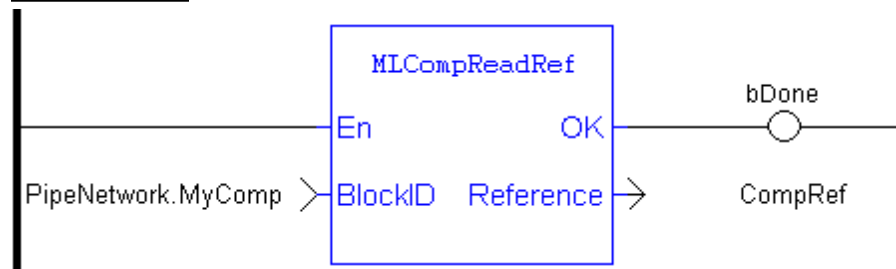
MLCompCheck

Example

Structured Text

```
//Return the Comparator Reference value
CompRef := MLCompReadRef( PipeNetwork.MyComp );
```

Ladder Diagram



Function Block Diagram



1.1.7.4 MLCompReset

Description

Clear the Transition Flag of a comparator Pipe Block. 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.

Arguments

Input

BlockID	Description	ID number of an initiated Comparator object
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Default (.Q)	Description	Returns TRUE when function starts to execute
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

Related Functions

MLCompCheck

MLCompReadRef

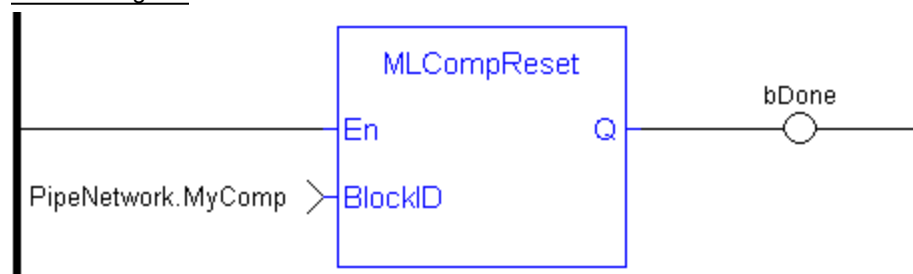
MLCompWriteRef

Example

Structured Text

```
//Clear the Transition Flag of a Comparator object
MLCompReset ( PipeNetwork.MyComp );
```

Ladder Diagram



Function Block Diagram



1.1.7.5 MLCompWriteRef

Description

Set the reference position of a comparator block. 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.

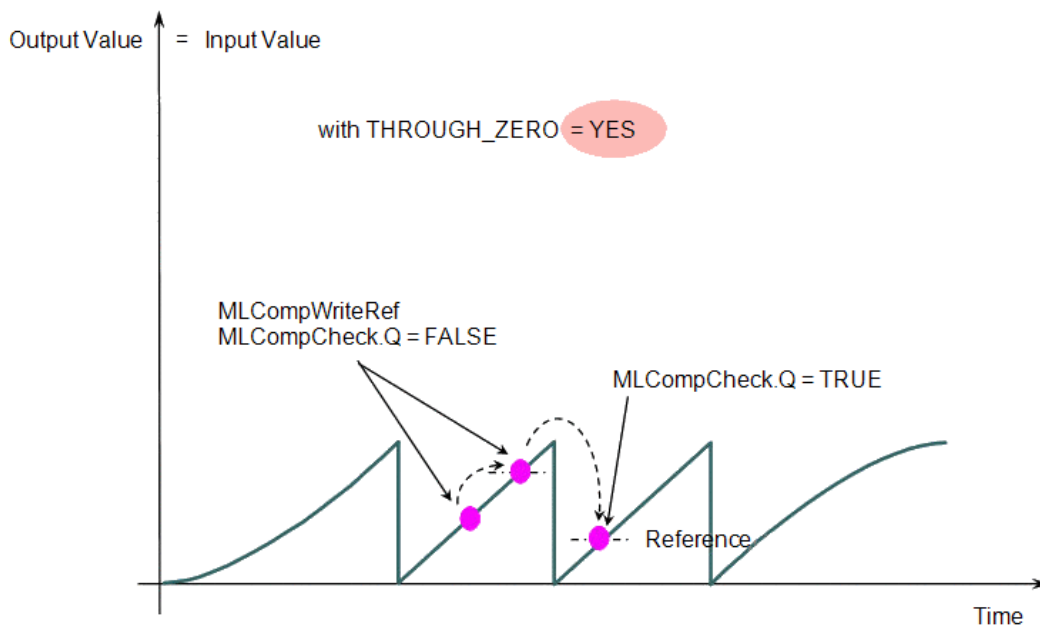


Figure 1-17: MLCompWriteRef

Arguments

Input

BlockID	Description	ID number of an initiated Comparator object
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
ThroughZero	Description	When TRUE, system must cross zero and then the reference position before the Transition Flag is set. If FALSE, Transition Flag is set immediately if the input pipe position is greater then or equal to the Reference value.
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Reference	Description	New reference position to set in the selected Comparator object
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

Default (.Q)	Description	Returns TRUE when function starts to execute
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

Related Functions

MLCompCheck

MLCompReadRef

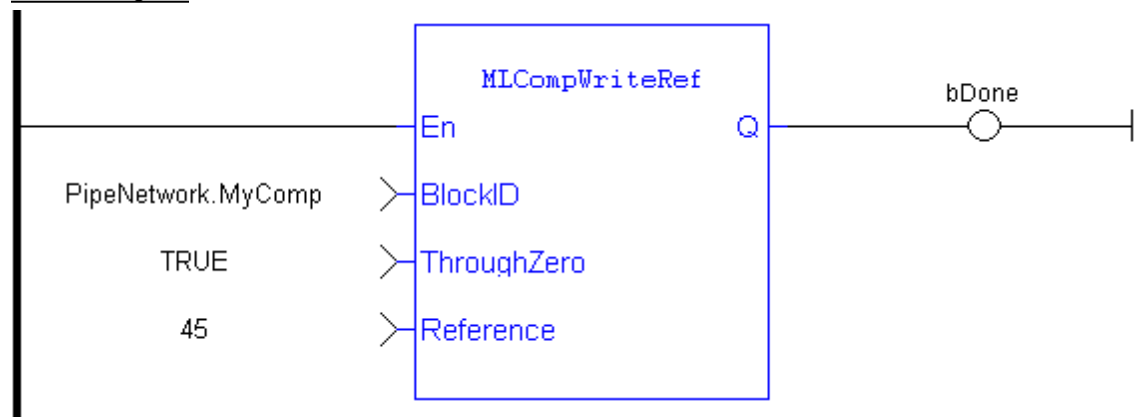
MLCompReset

Example

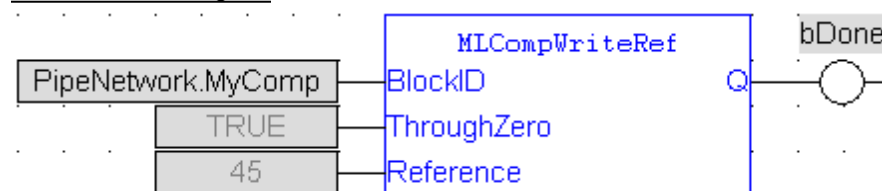
Structured Text

```
//Set the Comparator Reference value
MLCompWriteRef( PipeNetwork.MyComp , TRUE , 45 );
```

Ladder Diagram

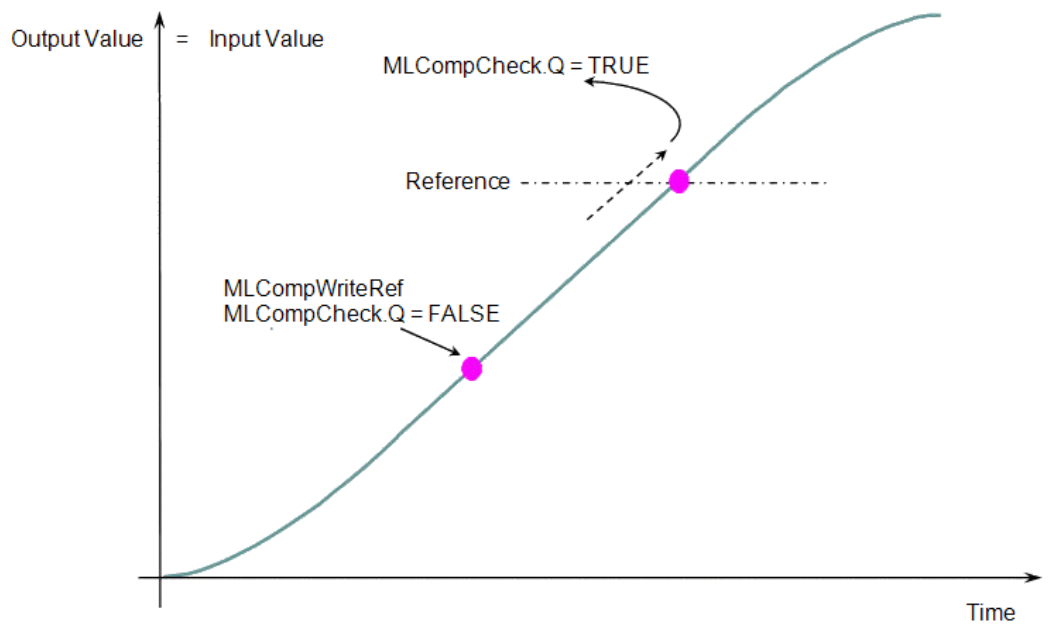
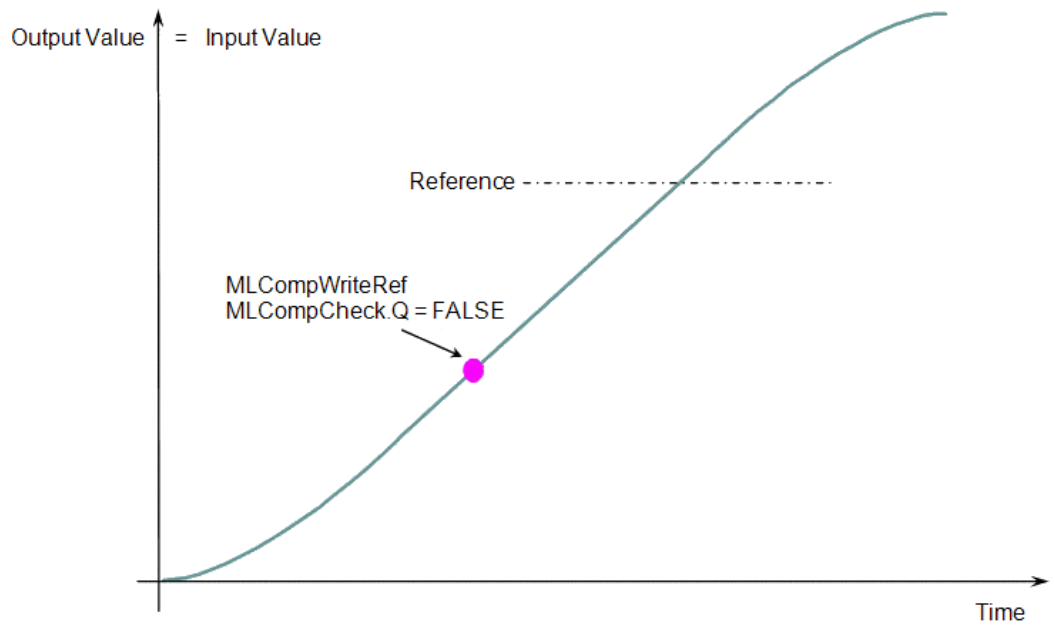


Function Block Diagram

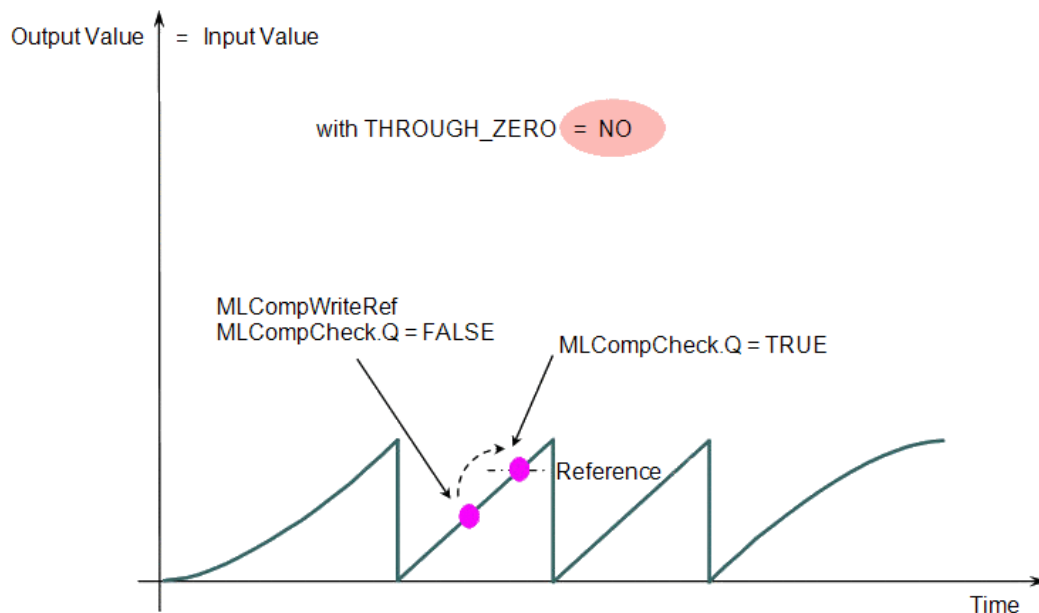
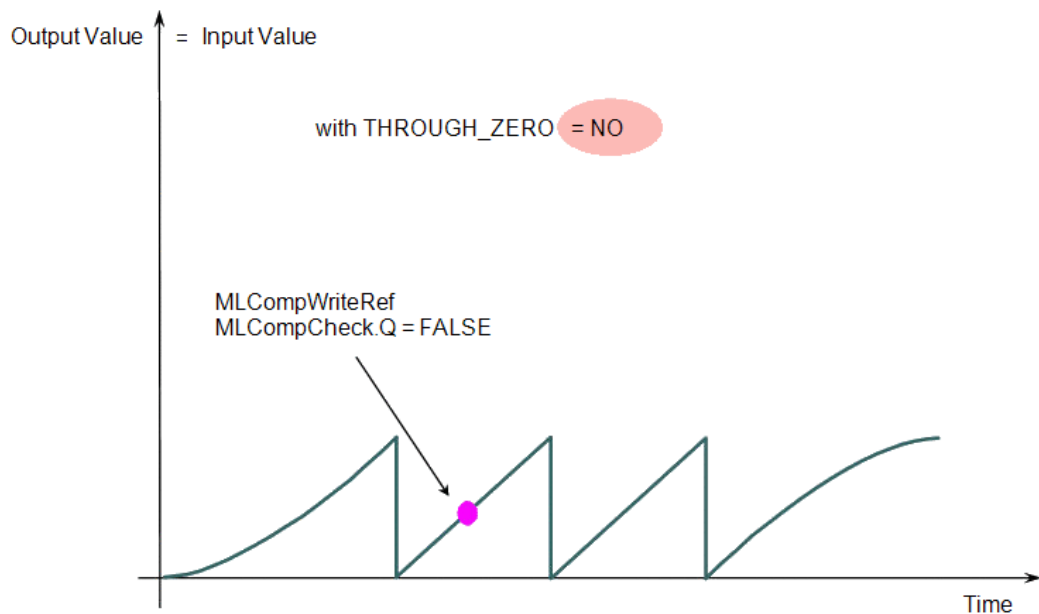


1.1.7.6 Usage example of Comparator Functions

When you call the **MLCompWriteRef** function, the output for MLCompCheck 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 becomes True as soon as the input value reaches the reference, but not before it has passed through zero.

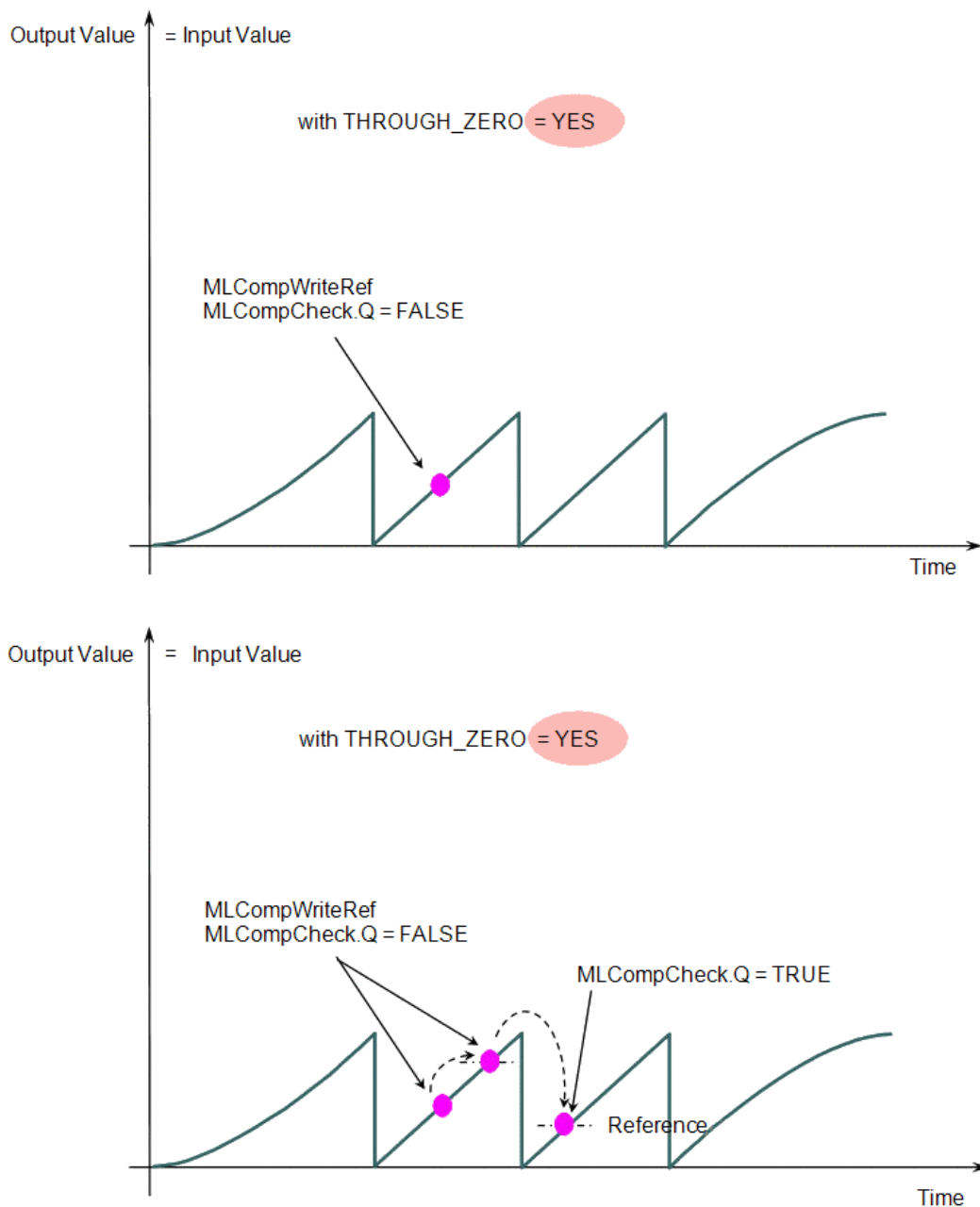


Figure 1-18: Comparator Functions Usage

1.1.8 Motion Library - Convertor

Name	Description	Return type
MLCNVConnect	Connects a converter Pipe Block to the specified axis	BOOL
MLCNVConnectEx	Connects an extra converter Pipe Block to the specified axis. This function connects the output of a pipe to an axis data other than the control position.	BOOL
MLCNVDisconnect	Disconnects a converter Pipe Block from its associated axis	BOOL

Name	Description	Return type
MLCNVInit	Initializes a converter Pipe Block in Position or Speed mode	BOOL

1.1.8.1 MLCNVConnect

Description

Connect a converter Pipe Block to the specified axis. 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, as seen in Figure 1 below.

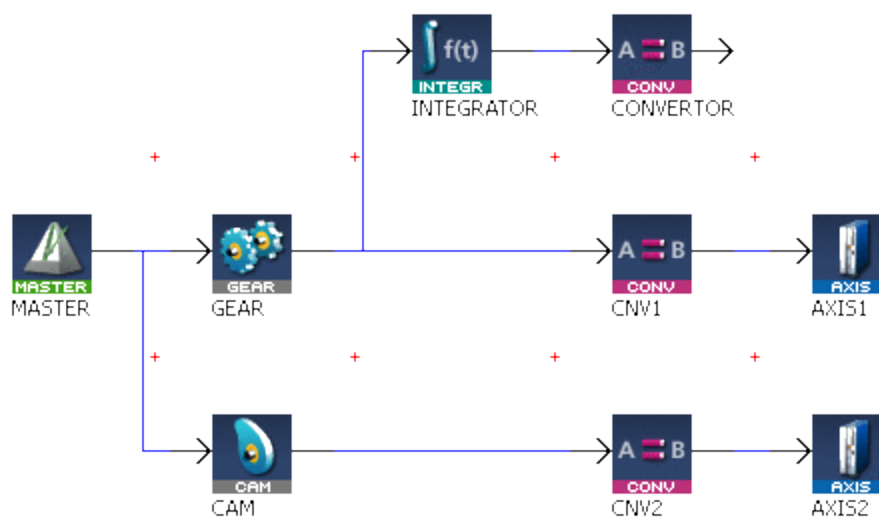


Figure 1-19: MLCNVConnect

NOTE All converters in the Pipe Network can be connected at once with the command PipeNetwork(MLPN_Connect). This calls automatically generated code with MLCNVConnect commands for each Converter block. Therefore, 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. See for information on the parameters.

Arguments

Input

BlockID	Description	ID number of an initiated Converter object
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
AxisID	Description	ID number of an initiated Axis object
	Data type	DINT

Range	[-2147483648, 2147483648]
Unit	n/a
Default	—

Output

Default (.Q)

Description	Returns TRUE if the converter is connected to the Axis object
Data type	BOOL
Unit	n/a

Return Type

BOOL

Related Functions

MLCNVConnectEx

MLCNVDisconnect

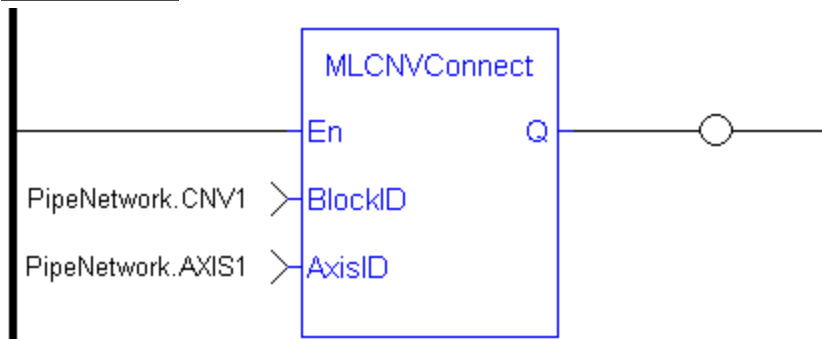
MLCNVInit

Example

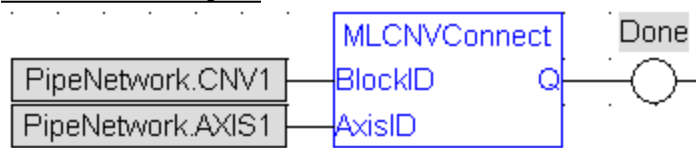
Structured Text

```
//Connect a converter Pipe Block to Axis1
MLCNVConnect( CNV1, AXIS1 );
```

Ladder Diagram



Function Block Diagram



1.1.8.2 MLCNVConnectEx

Description

Connect a converter Pipe Block to the specified axis. This function connects the output of a pipe to an axis data other than the control position. With this function,

several converter Pipe Blocks can connect 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 such as torque feed-forward (IDN 3056) that needs to be provided by a second converter.

NOTE This FB does not work when you choose to simulate the device. In such a case, the FB continuously generates error messages displayed in the Controller log window.

NOTE Need to add 16#8000 to desired IDN number for ValueID input. 8000 in hexadecimal signals a vendor-specific IDN value.

Arguments

Input

BlockID	Description ID number of an initiated Converter object Data type DINT Range [-2147483648, 2147483648] Unit n/a Default —
AxisID	Description ID number of an initiated Axis object Data type DINT Range [-2147483648, 2147483648] Unit n/a Default —

ValueID Description (for EtherCAT motion bus)

EtherCAT:

Specify the following constant:

- EC_ADDITIVE_TORQUE_VALUE
(for torque feed-forward)
- EC_ANALOG_OUTPUT
(for control of Analog Output: AKD
parameter: "AOUT.VALUEU")

NOTE If the Analog Output is mapped to a PLC variable, the connection to the analog output by EC_ANALOG_OUTPUT will not work as the output value will be overwritten by the PLC mapped variable data. In order to function properly the AOUT.MODE must be set to "User Mode (mode = 0)".

TIP The PDO values will be overwritten by Mapped PLC variables including a possible link to the mapping of variables or the section on MLParamWrite() warning indicating that the function block write of Analog output will be overwritten by the MLCnvConnectEx function.

Precedence rules:

1. A PLC variable mapped to Analog Output takes precedence.
2. If MLCNVConnect assigns a Pipe output to Analog Output it will take precedence over a DriveParamWrite function call.
3. DriveParamWrite will modify the Analog Output but get overwritten by the higher precedent options if they are present.

Data type DINT
 Range [-2147483648, 2147483648]
 Unit n/a
 Default —

ValueInfo	Description	EtherCAT: This value is ignored and must be set to zero
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Default (.Q)	Description	Returns TRUE if the converter is connected to the Axis object
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

Related Functions

MLCNVConnect

MLCNVDisconnect

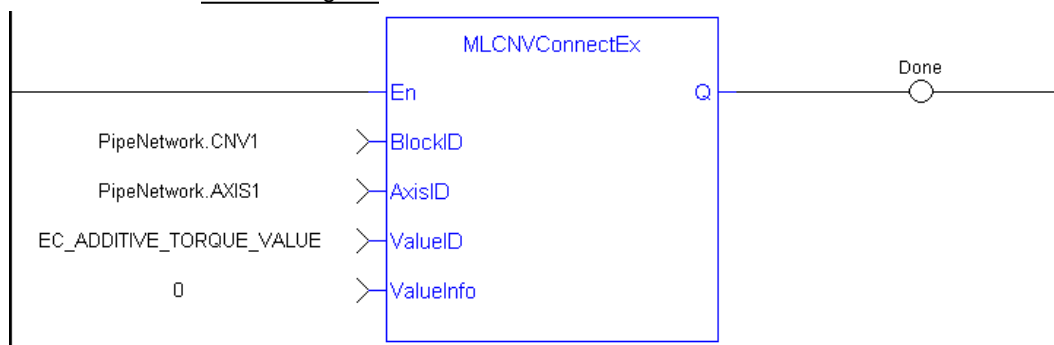
MLCNVInit

Example

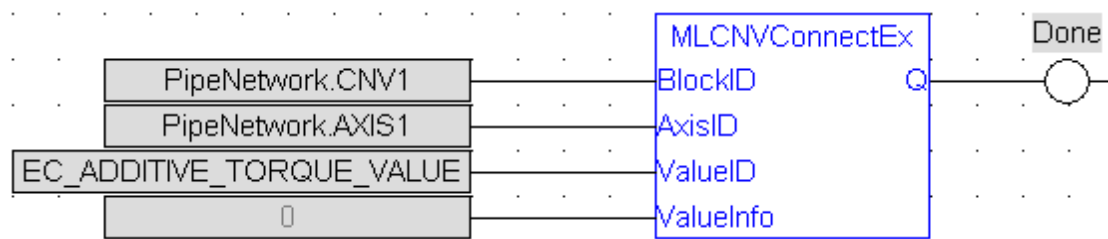
Structured Text

```
//Connect a converter Pipe Block to Axis1 to send feed-forward
MLCNVConnectEx( PipeNetwork.CNV1, PipeNetwork.AXIS1, EC_ADDITIVE_TORQUE_VALUE, 0 );
```

Ladder Diagram



Function Block Diagram



1.1.8.3 MLCNVDisconnect

Description

Disconnect a converter Pipe Block from its associated axis.

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.

Arguments

Input

<code>BlockID</code>	Description	ID number of an initiated Converter object
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

<code>Default (.Q)</code>	Description	Returns TRUE if the converter is disconnected from the Axis object
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

Related Functions

MLCNVConnect

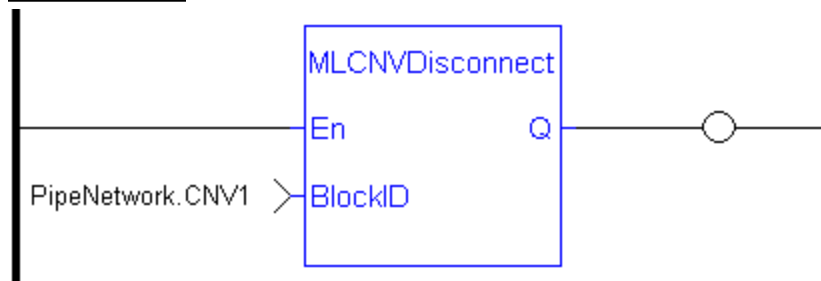
MLCNVInit

Example

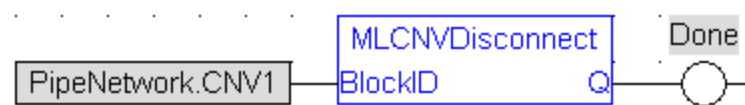
Structured Text

```
//Disconnect a converter Pipe Block from its axis
MLCNVDisconnect( CNV1);
```

Ladder Diagram



Function Block Diagram



1.1.8.4 MLCNVInit

Description

Initializes a converter Pipe Block. Function block is automatically called if a Converter 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. Then 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.

Arguments

Input

<code>BlockID</code>	Description	ID number of a created Pipe Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
<code>Mode</code>	Description	1 for Position mode, 2 for Speed mode. Determines the type of input to the Converter Object.
	Data type	DINT
	Range	[1, 2]
	Unit	n/a
	Default	—

Output

<code>Default (.Q)</code>	Description	Returns TRUE if the Converter Pipe Block is initialized
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

Related Functions

MLBlkCreate

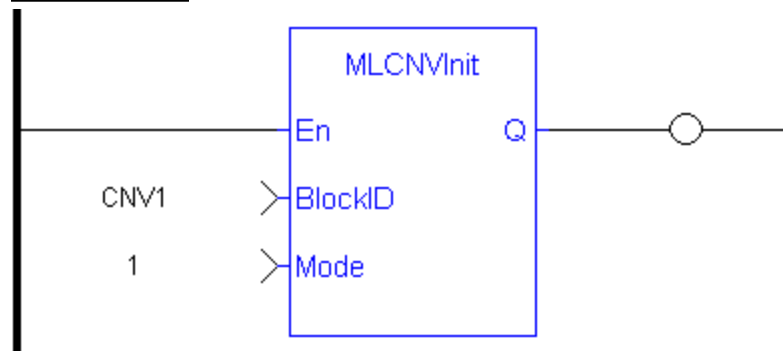
MLCNVConnect

Example

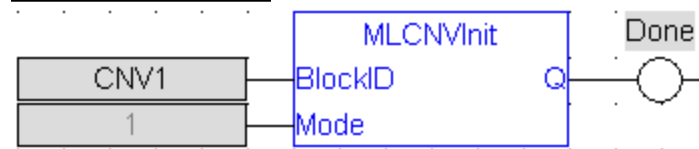
Structured Text

```
//Create and Initiate a Converter object
CNV1 := MLBlkCreate( 'CNV1', 'CONVERTOR' );
MLCNVInit( CNV1, 1 );
```

Ladder Diagram



Function Block Diagram



1.1.9 Motion Library - Delay

Name	Description	Return type
MLDelayInit	Initializes a delay object	BOOL

1.1.9.1 MLDelayInit

Description

Initializes a delay object. Returns TRUE if the function succeeded. This FB 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.

Arguments

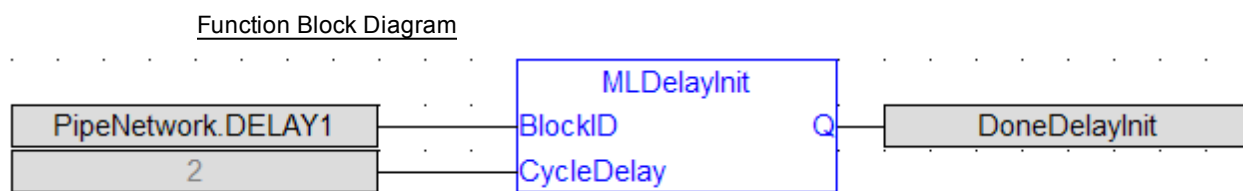
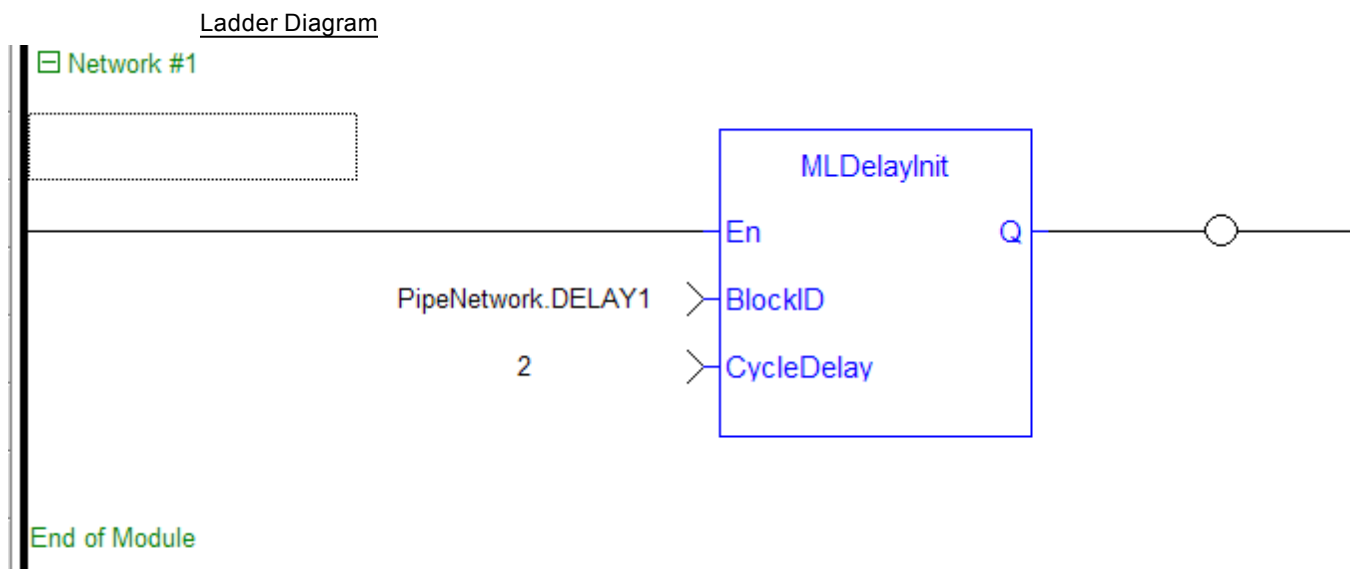
Input

BlockID	Description	ID number of a created Pipe Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
CycleDelay	Description	Number of delay cycles
	Data type	DINT
	Range	[0 , 9]
	Unit	Cycle
	Default	0

Example

Structured Text

```
MLDelayInit (PipeNetwork.DELAY1, 2 );
```



1.1.10 Motion Library - Derivator

Name	Description	Return type
MLDerInit	Initializes a derivator object	BOOL
MLDerReadInModPos	Returns the input MODULO_POSITION of the Derivator block	None
MLDerWriteInModPos	Sets the input MODULO_POSITION of the Derivator block	BOOL

1.1.10.1 MLDerInit

Description

Initializes an derivator object. 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. Then you do not have to add MLDerInit function blocks to their programs. Parameters are entered directly in pop-up windows, and the code is then automatically added to the current project.

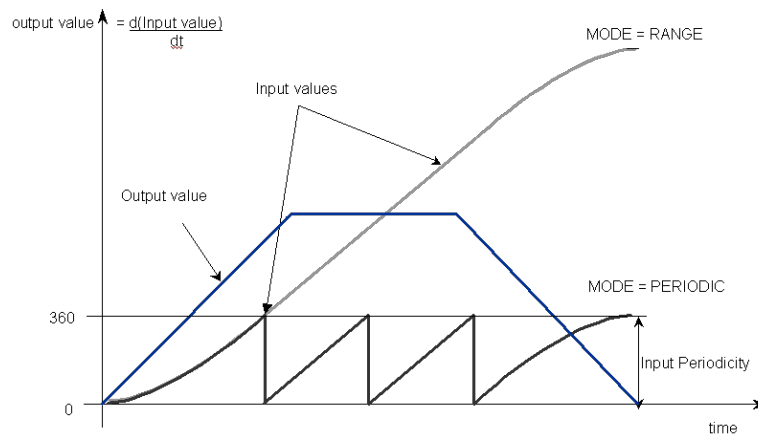


Figure 1-20: MLDerInit

Arguments

Input

BlockID	Description Data type Range Unit Default	ID number of a created Pipe Block DINT [-2147483648, 2147483648] n/a —
ModuloPosition	Description Data type Range Unit Default	Input ModuloPosition of Derivator object LREAL — User unit 360.0

Output

Default (.Q)	Description Data type Unit	Returns TRUE if the Derivator object is initialized BOOL n/a
---------------------	----------------------------------	--

Return Type

BOOL

Related Functions

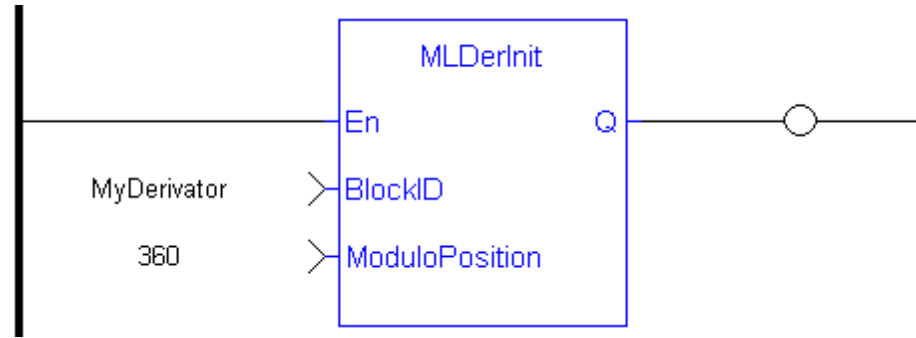
- MLBlkCreate
- MLDerReadInModPos
- MLDerWriteInModPos

Example

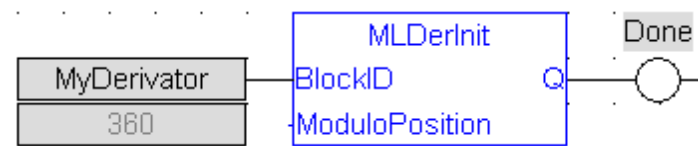
Structured Text

```
//Create and Initiate a Derivator object
MyDerivator := MLBlkCreate( 'MyDerivator', 'DERIVATOR' );
MLDerInit( MyDerivator, 360.0 );
```

Ladder Diagram



Function Block Diagram



1.1.10.2 MLDerReadInModPos

Description

Returns the Input ModuloPosition of the derivator block. Input ModuloPosition is defined to manage the periodicity (modulo) of the input values.

For example, if the input value increases each millisecond by one degree then the output value is 1000 degrees per second. Now lets imagine that the input value skips suddenly from 359 to 0

- If Input ModuloPosition = 360, the output continues to indicate 1000 degrees per second, indicating that rollover into the next period has been properly handled
- If Input ModuloPosition = 1000, the output then indicates 359,000 degrees per second, indicating that the input has incorrectly interpreted roll-over as a 359 degree move in one millisecond

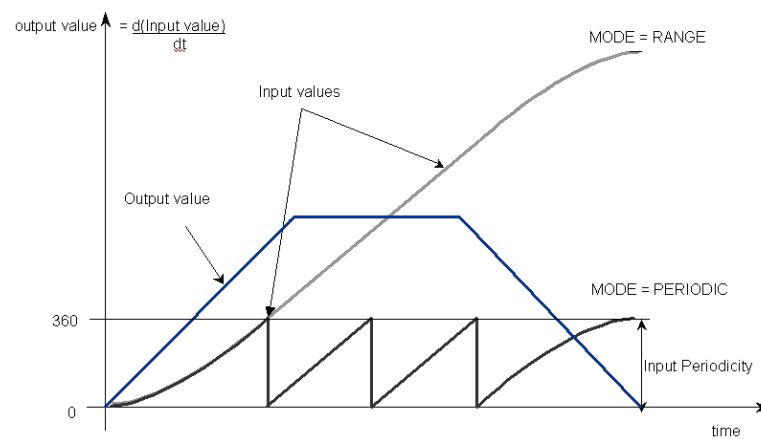


Figure 1-21: MLDerReadInModPos

NOTE

The first calculation of a Derivator Pipe Block just after the pipe installation indicates zero regardless of the initial input value.

Arguments

Input

ID	Description	ID number of an initiated Derivator object
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

ModuloPosition	Description	Current Input ModuloPosition of the selected Derivator object
	Data type	LREAL
	Unit	User unit
	Default	—

Related Functions

MLDerWriteInModPos

MLDerInit

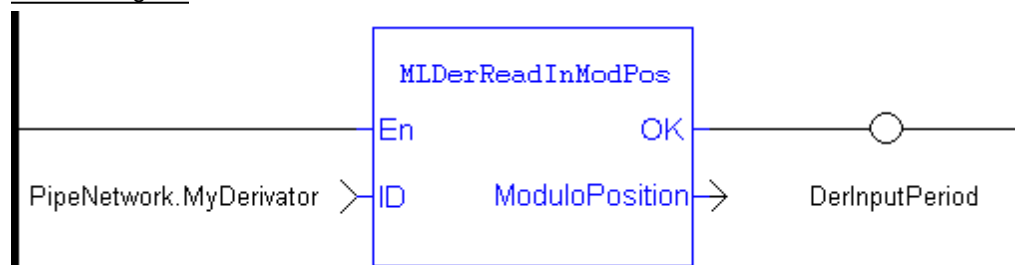
Example

Structured Text

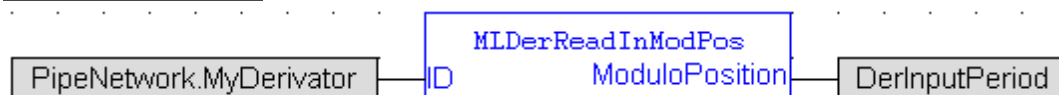
```
//save the current input MODULO_POSITION of a Derivator
object

DerInputPeriod := MLDerReadInModPos (
PipeNetwork.MyDerivator );
```

Ladder Diagram



Function Block Diagram



1.1.10.3 MLDerWriteInModPos

Description

Sets the Input ModuloPosition of the Derivator block. Input ModuloPosition is defined to manage the periodicity (modulo) of the input values.

For example, if the input value increases each millisecond by one degree then the output value is 1000 degrees per second. Now lets imagine that the input value skips suddenly from 359 to 0

-If Input ModuloPosition = 360, the output continues to indicate 1000 degrees per second, indicating that rollover into the next period has been properly handled

-If Input ModuloPosition = 1000, the output then indicates 359,000 degrees per second, indicating that the input has incorrectly interpreted roll-over as a 359 degree move in one millisecond

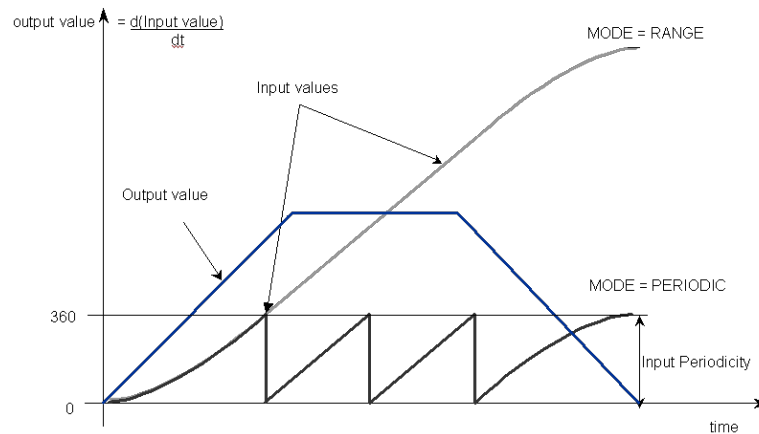


Figure 1-22: MLDerWriteInModPos

NOTE The first calculation of a Derivator Pipe Block just after the pipe installation indicates zero regardless of the initial input value.

Arguments

Input

ID	Description	ID number of an initiated Derivator object
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
ModuloPosition	Description	Desired new value of Input ModuloPosition of the selected Derivator object
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

Default (.Q)	Description	Returns TRUE if the Input ModuloPosition value is changed
--------------	-------------	---

Data type BOOL
 Unit n/a

Return Type
 BOOL

Related Functions

MLDerReadInModPos

MLDerInit

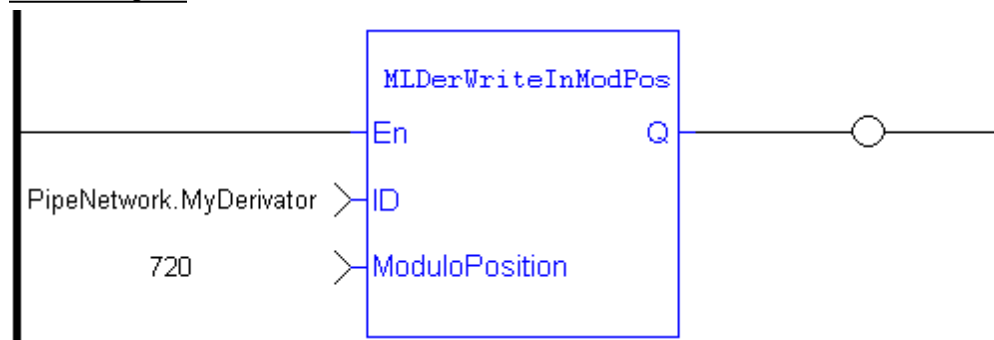
Example

Structured Text

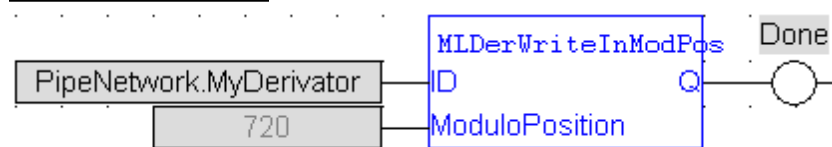
```
//change the input MODULO_POSITION of a Derivator object
to 720

MLDerWriteInModPos ( PipeNetwork.MyDerivator, 720 );
```

Ladder Diagram



Function Block Diagram



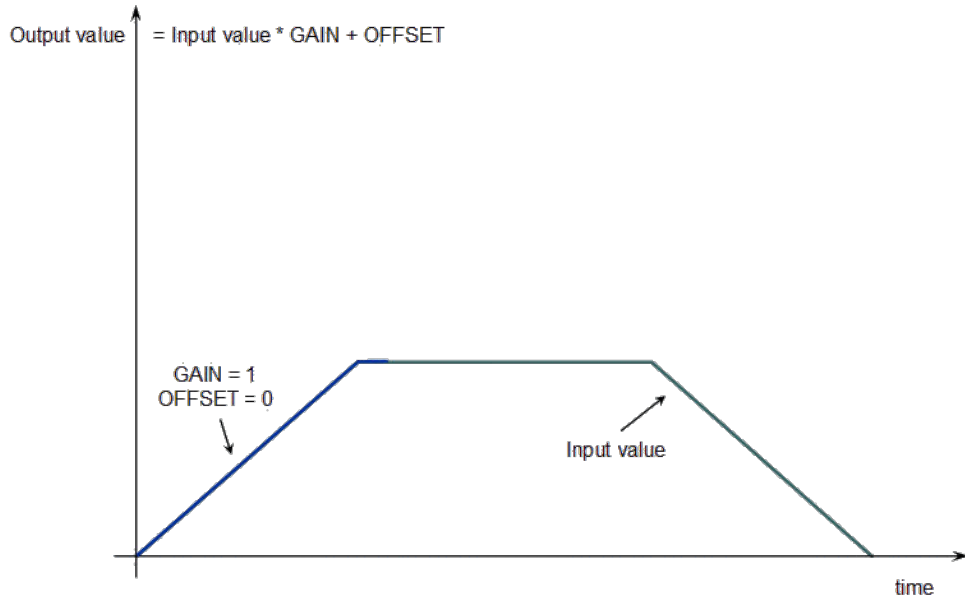
1.1.11 Motion Library - Gear

Name	Description	Return type
MLGearInit	Initializes a Gear Pipe Block with user-defined settings	BOOL
MLGearReadOffset	Returns the offset value of selected Gear Block	None
MLGearReadOffSlp	Returns the Offset Slope value of selected Gear Block	None
MLGearReadRatio	Returns the ratio value of a gear block	None
MLGearReadRatSlp	Returns the ratio slope value of a gear block	None
MLGearWriteOff	Sets the Offset value of a selected Gear Pipe Block	BOOL

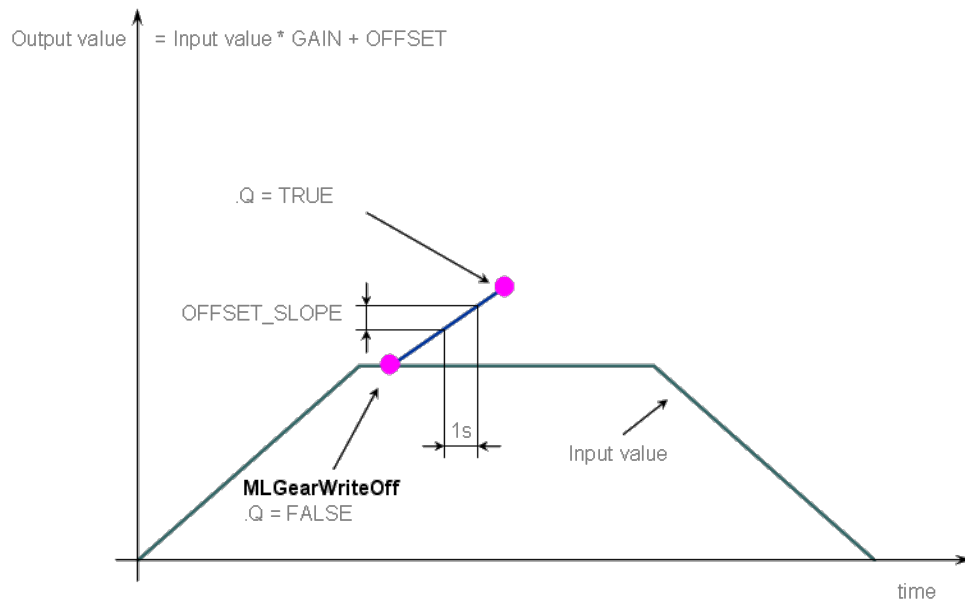
Name	Description	Return type
MLGearWriteOSlp	Sets the Offset Slope value of a selected Gear Pipe Block	BOOL
MLGearWriteRatio	Sets the Ratio value of a selected Gear Pipe Block	BOOL
MLGearWriteRatSlp	Sets the Ratio Slope value of a selected Gear Pipe Block	BOOL

1.1.11.1 Usage example of Gear Functions

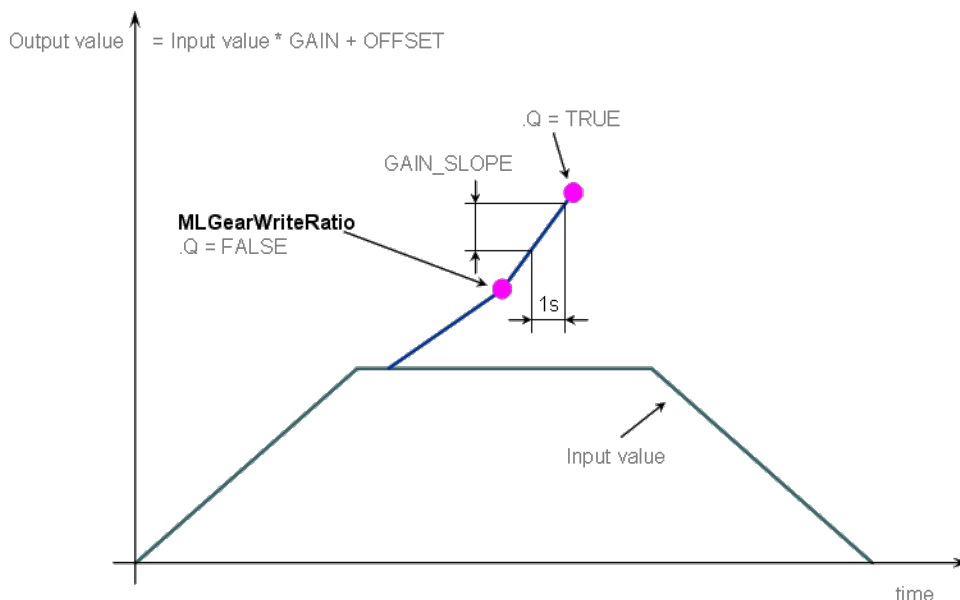
The output value starts with offset = 0 and gain = 1 (blue line)



You can call the **MLGearWriteOff** function to modify the Offset (where Offset_Slope is set with the **MLGearWriteOSlp** function).



After setting the Offset (Q=TRUE in the previous figure), you can call the **MLGearWriteRatio** function to modify the gear Ratio (where Gain_Slope is set with the **MLGearWriteRatSlp** function).



The output value is finally adapted with the gear offset and ratio (blue line).

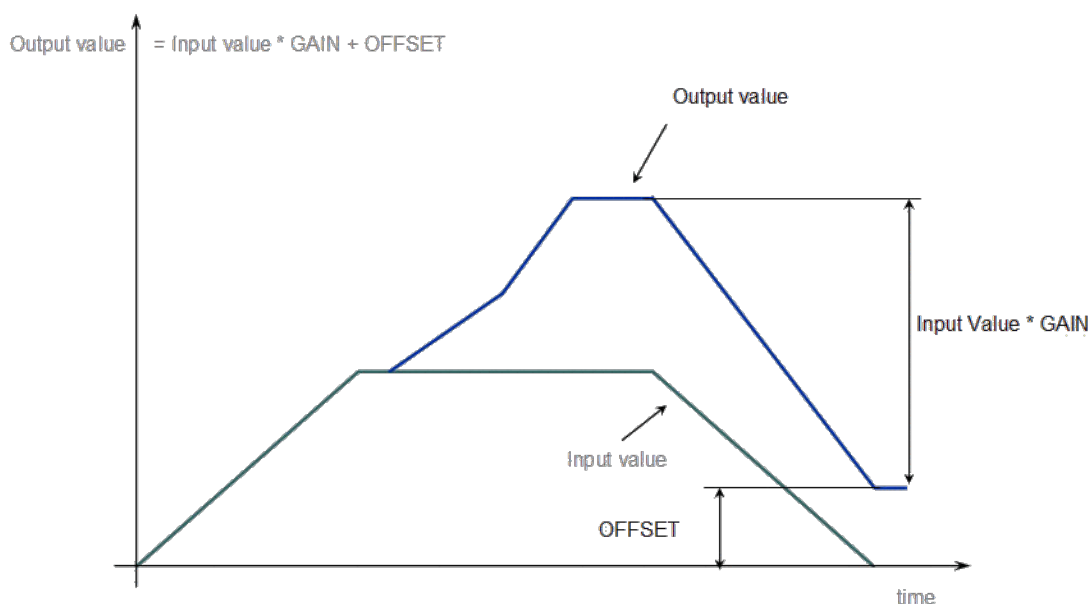


Figure 1-23: Gear Functions Usage

1.1.12 Motion Library - Integrator

Name	Description	Return type
MLIntInit	Initializes an integrator object	BOOL
MLIntWriteOutVal	Sets the output value of an integrator object	BOOL

1.1.12.1 MLIntInit

Description

Initializes an integrator object. Function block is automatically called if an Integrator Block is added to the Pipe Network, with user-defined settings entered in the Pipe Blocks Properties screen.

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.

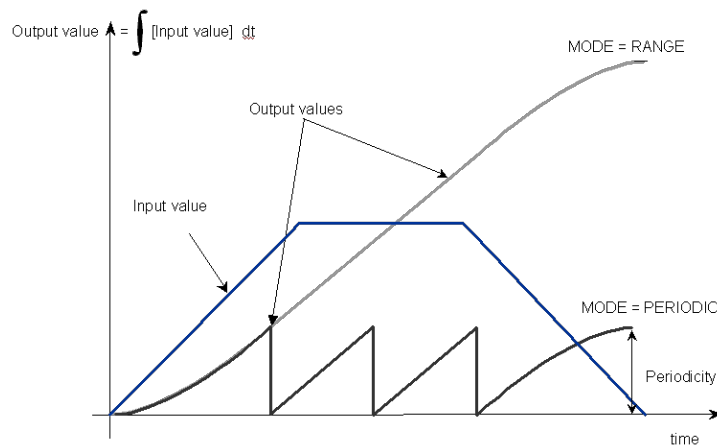


Figure 1-24: MLIntInit

NOTE

Integrator objects are normally created in the Pipe Network using the graphical engine. Then you do not have to add MLIntInit function blocks to their programs. Parameters are entered directly in pop-up windows, and the code is then automatically added to the current project.

Arguments

Input

BlockID	Description	ID number of a created Pipe Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
ModuloPosition	Description	Output ModuloPosition of Integrator object
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	360.0
Modulo	Description	TRUE when mode is modulo. Modulo mode adapts the output values according to the ModuloPosition (modulo)
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	TRUE

Output

Default (.Q)

Description	Returns TRUE if the Integrator object is initialized
Data type	BOOL
Unit	n/a

Return Type

BOOL

Related Functions

MLBlkCreate

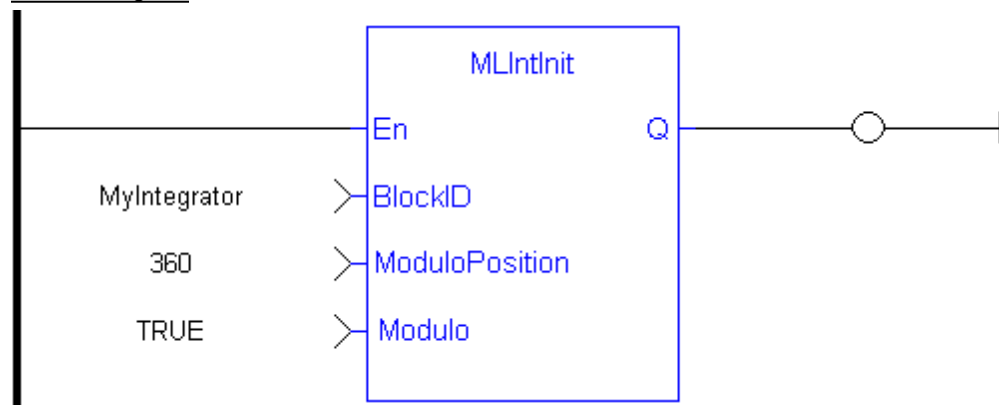
MLIntWriteOutVal

Example

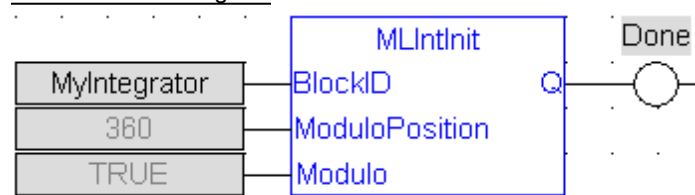
Structured Text

```
//Create and Initiate an Integrator object
MyIntegrator := MLBlkCreate( 'MyIntegrator', 'INTEGRATOR'
);
MLIntInit(MyIntegrator, 360.0, true );
```

Ladder Diagram



Function Block Diagram



1.1.12.2 MLIntWriteOutVal

Description

Sets the output value of an integrator object. This function can force the output to an entered value not dependent on the input value from the Pipe Network.

NOTE Output value can jump to another value instantly after the function is executed if the Pipe Network is running.

Arguments

Input

<u>BlockID</u>	Description	ID number of an initiated Integrator object
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

<u>Value</u>	Description	Desired new output value of the selected Integrator object
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

<u>Default (.Q)</u>	Description	Returns TRUE if the output value if the Integrator object is changed
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

Related Functions

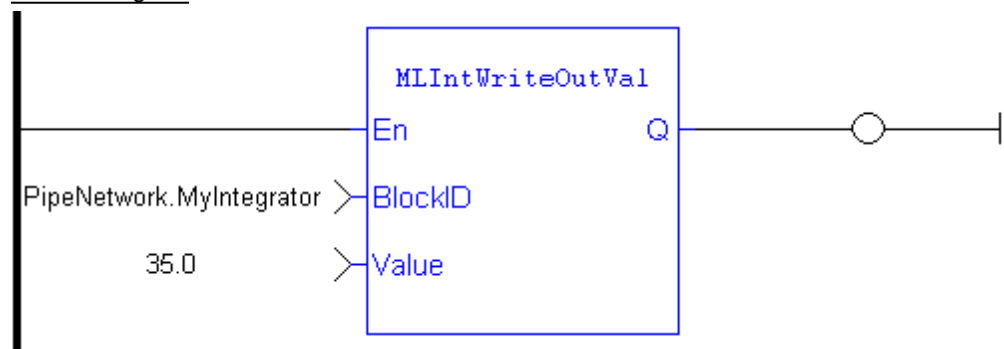
MLIntInit

Example

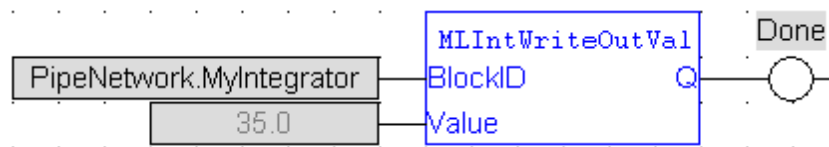
Structured Text

```
//change the output value of an integrator object to 35
MLIntWriteOutVal ( PipeNetwork.MyIntegrator, 35.0 );
```

Ladder Diagram



Function Block Diagram



1.1.13 Motion Library - Master



For usage example about Master Functions, see page 124

Function sorted by types:

Motion Control	Inquiry Functions	Position setting
MLMstInit	MLMstReadAccel	MLMstAbs
MLMstRun	MLMstReadDecel	MLMstAdd
MLMstWriteAccel	MLMstReadInitPos	MLMstForcePos
MLMstWriteDecel	MLMstReadSpeed	MLMstRel
MLMstWriteSpeed	MLMstStatus	

Functions sorted in alphabetical order:

Name	Description	Return type
MLMstAbs	Does an absolute move	BOOL
MLMstAdd	Does an additive move relative for a specified distance from the endpoint of the previous move	BOOL
MLMstForcePos	Forces the specified position. Possible only when the block is not moving.	BOOL
MLMstInit	Initializes a master object (TMP generator)	BOOL
MLMstReadAccel	Gets the present acceleration value of a master block	None
MLMstReadDecel	Gets the present deceleration value of a master block	None
MLMstReadInitPos	Gets the initial position of a master block	None
MLMstReadSpeed	Gets the speed of a master block	None
MLMstRel	Does an Relative move for a specified distance from the current position	BOOL
MLMstRun	Jogs at the specified speed. Returns TRUE if the function succeeded	BOOL
MLMstStatus	Returns the status of the generator	DINT
MLMstWriteAccel	Sets the acceleration of a master block	BOOL
MLMstWriteDecel	Sets the deceleration of a master block	BOOL
MLMstWriteInitPos	Sets the initial position of a master block	BOOL
MLMstWriteSpeed	Sets the speed of a master block	BOOL

1.1.13.1 MLMstAbs

Description

Performs a move to an absolute position. Returns TRUE if the function succeeded.

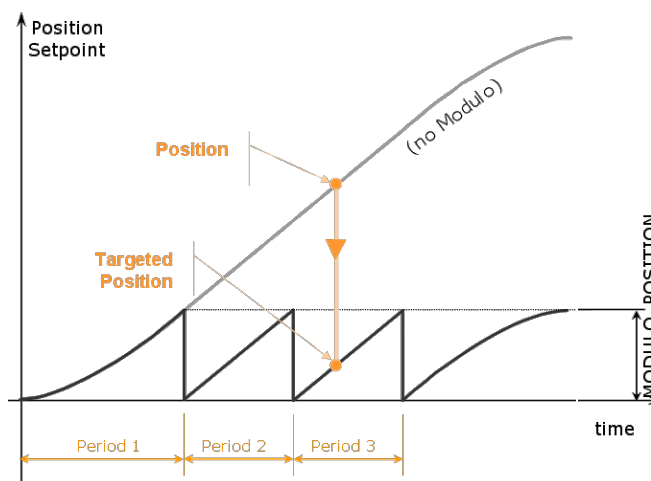
Arguments

Input

BlockID	Description	ID name of the Master Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Position

Description	<p>Sets the value of the absolute destination position.</p> <p>When the Modulo is turned on, the Master Block moves to the targeted position during the corresponding period, calculated as follows:</p> <ul style="list-style-type: none"> • If the Position input is between 0 and the Modulo Position, then the Master Block moves within the current period (no position rollover). • If the Position input is greater than the Modulo Position, then the Master Block moves during one of the next period (positive position rollover).
-------------	--



The Master Block works similarly for negative positions: if the Position input is less than zero, then the Master Block moves during one of the **previous** period (negative position rollover).

Data type	LREAL
Range	—
Unit	User unit
Default	—

Output

Default (.Q)	Description	Returns true when function successfully executes
--------------	-------------	--

Data type BOOL
 Unit n/a

Related Functions

MLMstWriteSpeed

MLMstWriteDecel

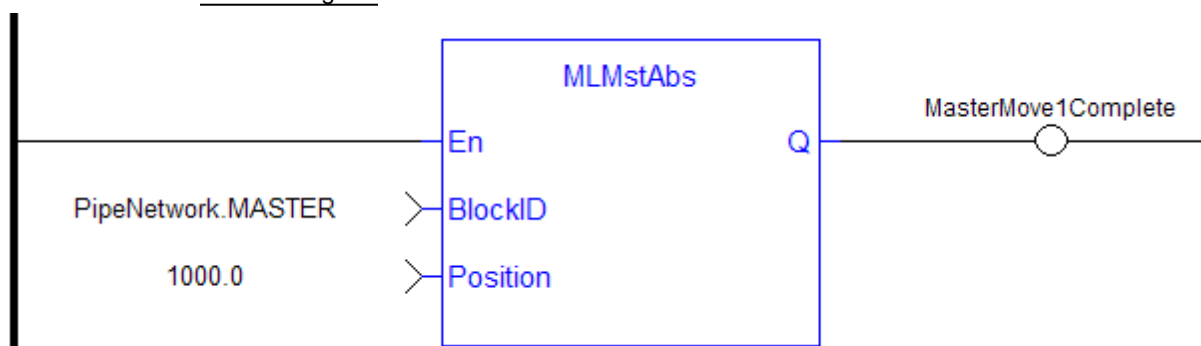
MLMstWriteSpeed

Example

Structured Text

```
MLMstAbs( PipeNetwork.MASTER, 1000.0 );
```

Ladder Diagram



Function Block Diagram



1.1.13.2 MLMstAdd

Description

Performs a move for a specified distance relative to the endpoint of the previous move. Returns TRUE if the function succeeded.

Arguments

Input

EN	Description	Enables FB to be executed
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Block ID	Description	ID name of the Master Block

Data type	DINT
Range	[-2147483648, 2147483648]
Unit	n/a
Default	—

DeltaPos	Description	Relative distance to move
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

<u>Output</u>	Description	Returns true when function successfully executes
Default (.Q)	Data type	BOOL
	Unit	n/a

Related Functions

MLMstWriteSpeed

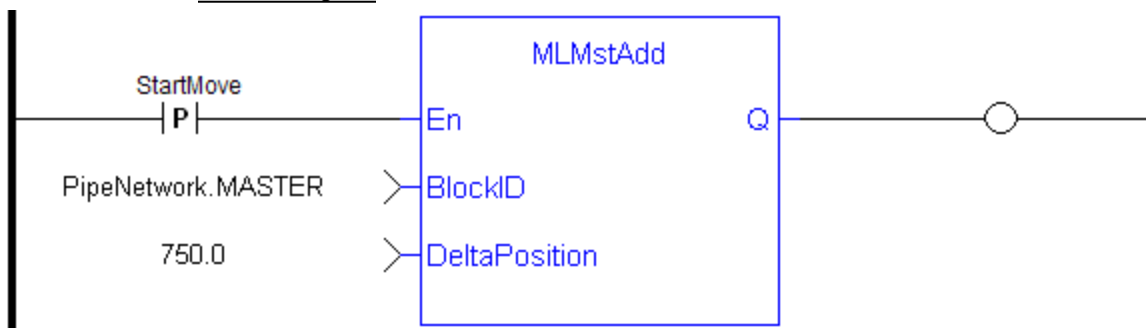
MLMstWriteDecel

Example

Structured Text

```
MLMstAdd( PipeNetwork.MASTER, 750.0 );
```

Ladder Diagram



NOTE You must use a pulse contact to start the FB

Function Block Diagram



1.1.13.3 MLMstForcePos

Description

Forces the position of a Master Block to a specified position. 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.

Arguments

Input

EN	Description	Enables FB to be executed
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Block ID	Description	ID name of the Master Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Position	Description	Defines the Master starting position when the motion starts
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

Default (.Q)	Description	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a

Related Functions

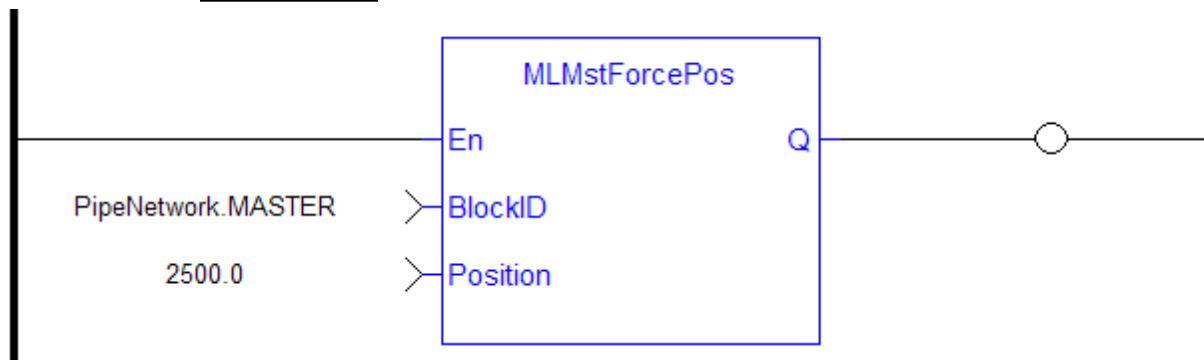
MLMstReadInitPos

Example

Structured Text

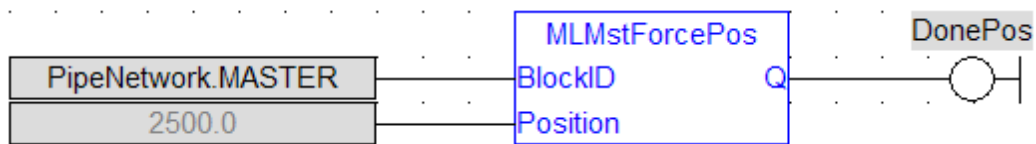
```
MLMstForcePos ( PipeNetwork.MASTER, 2500.0 );
```

Ladder Diagram



NOTE You must use a pulse contact to start the FB

Function Block Diagram



1.1.13.4 MLMstInit

Description

Initializes a Master TMP (trapezoidal motion profile) generator block. This function is automatically created when the MLMaster Block is included in the Pipe Network Editor. Based on the parameters defined in the Master pipe block (see figure below), the Inputs for this function are initialized by default.

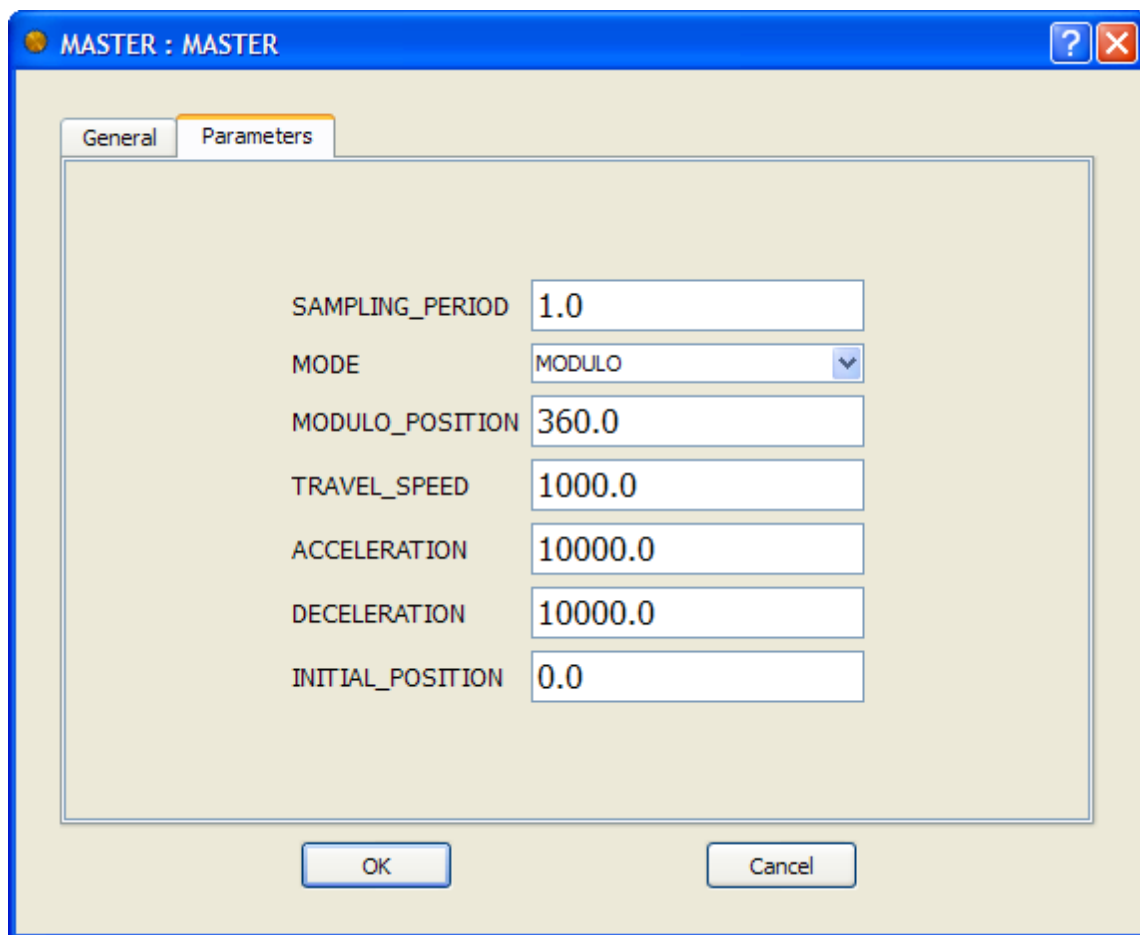


Figure 1-25: TMP Initialization

Arguments

Input

Block ID	Description	ID name of the Master Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
ModuloPosition	Description	Modulo Position for cyclic motion systems expressed in user logical units (Position Rollover Value)
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
Period	Description	Sampling period of the generator expressed according to the update cycle (e.g. 2.0 means the sampling is done once every 2 cycles)
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

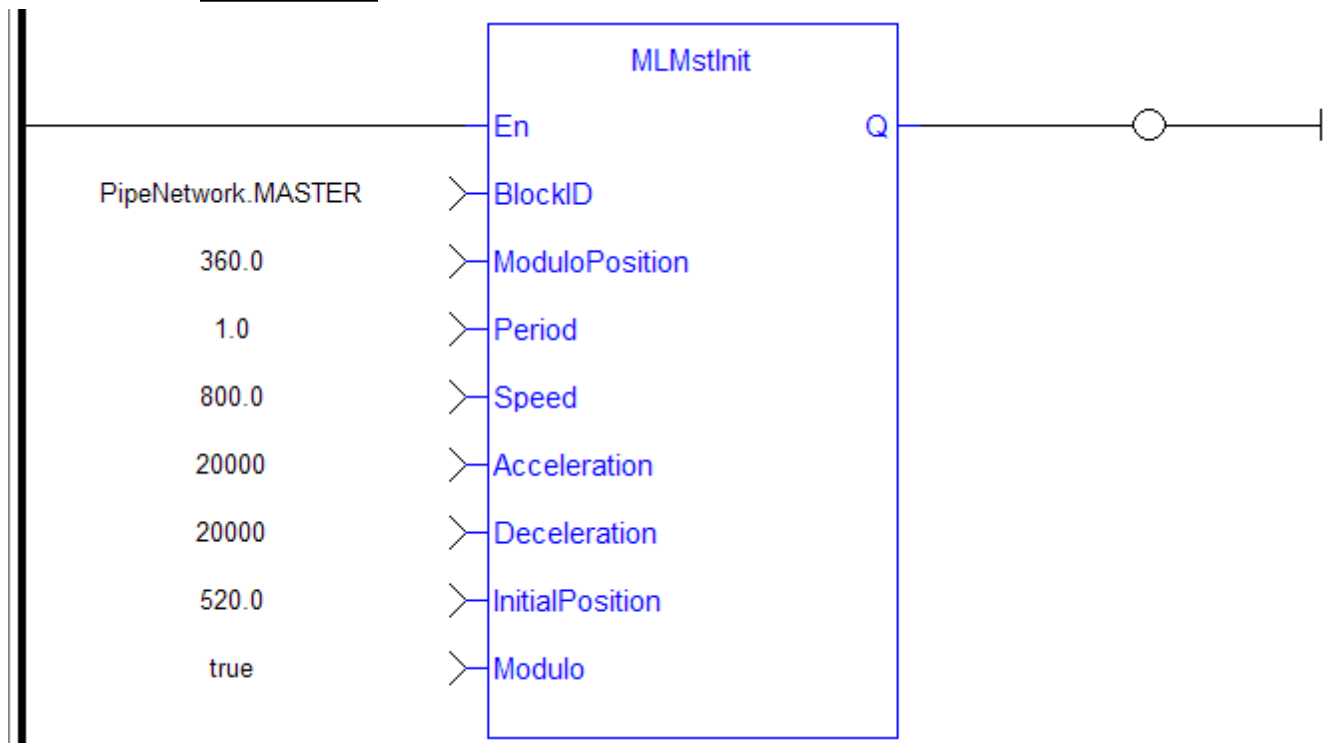
Speed	Description	Travel speed value expressed in user logical units per second. The travel speed value is used to set the constant speed part of the trapezoidal motion profile
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	Description	Acceleration value expressed in user logical units per second squared. The acceleration value is always used to generate the first part of the trapezoidal motion profile
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Deceleration	Description	Deceleration value expressed in user logical units per second squared. The deceleration value is always used to generate the last part of the trapezoidal motion profile
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Initial Position	Description	Initial position value expressed in user logical units. Used only at the pipe activation to initialize the position starting point
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
Modulo	Description	The available modes are Modulo (True) or No modulo (False)
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
<u>Output</u> Default (.Q)	Description	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a

Example

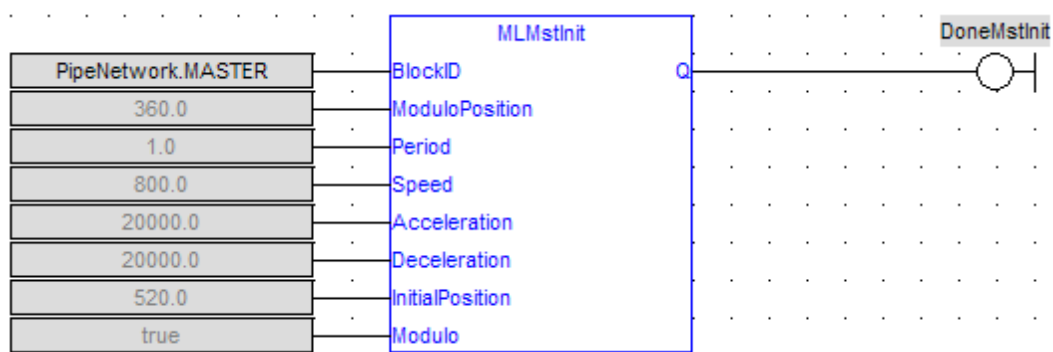
Structured Text

```
MLMstInit( PipeNetwork.MASTER, 360.0, 1.0, 1000.0,
10000.0, 10000.0, 0.0, true );
```

Ladder Diagram



Function Block Diagram



1.1.13.5 MLMstReadAccel

Description

Get the presently used value for acceleration of a master block.

Arguments

Input

EN	Description	Enables FB to be executed
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Block ID	Description	ID name of the Master Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

OK	Description	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a
Acceleration	Description	Returns Acceleration value
	Data type	LREAL
	Unit	User unit/sec ²

Related Functions

MLMstReadSpeed

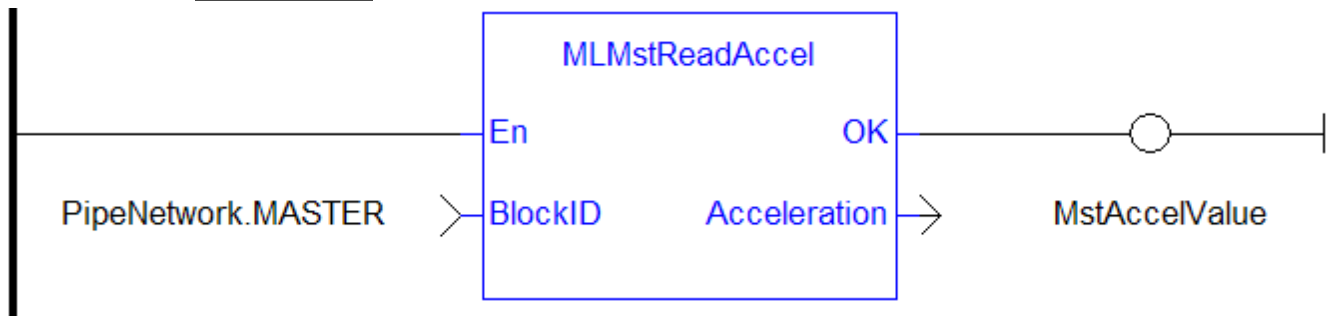
MLMstReadDecel

Example

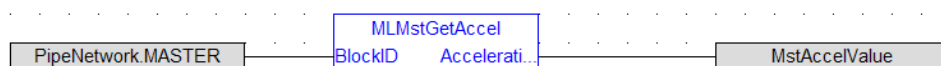
Structured Text

```
MLMstReadAccel( PipeNetwork.MASTER );
```

Ladder Diagram



Function Block Diagram



1.1.13.6 MLMstReadDecel

Description

Get the presently used value for deceleration of a master block.

Arguments

Input

EN	Description Data type Range Unit Default	Enables FB to be executed BOOL 0, 1 n/a —
Block ID	Description Data type Range Unit Default	ID name of the Master Block DINT [-2147483648, 2147483648] n/a —

Output

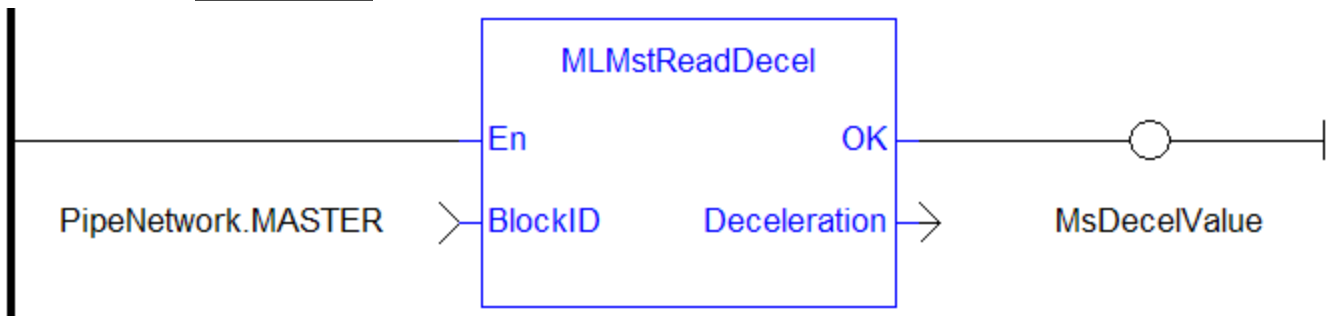
OK	Description Data type Unit	Returns true when function successfully executes BOOL n/a
Deceleration	Description Data type Unit	Returns Deceleration value LREAL User unit/sec ²

Example

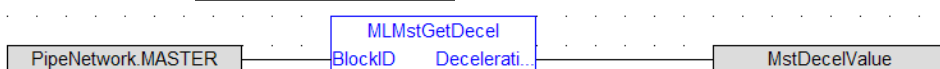
Structured Text

```
MLMstReadDecel( PipeNetwork.MASTER );
```

Ladder Diagram



Function Block Diagram



1.1.13.7 MLMstReadInitPos

Description

Get the presently used value for initial position of a master block.

Arguments

Input

EN	Description	Enables FB to be executed
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—

Block ID	Description	PipeNetwork Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

OK	Description	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a

Position	Description	Returns Initial Position
	Data type	LREAL
	Unit	User unit

Example

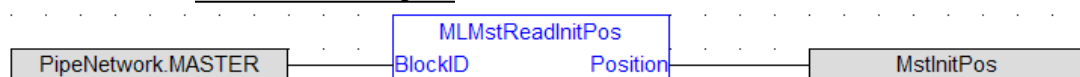
Structured Text

```
MstInitPos := MLMstReadInitPos ( PipeNetwork.MASTER );
```

Ladder Diagram



Function Block Diagram



1.1.13.8 MLMstReadSpeed

Description

Get the presently used value for speed of a master block.

Arguments

Input

EN	Description Data type Range Unit Default	Enables FB to be executed BOOL 0, 1 n/a —
Block ID	Description Data type Range Unit Default	ID name of the Master Block DINT [-2147483648, 2147483648] n/a —

Output

OK	Description Data type Unit	Returns true when function successfully executes BOOL n/a
Speed	Description Data type Unit	Returns current Speed LREAL User unit/sec

Related Functions

MLMstReadAccel

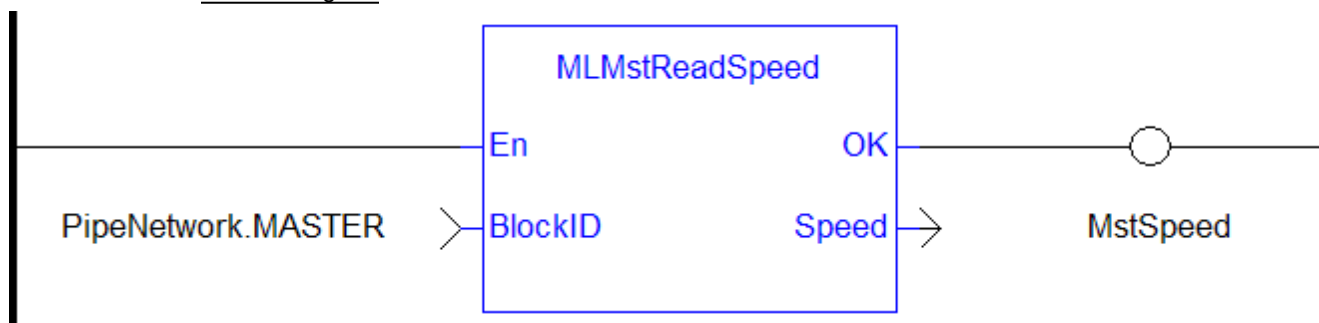
MLMstReadDecel

Example

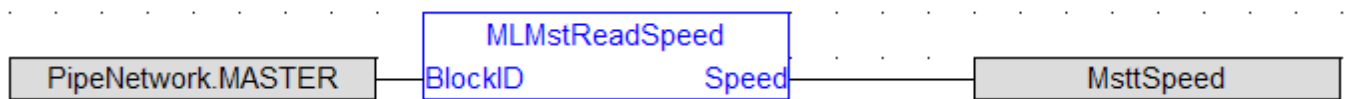
Structured Text

```
MstSpeed := MLMstReadSpeed( PipeNetwork.MASTER );
```

Ladder Diagram



Function Block Diagram

**1.1.13.9 MLMstRel****Description**

Performs a move for a specified distance relative to the current position. Returns TRUE if the function succeeded.

Arguments**Input**

EN	Description Data type Range Unit Default	Enables FB to be executed BOOL 0, 1 n/a —
Block ID	Description Data type Range Unit Default	ID name of the Master Block DINT [-2147483648, 2147483648] n/a —
DeltaPos	Description Data type Range Unit Default	Relative distance to move LREAL — User unit —
<u>Output</u>		
Default (.Q)	Description Data type Unit	Returns true when function successfully executes BOOL n/a

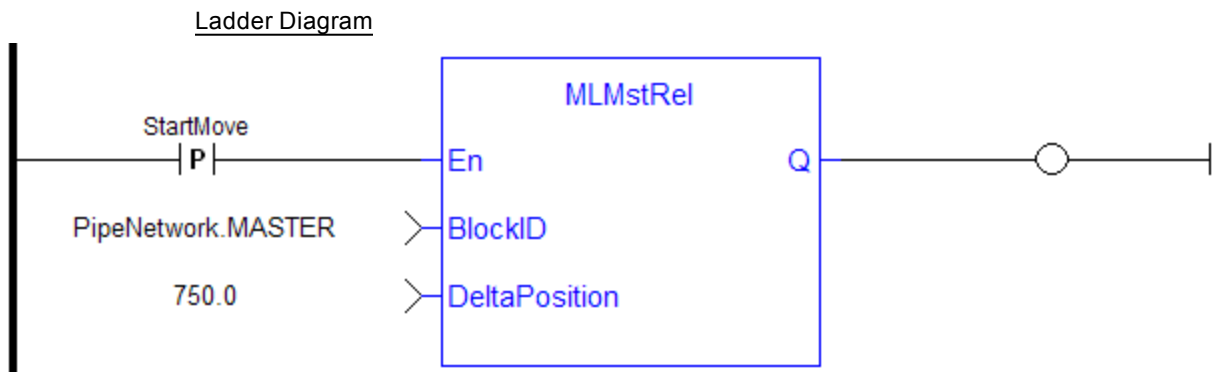
Related Functions

MLMstWriteSpeed

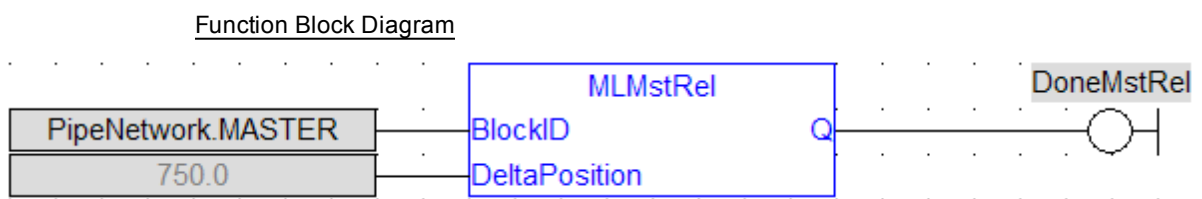
MLMstWriteDecel

Example**Structured Text**

```
MLMstRel ( PipeNetwork.MASTER, 750.0 );
```



NOTE You must use a pulse contact to start the FB



1.1.13.10 MLMstRun

Description

Jog at the specified speed. Returns TRUE if the function succeeded.

Arguments

Input

EN	Description Data type Range Unit Default	Enables FB to be executed BOOL 0, 1 n/a —
Block ID	Description Data type Range Unit Default	ID name of the Master Block DINT [-2147483648, 2147483648] n/a —
Speed	Description Data type Range Unit Default	Speed to jog motor LREAL — User unit/sec —

Output

Default (.Q)	Description Data type Unit	Returns true when function successfully executes BOOL n/a
---------------------	----------------------------------	---

Related Functions

MLMstWriteSpeed

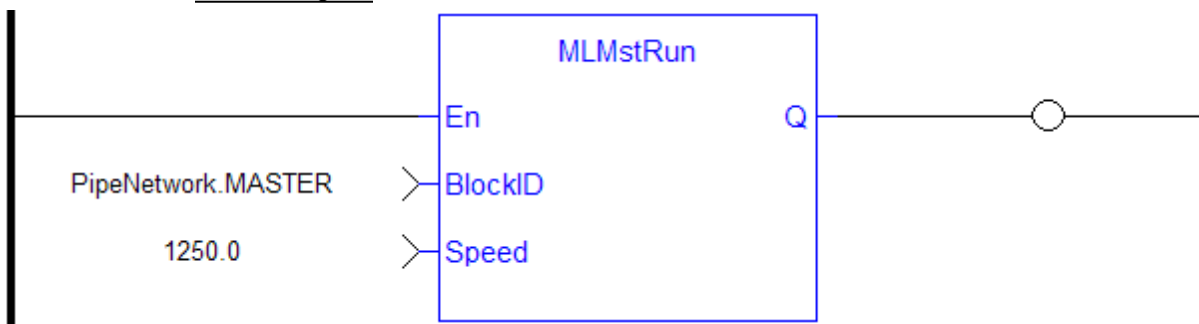
MLMstWriteDecel

Example

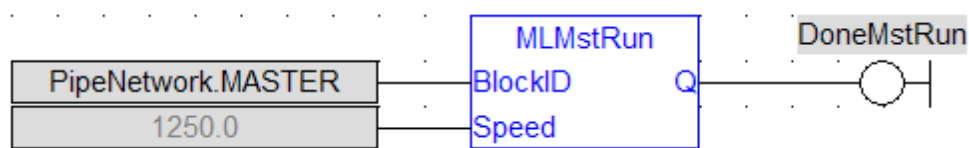
Structured Text

```
MLMstRun( PipeNetwork.MASTER, 1250.0 );
```

Ladder Diagram



Function Block Diagram



1.1.13.11 MLMstStatus

Description

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. The following terms are relevant to the returned values.

Term	Definition
Running	Speed is non-zero
Stopped	Speed is zero
Positioning	A target position has been programmed with a relative, additive or absolute command.

Status	Definition
0	(New speed programmed) is entered when a MLMstRun command is programmed and the Generator is not at the new speed.

Status	Definition
1	(Stable state Running or Stopped) is entered when the programmed speed is reached and a run command has been programmed. (Stable state Running or Stopped) is entered when motion is completed and a positioning command has been programmed
2	(Speed change) is entered when the current speed is greater than the travel speed when a positioning move is commanded.
3	(Speed reversal while positioning) is entered when the distance to go requires a speed reversal.
4	(Acceleration while positioning) current speed is below the travel speed
5	(Constant Speed while positioning) current speed is at the travel speed.
6	(Deceleration while positioning) current speed is changing to achieve the target position at zero speed.
7	(Micro step) is entered when a small change in position is required and the current speed is zero.

Arguments

Input

EN

Description	Enables FB to be executed
Data type	BOOL
Range	0, 1
Unit	n/a
Default	—

Block ID

Description	ID name of the Master Block
Data type	DINT
Range	[-2147483648, 2147483648]
Unit	n/a
Default	—

Output

OK

Description	Returns true when function successfully executes
Data type	BOOL
Unit	n/a

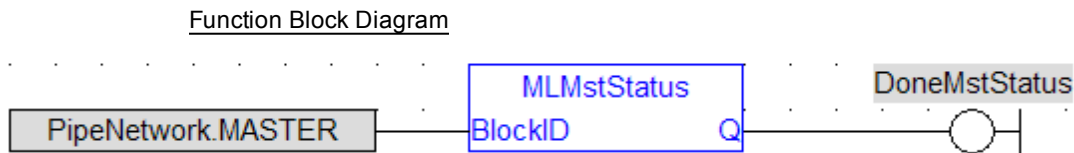
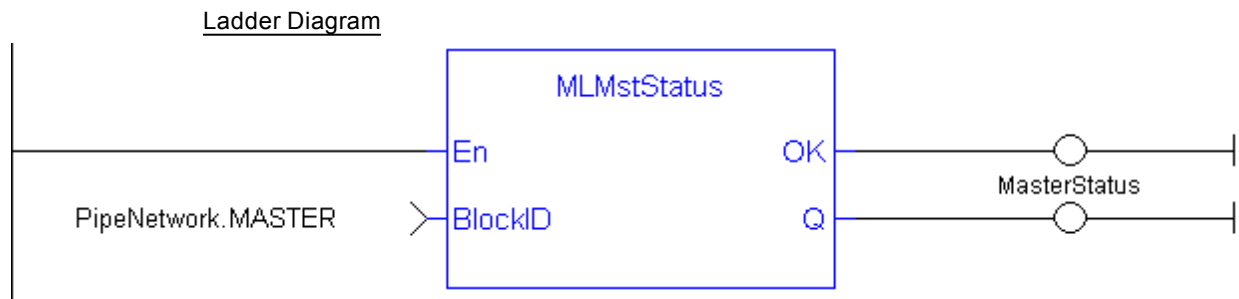
Default (.Q)

Description	Returns the status of the generator
Data type	DINT
Unit	n/a

Example

Structured Text

```
MasterStatus := MLMstStatus( PipeNetwork.MASTER );
```



1.1.13.12 MLMstWriteAccel

Description

Set the acceleration of a master block. Returns TRUE if the function succeeded.

Arguments

Input

EN	Description Data type Range Unit Default	Enables FB to be executed BOOL 0, 1 n/a —
Block ID	Description Data type Range Unit Default	ID name of the Master Block DINT [-2147483648, 2147483648] n/a —
Acceleration	Description Data type Range Unit Default	Acceleration value expressed in user logical units per second squared LREAL — User unit/sec ² —

Output

Default (.Q)	Description Data type Unit	Returns true when function successfully executes BOOL n/a
---------------------	----------------------------------	---

Related Functions

MLMstAbs

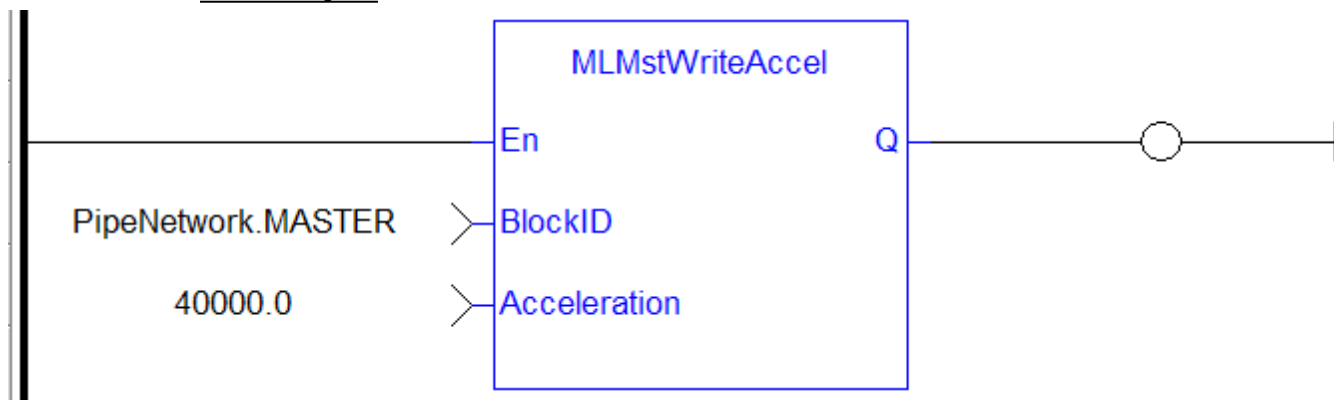
MLMstRel
 MLMstWriteSpeed
 MLMstWriteDecel

Example

Structured Text

```
MLMstWriteAccel( PipeNetwork.MASTER, 40000.0 );
```

Ladder Diagram



Function Block Diagram



1.1.13.13 MLMstWriteDecel

Description

Set the deceleration of a master block. Returns TRUE if the function succeeded.

Arguments

Input

EN	Description Data type Range Unit Default	Enables FB to be executed BOOL 0, 1 n/a —
Block ID	Description Data type Range Unit Default	ID name of the Master Block DINT [-2147483648, 2147483648] n/a —

<u>Deceleration</u>	Description	Deceleration value
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—

<u>Output</u>	Description	Returns true when function successfully executes
<u>Default (.Q)</u>	Data type	BOOL
	Unit	n/a

Related Functions

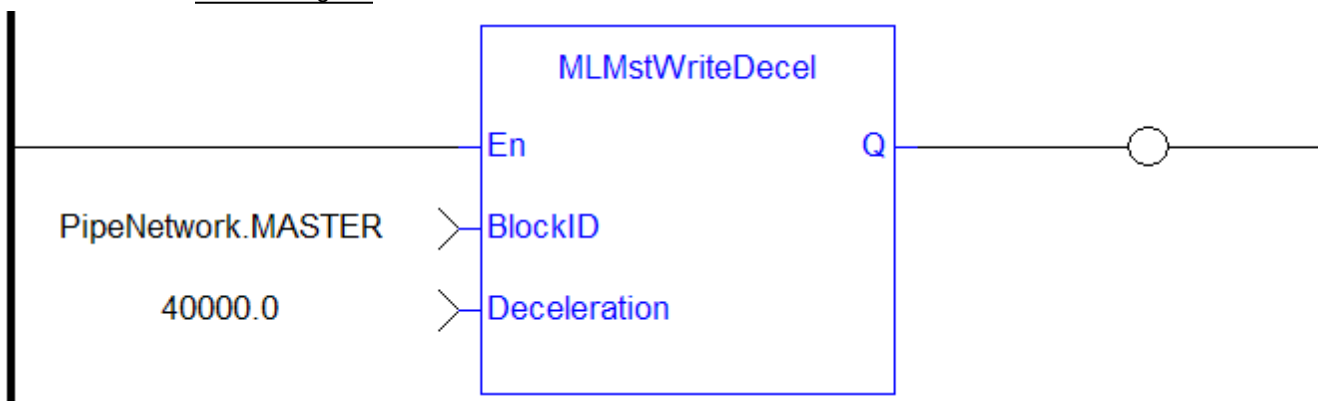
MLMstWriteSpeed

Example

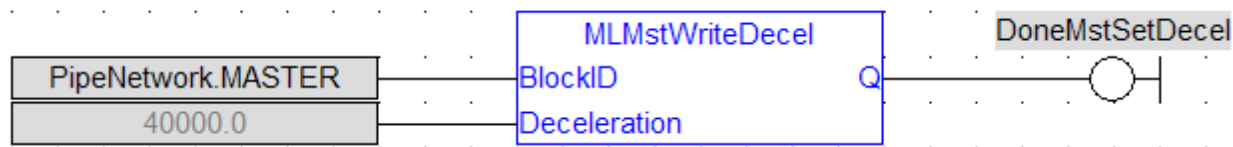
Structured Text

```
MLMstWriteDecel( PipeNetwork.MASTER, 40000.0 );
```

Ladder Diagram



Function Block Diagram



1.1.13.14 MLMstWriteInitPos

Description

Set the initial position of a master block. Returns TRUE if the function succeeded.

Arguments

Input

EN	Description	Enables FB to be executed
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Block ID	Description	ID name of the Master Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Position

Description	Initial position
Data type	LREAL
Range	—
Unit	User unit
Default	—

Output

Default (.Q)	Description	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a

Example

Structured Text

```
MLMstWriteInitPos ( PipeNetwork.MASTER, 120.0 );
```

Ladder Diagram



Function Block Diagram



1.1.13.15 MLMstWriteSpeed

Description

Set the speed of a master block. Returns TRUE if the function succeeded.

Arguments

Input

EN	Description Data type Range Unit Default	Enables FB to be executed BOOL 0, 1 n/a —
Block ID	Description Data type Range Unit Default	ID name of the Master Block DINT [-2147483648, 2147483648] n/a —
Speed	Description Data type Range Unit Default	Speed of the motion LREAL — User unit/sec —
<u>Output</u>		
Default (.Q)	Description Data type Unit	Returns true when function successfully executes BOOL n/a

Related Functions

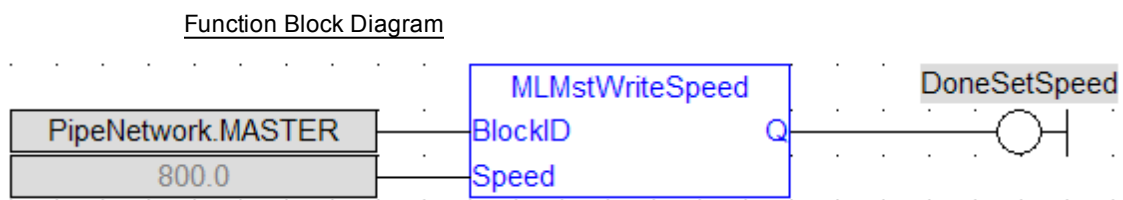
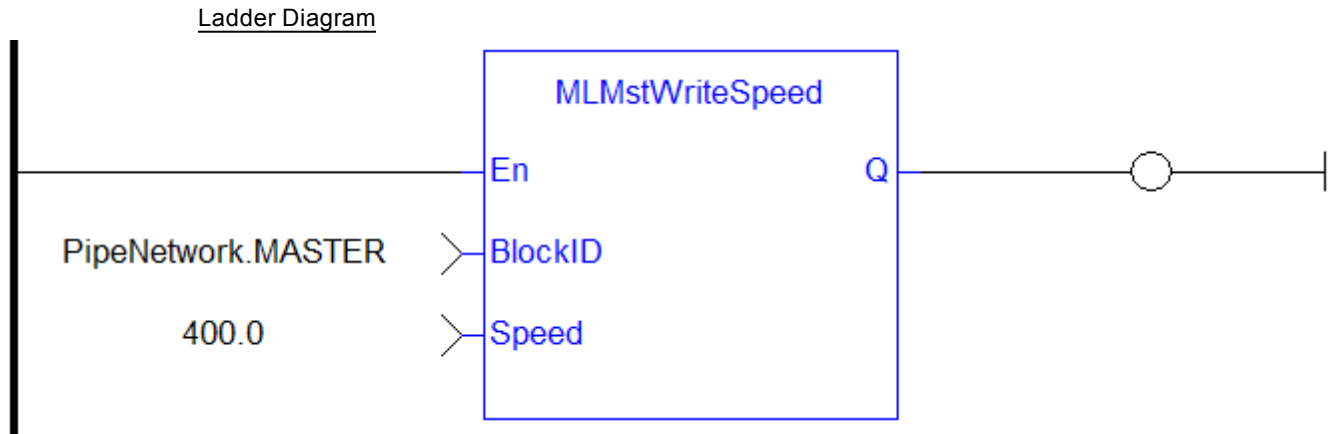
MLMstWriteSpeed

MLMstWriteDecel

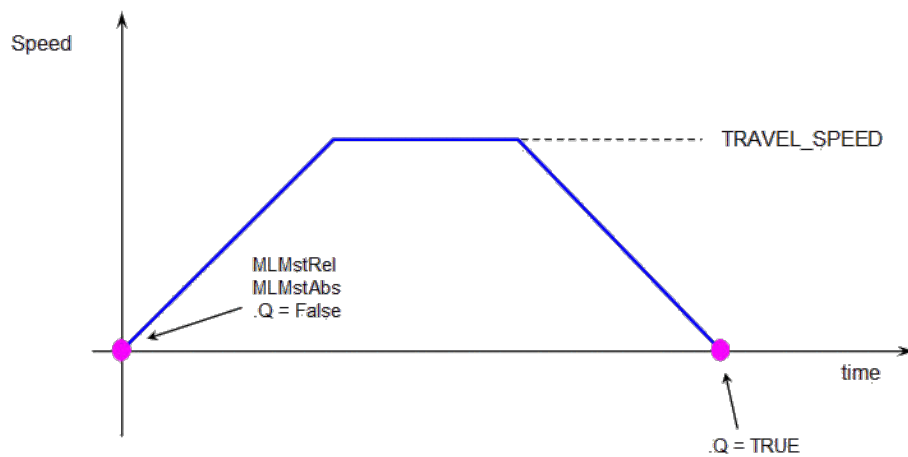
Example

Structured Text

```
MLMstWriteSpeed( PipeNetwork.MASTER, 400.0 );
```



1.1.13.16 Usage example of Master Functions



MLMstRun(0.0) reduce the speed down to 0.

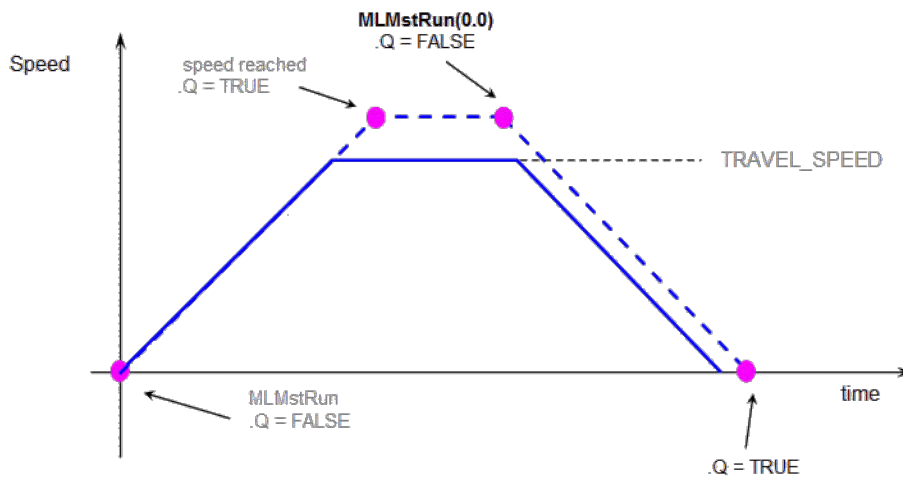


Figure 1-26: Master Functions Usage

1.1.14 Motion Library - Phaser

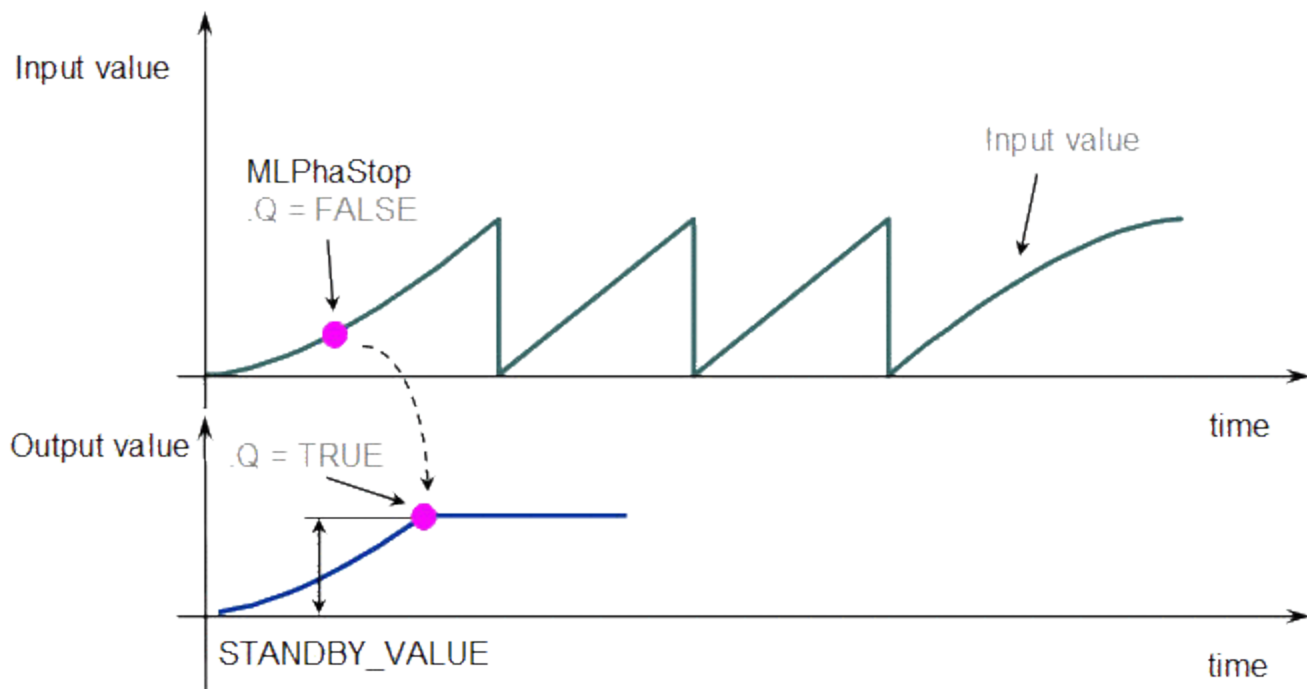
TIP For usage example about Phaser Functions, see page 125

Names	Description	Return type
MLPhaInit	Initializes a phaser Pipe Block	BOOL
MLPhaReadPhase	Gets the phase value of a phaser block	None
MLPhaReadSlope	Gets the phase slope value of a phaser block	None
MLPhaWritePhase	Sets the phase value of a phaser block	BOOL
MLPhaWriteSlope	Sets the phase slope value of a phaser block	BOOL
MLPhaReadActPhase		

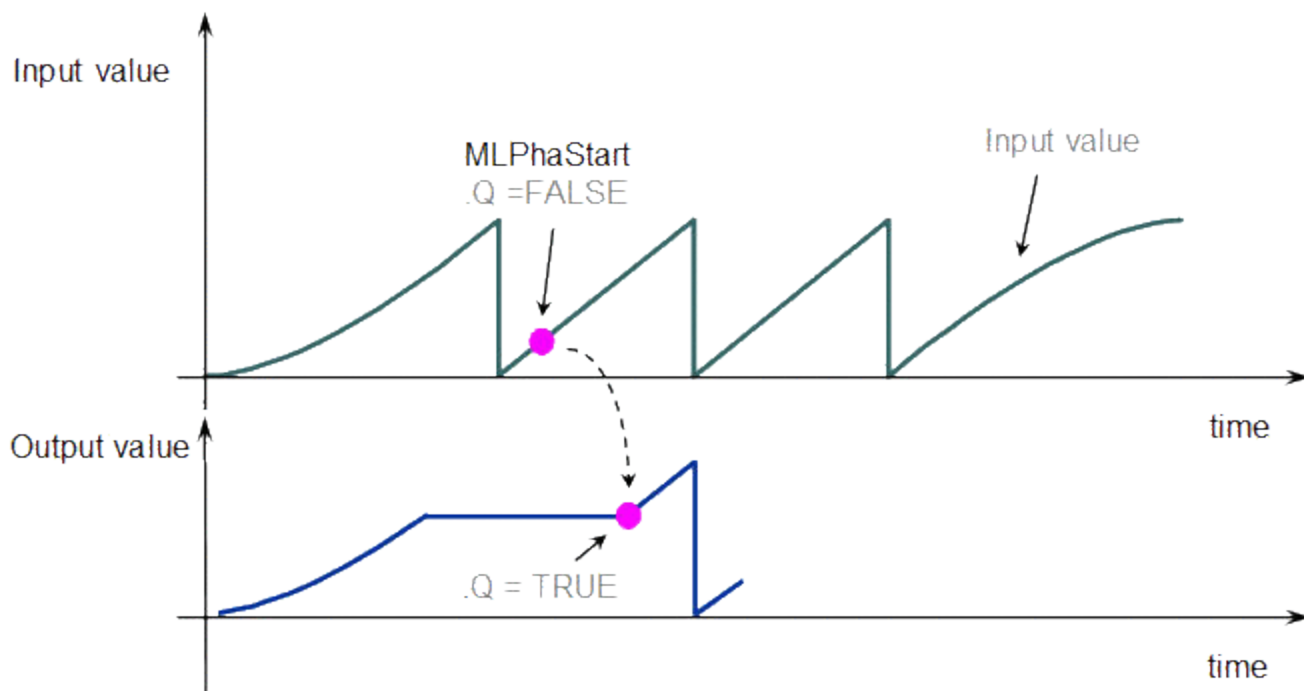
TIP There is a delay when using an external encoder. The delay is five cycles (2 cycles to read the encoder from the AKD via EtherCAT, 1 cycle for computing, 2 cycles for sending the new position set point to the AKD). This lag error is speed proportional (5 cycles * speed). A Phaser block can be used to compensate for this lag.

1.1.14.1 Usage example of Phaser Functions

When you call the **MLPhaStop** function, the output becomes True as soon as the Standby_Value is reached.



When you call the **MLPhaStart** function, the output becomes True as soon as the Input value equals the Standby_Value.



You can call the **MLPhaWritePhase** function to modify the phase slope.

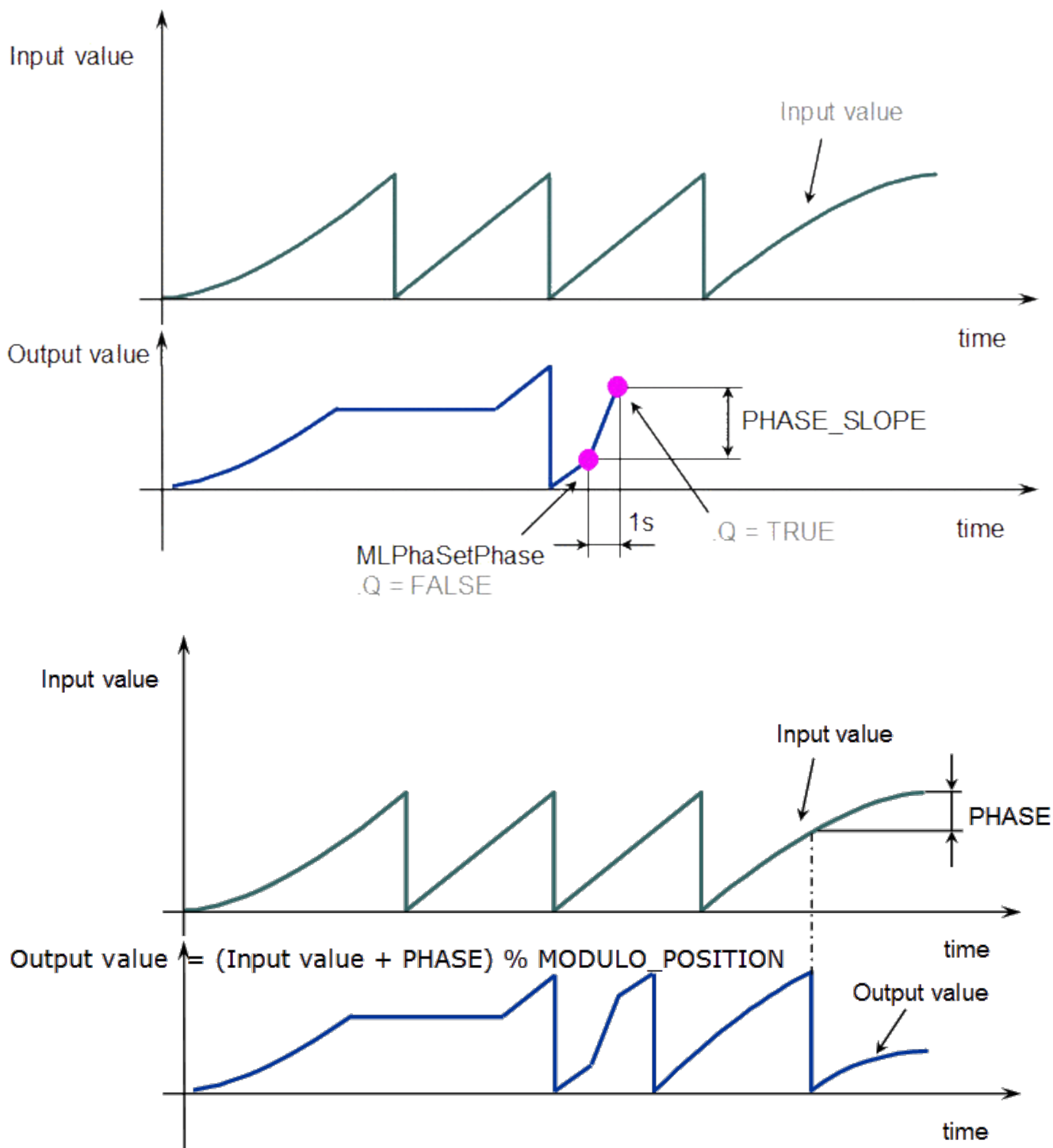


Figure 1-27: Phaser Functions Usage

NOTE % MODULO_POSITION is in the equation to take into account the modulo (periodicity) of the value.

1.1.15 Motion Library - PMP

Name	Description	Return type
MLPmpAbs	Moves to an Absolute Position	BOOL

Name	Description	Return type
MLPmpForcePos	Forces the specified position. Possible only when the block is not moving.	BOOL
MLPmpInit	Initializes a PMP object (Parabolic Motion Profile generator) with user-defined settings	BOOL
MLPmpReadAccel	Gets the Acceleration parameter of a PMP block	None
MLPmpReadFstSpd	Gets the FirstTravelSpeed parameter of a PMP block	None
MLPmpReadInitPos	Gets the InitialPosition parameter of a PMP block	None
MLPmpReadJerk	Gets the Jerk parameter of a PMP block	None
MLPmpReadLstSpd	Gets the LastTravelSpeed parameter of a PMP block	None
MLPmpRel	Does two subsequent relative moves	BOOL
MLPmpRun	Jog the generator at the specified speed	BOOL
MLPmpStatus	Returns the status of the PMP block generator	None
MLPmpWriteAccel	Sets the acceleration parameter of a PMP block	BOOL
MLPmpWriteFstSpd	Sets the FirstTravelSpeed parameter of a PMP block	BOOL
MLPmpWriteJerk	Sets the jerk parameter of a PMP block	BOOL
MLPmpWriteLstSpd	Sets the LastTravelSpeed parameter of a PMP block	BOOL

1.1.16 Motion Library - Sampler

Name	Description	Return type
MLSmpConnect	Connects a sampler to a pipe network axis or pipe block	BOOL
MLSmpConnectEx	Connects a sampler to the specified external data source	BOOL
MLSmpInit	Initializes a sampler object	BOOL

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.

1.1.16.1 MLSmpConnect

Description

Connect a sampler to an axis or pipe block as a value source. Returns TRUE if the function succeeded.

Arguments

Input

BlockID	Description	ID Name of the SMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]

AxisID	Unit	n/a
	Default	—
	Description	ID Name of the Axis or Pipe Block the sampler is connected to
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Default (.Q)	Description	Returns True if the Sampler is connected
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

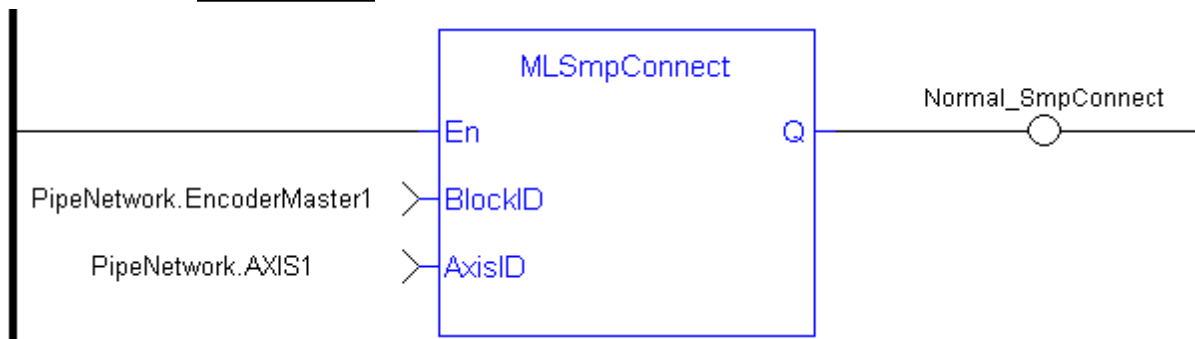
Example

Structured Text

```

MLSmpConnect ( PipeNetwork.EncoderMaster1, AxisID(*DINT*)
) ;
    
```

Ladder Diagram



Function Block Diagram



1.1.16.2 MLSmpConnectEx

Description

Connect a sampler to the specified data source. Returns TRUE if the function succeeded.

Arguments

	<u>Input</u>	
BlockID	Description	ID Name of the SMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
DriverName	Description	Driver name of the external source or Axis Name.
		Examples: 'EtherCATDriver' 'AXIS1' 'PLCOpenAxis1'
	Data type	STRING
	Range	—
	Unit	n/a
	Default	—

SourceConfig

Description Configuration of the source (for EtherCAT motion bus) or the specific variable inside an axis object.

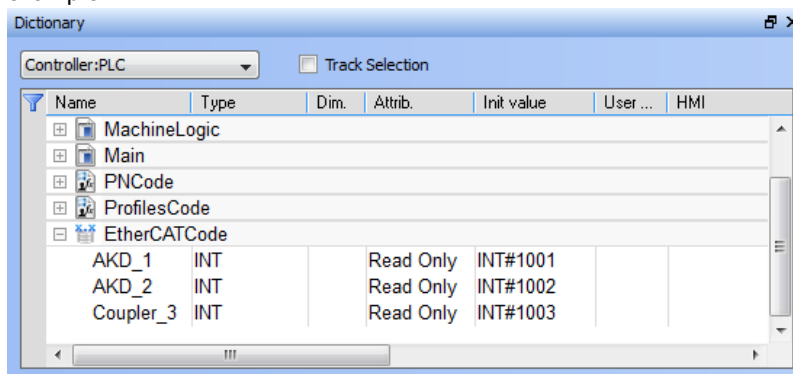
EtherCAT:

The string must have the form '<EtherCAT address>:<source type>'

For example: '1001:Position actual value 2'

- This maps the secondary feedback from an axis in the created pipe network

- The <EtherCAT address> can be found by looking in *EtherCATcode* item in the project Dictionary. For example:



Example for AXIS1 'PipePosition'

(One of the following:

- 'PipePosition'
- 'ReferencePosition'
- 'GeneratorPosition'
- 'ActualPosition'
- 'FeedbackPosition'
- '2ndFeedbackPosition'
- 'ActualVelocity'
- 'ActualTorque'
- 'FollowingError'
- 'CurrentPosition')

PLCOpen Axis:

Example: PLCOpenAxis1 'ActualPosition'

(One of the following:

- 'ActualPosition'
- 'CommandPosition'
- 'NormalCmdPos'
- 'SuperimposedCmdPos'
- 'PhaseCmdPos')

Data Type STRING

Range —

Unit n/a

Default —

Output

Default (.Q)

Description Function block is operational

Data type BOOL

Unit n/a

Return Type

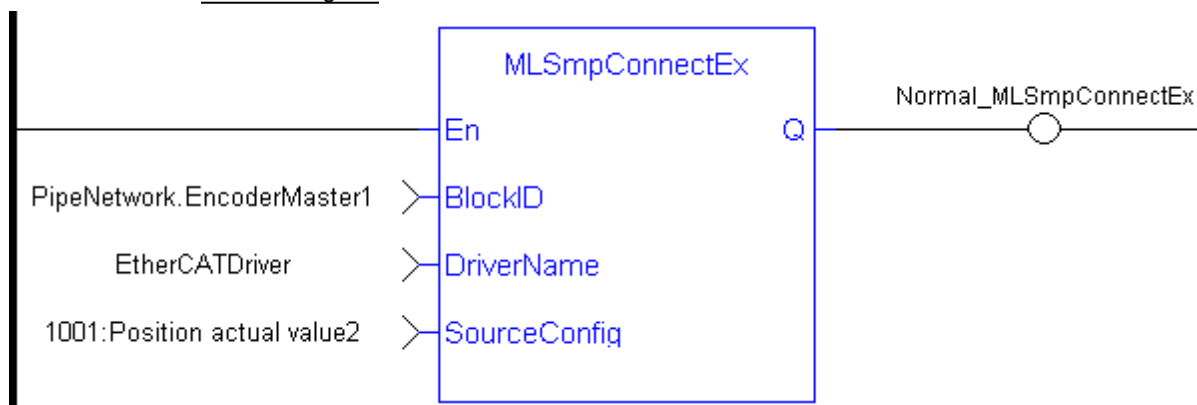
BOOL

Example

Structured Text

```
MLSmpConnectEx(PipeNetwork.Feedback2, 'EtherCATDriver',
'1001:Position actual value 2');
```

Ladder Diagram



Function Block Diagram



1.1.16.3 MLSmplnit

Description

The purpose of the sampler block is to periodically sample and place into a pipe some output of a source object. The sampled output can typically 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) and one or more pipes and 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.

Arguments

Input

BlockID	Description
BlockID	ID Name of the PMP function block in the Pipe Network

	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
SamplingPeriod	Description	period that the device is sampled
	Data type	LREAL
	Range	0.25 to ?
	Unit	millisecond
	Default	1.0
Mode	Description	Sampled output can be either position or velocity
	Data type	DINT
	Range	[1 , 2] Position or Speed
	Unit	n/a
	Default	position
InputModuloPosition	Description	Period of the input signal. This should be set equal to 2^{32} (4294967296).
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	360.0
OutputModuloPosition	Description	Period of the output signal
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	360.0
Default (.Q)	<u>Output</u>	
	Description	Smp Block successfully initiated
	Data type	BOOL
	Unit	n/a

Example**Structured Text**

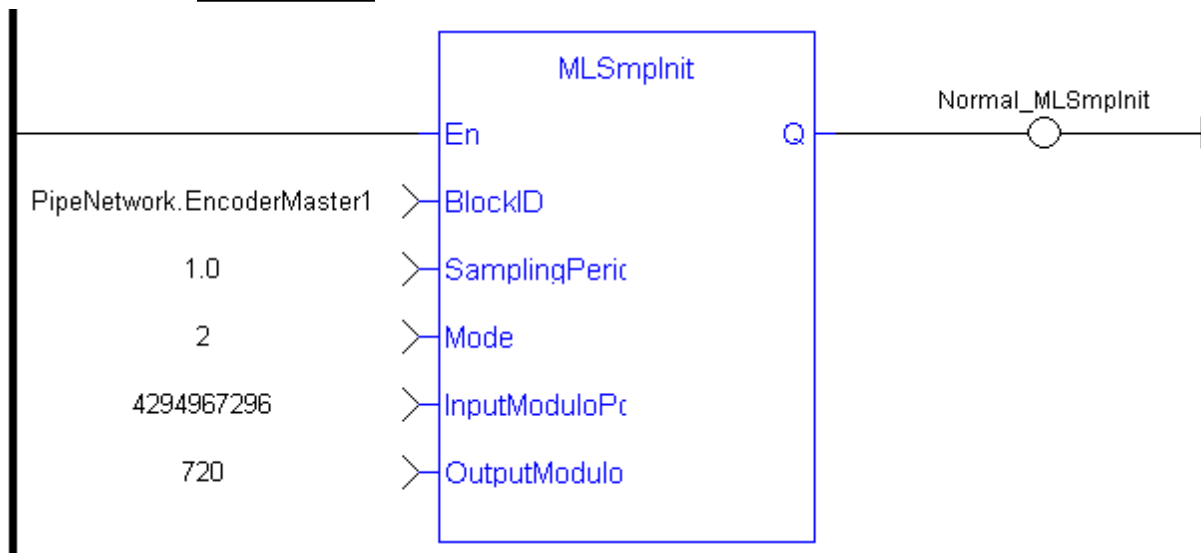
```

MLSmpInit( PipeNetwork.EncoderMaster1, SamplingPeriod
(*LREAL*), Mode(*DINT*),

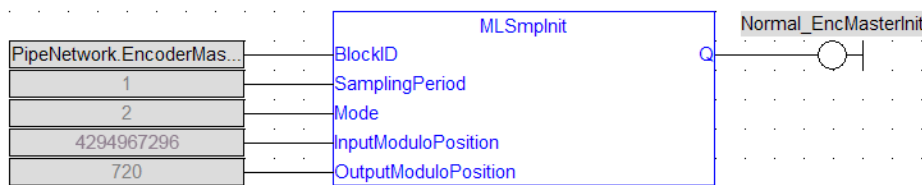
InputModuloPosition(*LREAL*), OutputModuloPosition(*LREAL*)
) );

```

Ladder Diagram



Function Block Diagram



1.1.17 Motion Library - Synchronizer

TIP For usage example about Synchronizer Functions, see page 142

Name	Description	Return type
MLSynclnit	Initializes a synchronizer Pipe Block	BOOL
MLSyncReadDeltaS	Gets the output phasing value of a synchronizer block	None
MLSyncStart	Starts a synchronization of a synchronizer Pipe Block	BOOL
MLSyncStop	De-synchronizes a synchronizer Pipe Block	BOOL
MLSyncWriteDeltaS	Sets the output phasing value of a synchronizer block	BOOL

1.1.17.1 MLSynclnit

Description

Initializes a synchronizer Pipe Block. 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);
```

Arguments**Input**

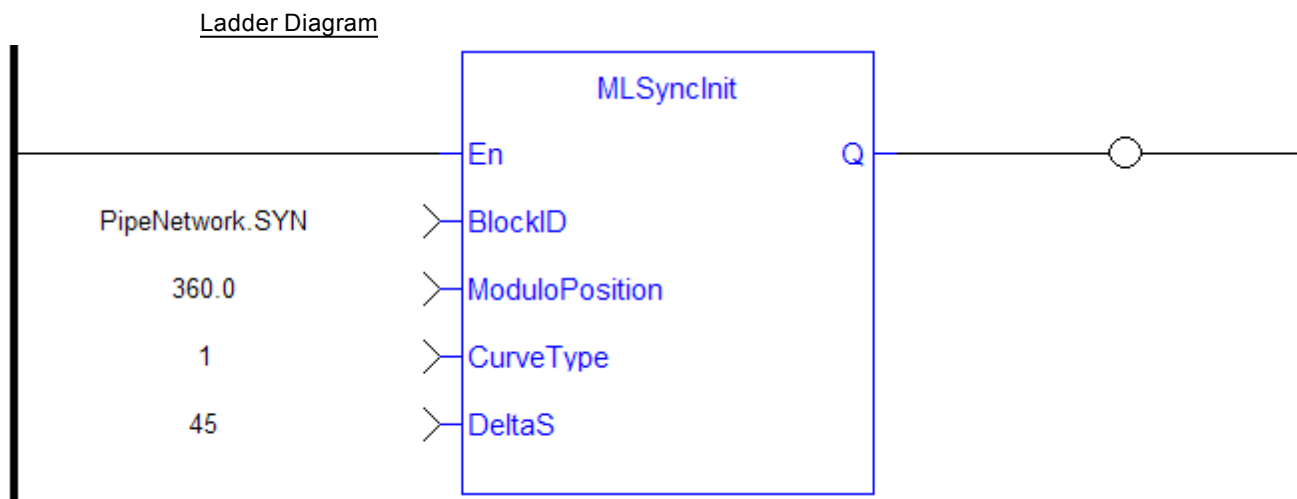
BlockID	Description	Name of the Pipe Network Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
ModuloPosition	Description	The modulo distance
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
CurveType	Description	The curve type to the motion when starting and stopping synchronization. Option are Parabolic or Polynomial
	Data type	DINT
	Range	[1 , 2] (1 = Parabolic, 2 = Polynomial)
	Unit	n/a
	Default	—
DeltaS	Description	The Distance to get in or out of synchronization. This parameter is used in the MLSyncStart and MLSyncStop FunctionBlocks
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
<u>Output</u>		
Default (.Q)	Description	Function Block Execute Successfully
	Data type	BOOL
	Unit	n/a

Related Functions

MLSyncWriteDeltaS

Example**Structured Text**

```
MLSyncInit( PipeNetwork.SYN, 360, 1, 30 );
```



1.1.17.2 MLSyncReadDeltaS

Description

Gets the output phasing value of a synchronizer block. Output phasing is the distance or the slope the output takes to synchronize with the input when MLSyncStart Block is executed (see "Figure 1-28: Get Output Phasing after MLSyncStart " on page 136). It also affects the distance or the slope the output takes to desynchronize with the input and come to a stop when MLSyncStop Block is executed (see "Figure 1-29: Get Output Phasing after MLSyncStop " on page 137).

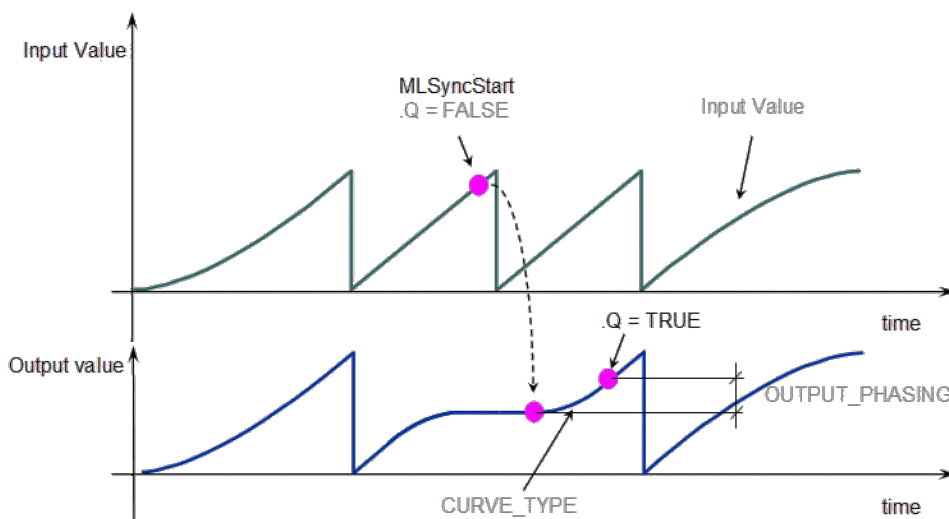


Figure 1-28: Get Output Phasing after MLSyncStart

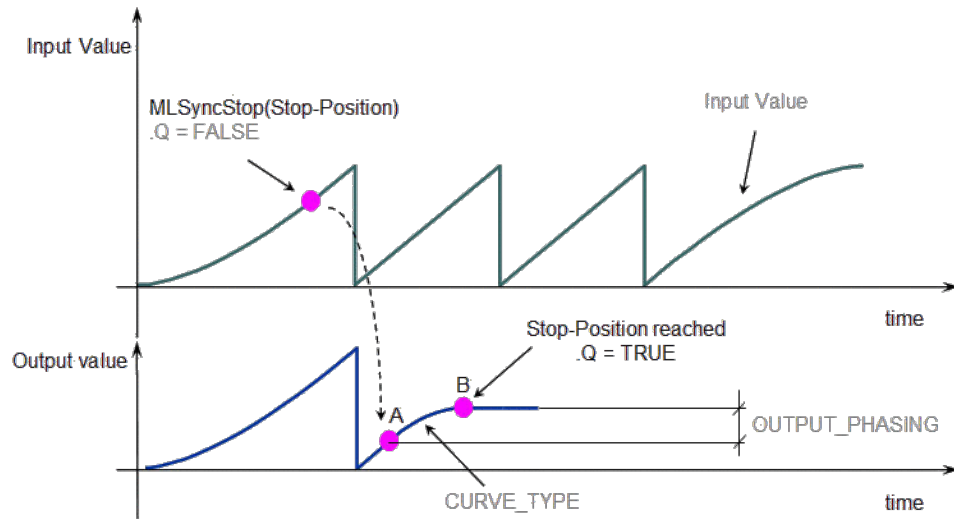


Figure 1-29: Get Output Phasing after MLSyncStop

Arguments

Input

BlockID	Description	Name of the Pipe Network Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

DeltaS	Description	Present Delta Slope value
	Data type	LREAL
	Unit	User unit

Related Functions

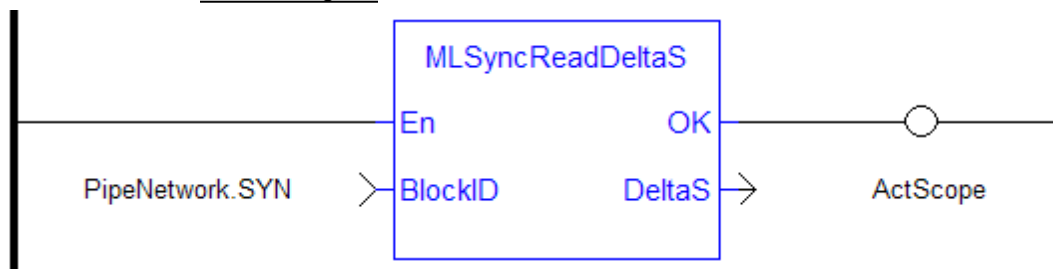
MLSyncWriteDeltaS

Example

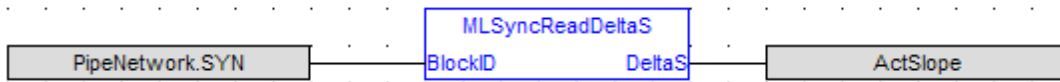
Structured Text

```
ActScope := MLSyncReadDeltaS( PipeNetwork.SYN );
```

Ladder Diagram



Function Block Diagram



1.1.17.3 MLSyncStart

Description

Start a synchronization of a synchronizer Pipe Block. Returns TRUE if the function succeeded.

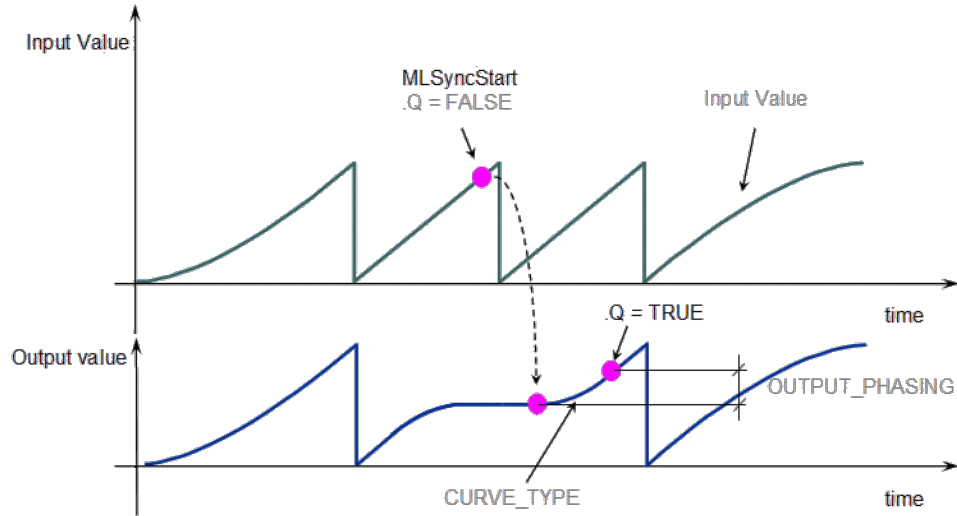


Figure 1-30: MLSyncStart

Arguments

Input

<code>BlockID</code>	Description	Name of the Pipe Network Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

<code>Default (.Q)</code>	Description	Function Block Execute Successfully
	Data type	BOOL
	Unit	n/a

Example

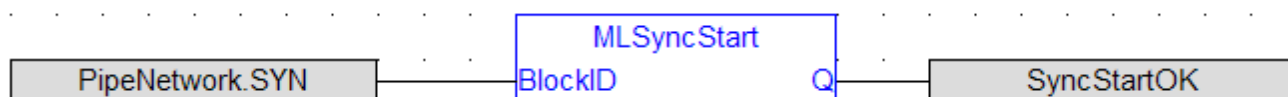
Structured Text

```
MLSyncStart( PipeNetwork.SYN );
```

Ladder Diagram



Function Block Diagram



1.1.17.4 MLSyncStop

Description

De-synchronizes a synchronizer Pipe Block. Returns TRUE if the function succeeded.

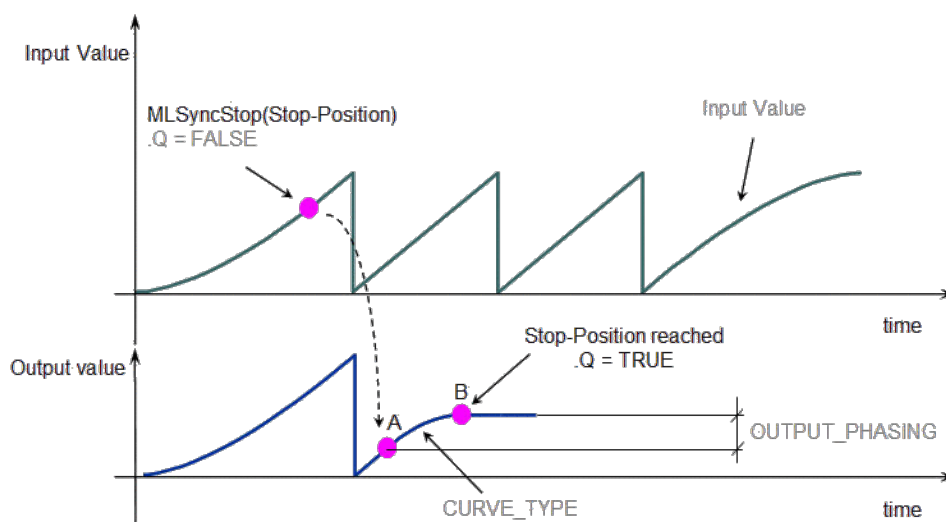


Figure 1-31: MLSyncStop

Arguments

Input

Position	Description	Motion Stop Position
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

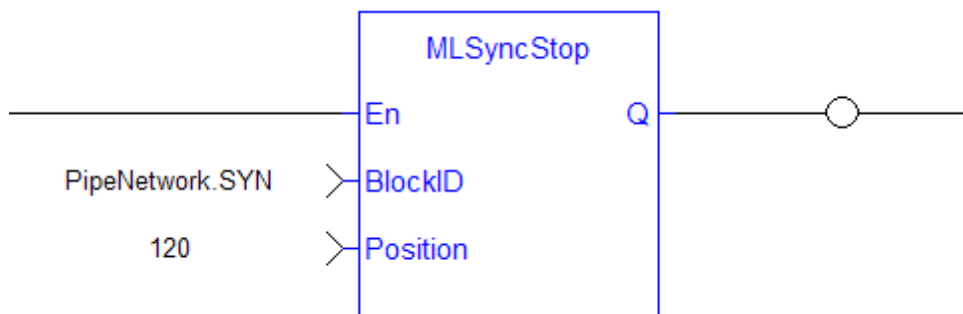
Default (.Q)	Description	Function Block Execute Successfully
	Data type	BOOL
	Unit	n/a

Example

Structured Text

```
MLSyncStop( PipeNetwork.SYN , 120 );
```

Ladder Diagram



Function Block Diagram



1.1.17.5 MLSyncWriteDeltaS

Description

Set the output phasing value of a synchronizer block. 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 desynchronize with the input and come to a stop when MLSyncStop Block is executed.

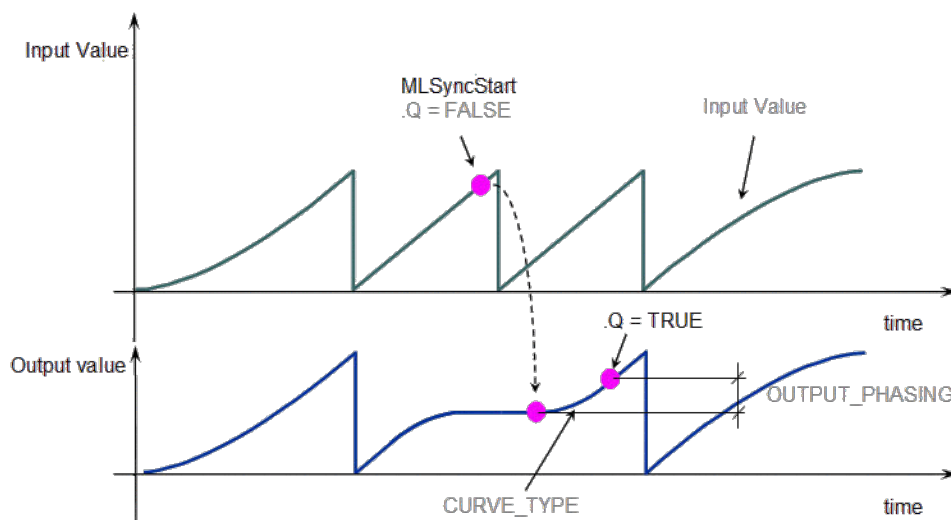


Figure 1-32: Set output phasing after MLSyncStart

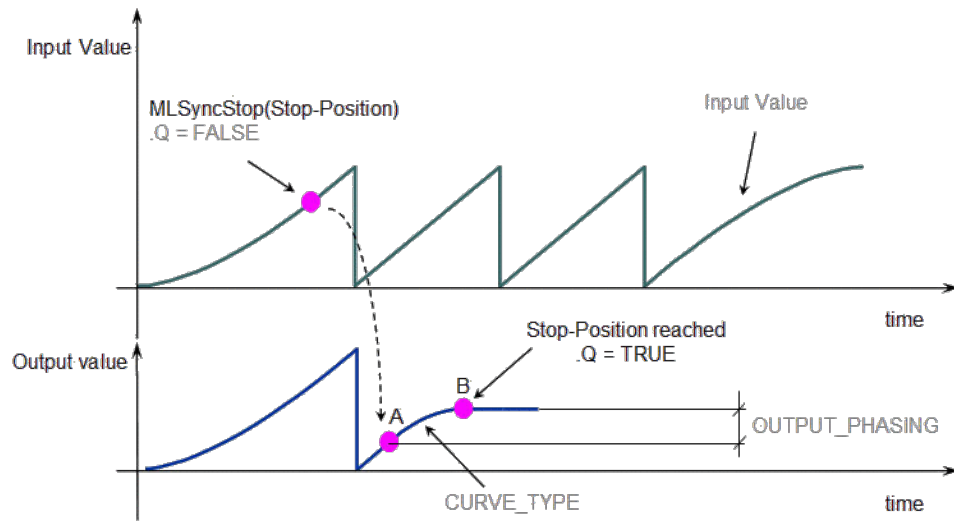


Figure 1-33: Set output phasing after MLSyncStop

Arguments

Input

B LockID	Description Data type Range Unit Default	Name of the Pipe Network Block DINT [-2147483648, 2147483648] n/a —
Delta S	Description Data type Range Unit Default	Slope to be used during Start and stop of Synchronization LREAL — User unit —

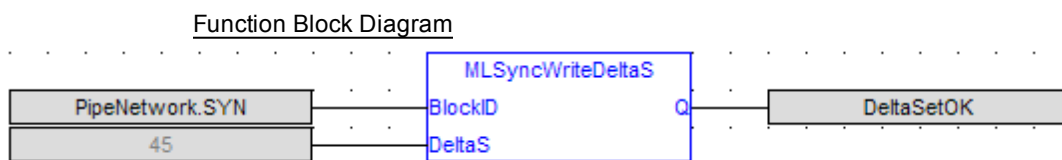
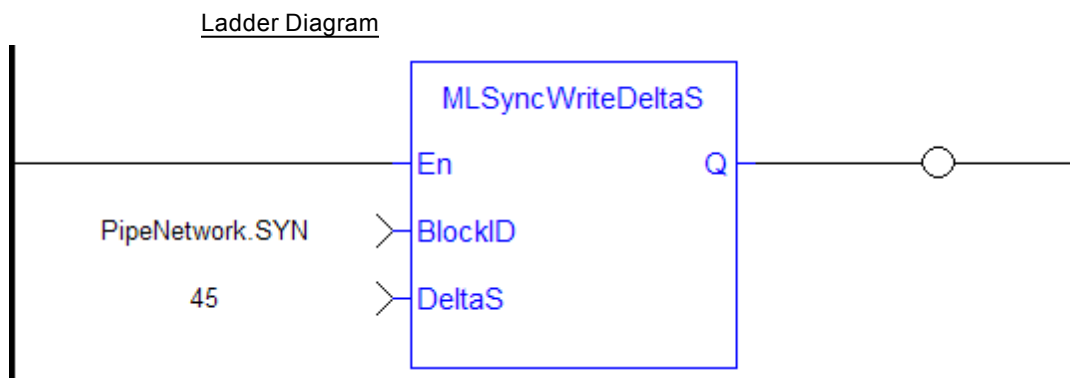
Output

Default (.Q)	Description Data type Unit	Function Block Execute Successfully BOOL n/a
---------------------	----------------------------------	--

Example

Structured Text

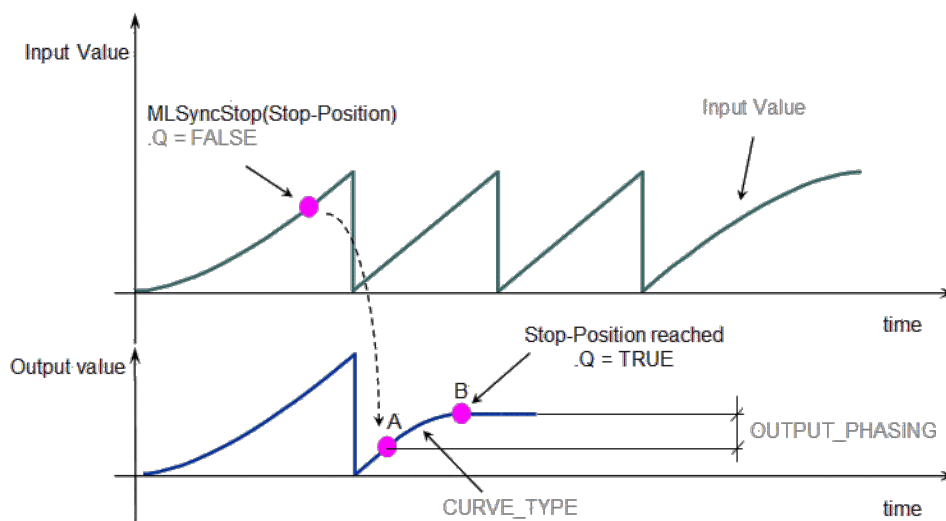
```
MLSyncWriteDeltaS( PipeNetwork.SYN, 45 );
```



1.1.17.6 Usage example of Synchronizer Functions

When you call the **MLSyncStop** function, 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.



When you call the **MLSyncStart** function, 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.

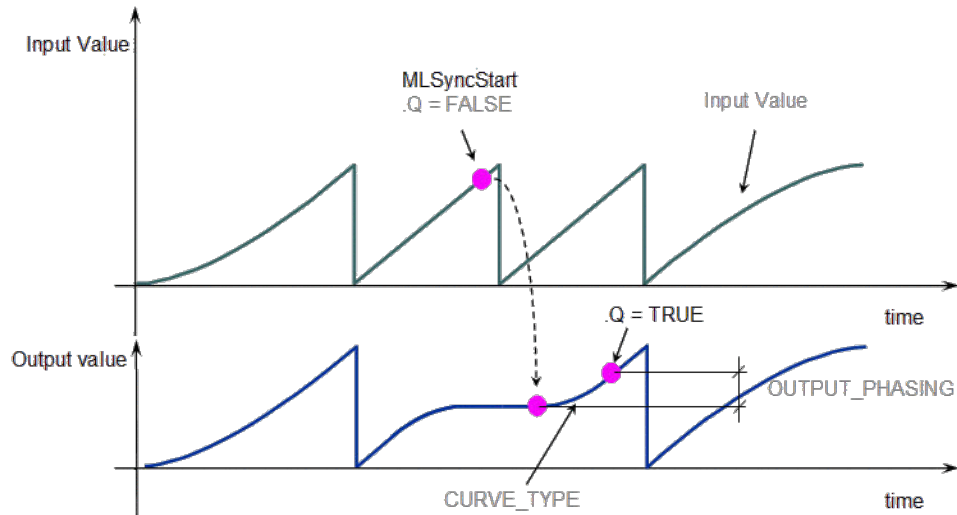


Figure 1-34: Synchronizer Functions Usage

1.1.18 Motion Library - Trigger

TIP For usage example about Trigger Functions, see page 154

Name	Description	Return type
MLTrigClearFlag	Clears the flag of an initiated Trigger block	BOOL
MLTrigInit	Initializes a Trigger object	BOOL
MLTrigIsTriggered	Checks if the selected block has been triggered	BOOL
MLTrigReadDelay	Returns the time that the trigger block uses to compensate the delay of the sensor that captures the triggering signal	None
MLTrigReadPos	Returns the position of the block at the moment when it was triggered	None
MLTrigReadTime	Returns the time of the moment where the block was triggered in milliseconds	None
"MLTrigSetEdge" (see page 152)	Sets the edge configuration for a Trigger object	BOOL
MLTrigWriteDelay	Sets the time that the trigger block uses to compensate for the delay introduced by the sensor that captures the triggering signal	BOOL

1.1.18.1 MLTrigClearFlag

Description

Clears the flag of an initiated Trigger block so the block can capture the position and time of the next event. Once triggered, a block has to be reset with this command before it can be triggered again. All events that are sent to a block while in a triggered state are ignored and the position and time information is lost.

IMPORTANT The Fast Input assigned to a Trigger block has to be reset as well before information on a new event can be captured. MLAxisRstFastIn is generally used at the same time as MLTrigClearFlag

Arguments

Input

<code>BlockID</code>	Description	ID number of an initiated Trigger object
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

<code>Default (.Q)</code>	Description	Returns TRUE if function block is executed
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

Related Functions

- MLAxisRstFastIn
- MLTrigsTriggered
- MLTrigReadPos
- MLTrigReadTime

Example

Structured Text

```
//Clear Trigger Flag
MLTrigClearFlag( PipeNetwork.TRIGGER );
```

Ladder Diagram



Function Block Diagram



1.1.18.2 MLTrigInit

Description

Initializes a Trigger object for use in a PLC Program. 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, and 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. Then you do not have to add MLTrigInit function blocks to their programs. Parameters are entered directly in pop-up windows, and the code is then automatically added to the current project.

Arguments

Input

BlockID	Description	ID number of a created Pipe Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
Input_Axis	Description	Name of the axis where the Fast Input is located
	Data type	STRING
	Range	—
	Unit	n/a
	Default	—
InputID	Description	ID number of the Fast Input 0 = Capture Engine 0 1 = Capture Engine 1 Range is [0,1] For information on configuring the capture engines, refer to AKD Capture Engine Configuration.
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
EdgeID	Description	Trigger at rising or falling edge of Fast Input. Enter 1 for rising edge, 2 for falling edge, and 0 disables the Fast Input
	Data type	DINT
	Range	[0 , 2]
	Unit	n/a
	Default	1 (Rising edge)

Output

Default (.Q)

Description	Returns TRUE if function block is executed
Data type	BOOL
Unit	n/a

Return Type

BOOL

Related Functions

MLTrigsTriggered

MLTrigReadPos

MLTrigClearFlag

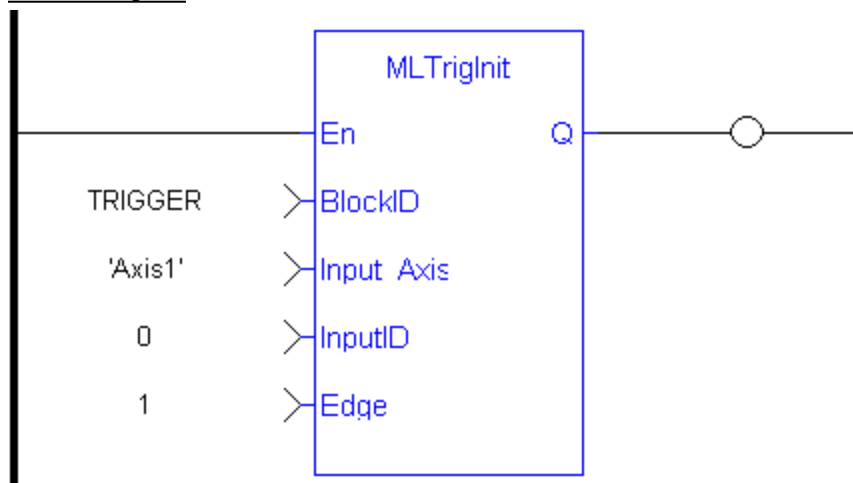
MLAxisRstFastIn

Example

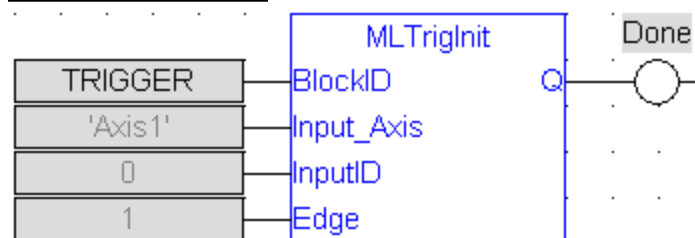
Structured Text

```
//Create and Initiate a Trigger object
TRIGGER := MBlkCreate( 'TRIGGER', 'TRIGGER' );
MLTrigInit( TRIGGER, 'Axis1', 0, 1 );
```

Ladder Diagram



Function Block Diagram



1.1.18.3 MLTrigsTriggered

Description

Checks if the selected block has been triggered. 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.

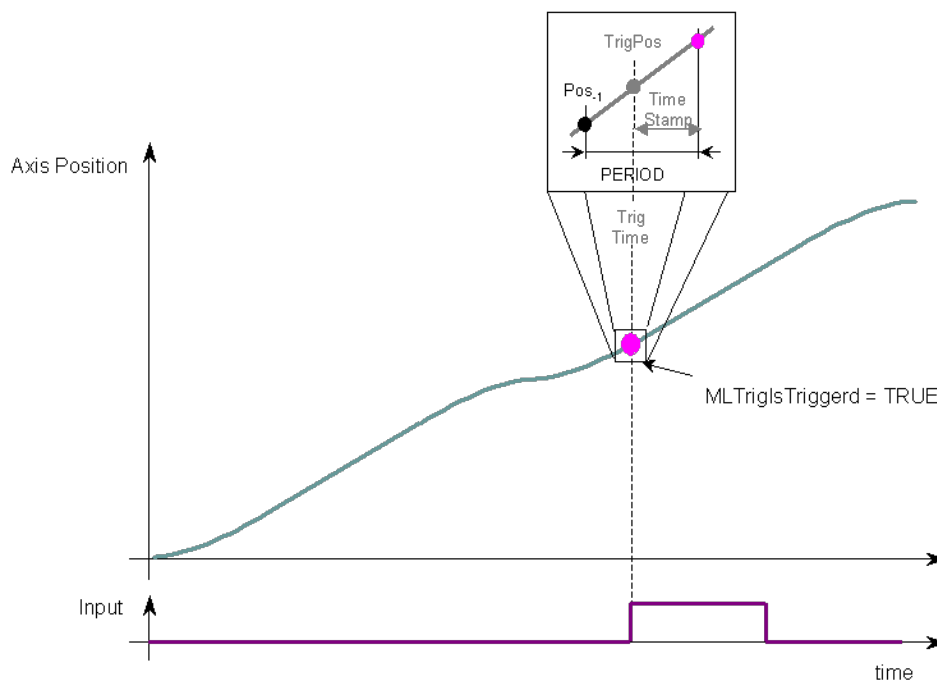


Figure 1-35: MLTrigsTriggered

NOTE Once triggered, a block has to be reset before it can be triggered again. All events that are sent to a block while in a triggered state are ignored and the position and time information is lost.

Arguments

Input

BlockID	Description	ID number of an initiated Trigger object
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Default (.Q)	Description	Returns TRUE if the selected Trigger Object has Triggered
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

Related Functions

MLTrigReadPos

MLTrigReadTime

Example

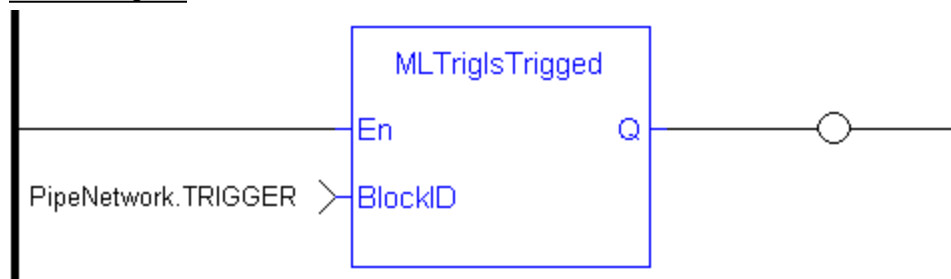
//Check if a Trigger Block has been triggered, then save position

IF MLTrigsTriggered(PipeNetwork.TRIGGER) THEN

Trig_Position := MLTrigReadPos(PipeNetwork.TRIGGER);

END_IF

Ladder Diagram



Function Block Diagram



1.1.18.4 MLTrigReadDelay

Description

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. This function returns the delay that has been programmed in a trigger block by the MLTrigWriteDelay function to compensate for this reaction time required by the sensor.

Input

BlockID	Description	Identifier of the trigger block whose delay is requested
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
En	Description	Enables execution
	Data type	BOOL
	Unit	n/a
	Default	-

<u>Output</u>		
Delay	Description	Value of the delay compensation currently applied by the trigger block
	Data type	LREAL
	Unit	microseconds
OK	Description	Returns true when the function successfully executes
	Data type	BOOL
	Unit	n/a

Related Functions

MLTrigWriteDelay

1.1.18.5 MLTrigReadPos

Description

Returns the position of the block at the moment when it was triggered by the Trigger Block's selected Fast Input. This value is only valid when TrigsTriggered() 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 that are sent to a block while in a triggered state are ignored and the position and time information is lost.

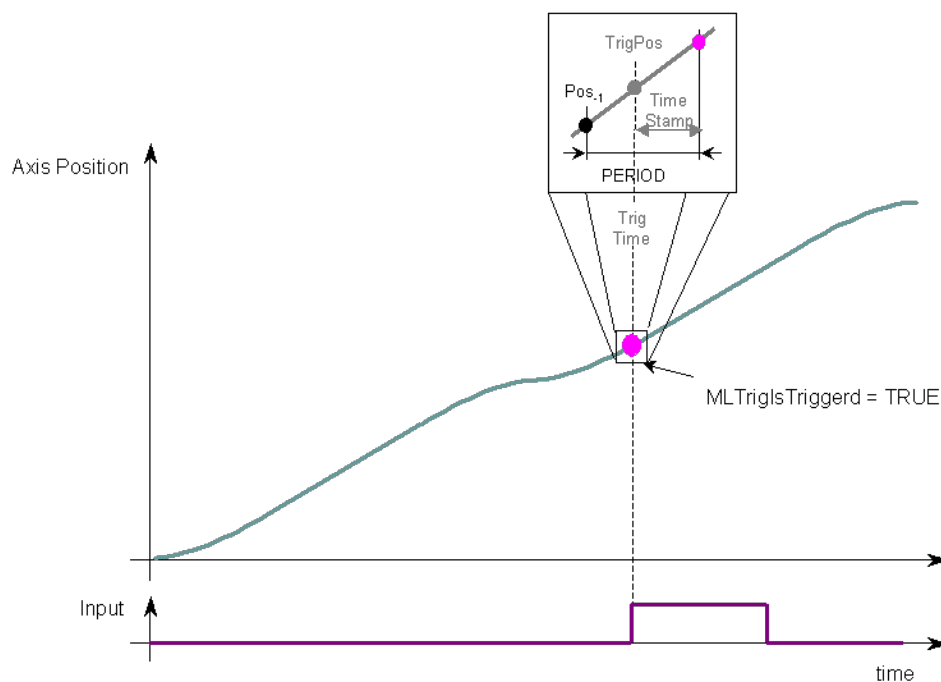


Figure 1-36: MLTrigReadPos

Arguments

Input

BlockID	Description	ID number of an initiated Trigger object
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Position	Description	Returns the position of the selected block's Axis at the moment when it was triggered
	Data type	LREAL
	Unit	User unit

Related Functions

MLTrigsTriggered

MLTrigReadTime

MLTrigClearFlag

MLAxisRstFastIn

Previous Function Name

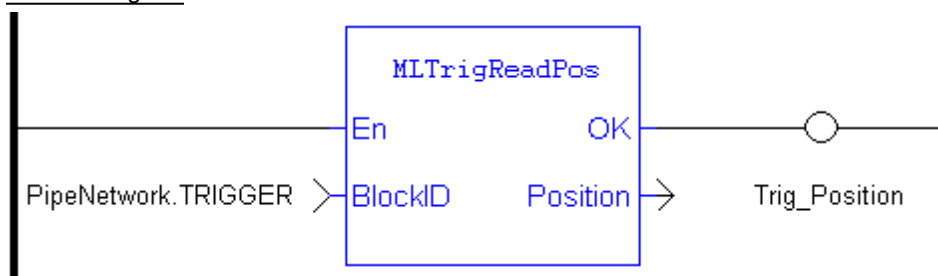
MLTrigGetPos

Example

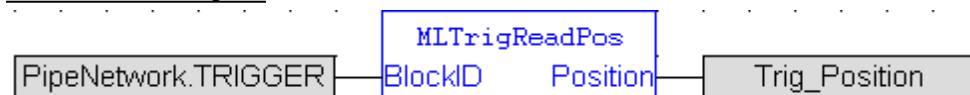
Structured Text

```
//Save position of Axis when Fast Input event occurs
Trig_Position := MLTrigReadPos( PipeNetwork.TRIGGER );
```

Ladder Diagram



Function Block Diagram



1.1.18.6 MLTrigReadTime

Description

Returns the time of the moment where the block was triggered in milliseconds. This value is only valid when TrigsTriggered() 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 that are sent to a block while in a triggered state are ignored and the position and time information is lost.

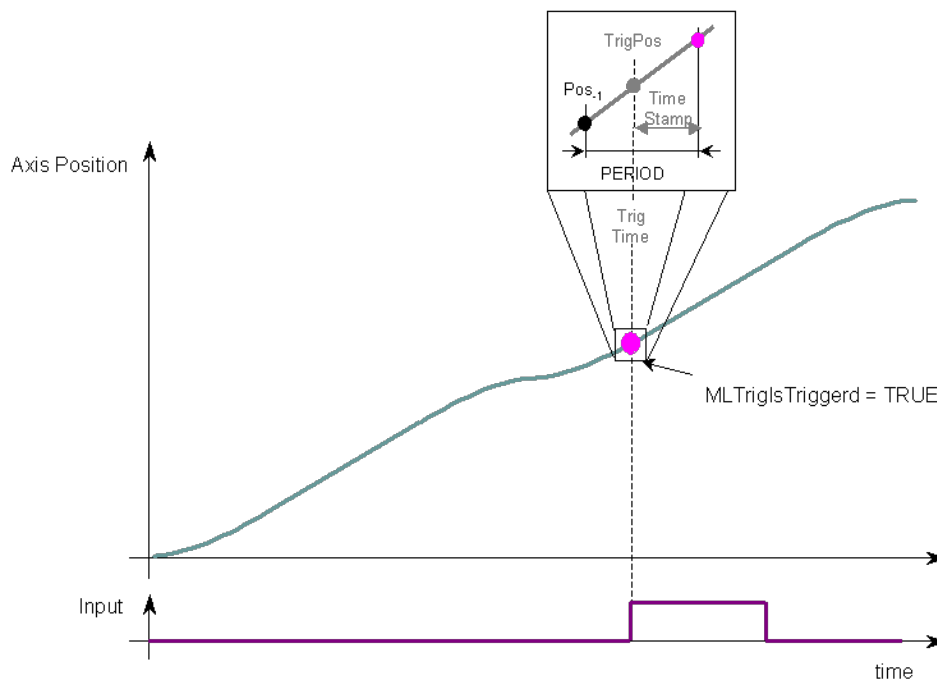


Figure 1-37: MLTrigReadTime

Arguments

Input

BlockID	Description	ID number of an initiated Trigger object
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Time	Description	Returns the time that the Trigger Block's selected Fast Input was triggered
	Data type	LREAL
	Unit	milliseconds

Related Functions

MLTrigsTriggered

MLTrigReadPos

MLTrigClearFlag

MLAxisRstFastIn

Previous Function Name

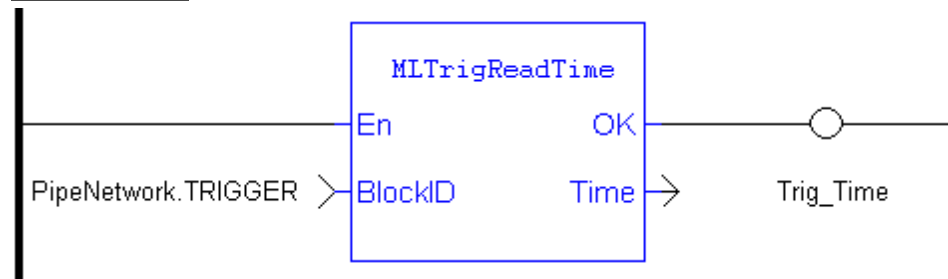
MLTrigGetTime

Example

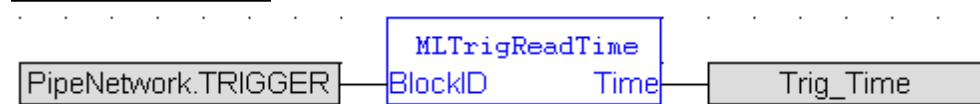
//Save time when Fast Input event occurs

Trig_Time := MLTrigReadTime(PipeNetwork.TRIGGER);

Ladder Diagram



Function Block Diagram



1.1.18.7 MLTrigSetEdge

Description

Sets the edge configuration (rising, falling, etc.) for a trigger block. This block should be called prior to calling MLAxisCfgFastIn. Also the value at the Edge input must match the value at MLAxisCfgFastIn's Mode input.

Arguments

Input

<u>BlockID</u>	Description Data type Range Unit Default	Identifier of the trigger block DINT [-2147483648, 2147483647] n/a —
<u>Edge</u>	Description Data type Range Unit Default	The edge on which to trigger 0 = disable the fast input 1 = rising edge 2 = falling edge DINT [0,2] n/a 1 (rising edge)

Output

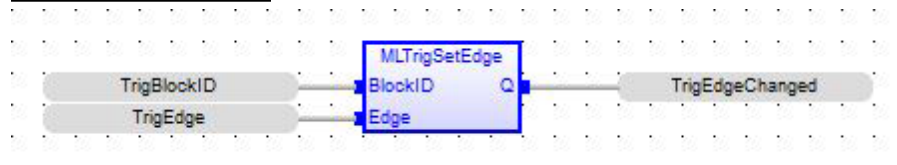
Q	Description	True if block executed successfully False if execution is not successful
	Data type	BOOL
	Unit	n/a

Return Type

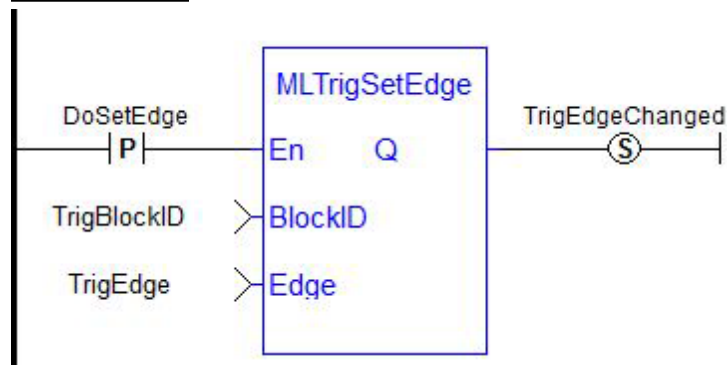
BOOL

Examples

Function Block Diagram



Ladder Diagram



Structured Text

```
TrigEdgeChanged := MLTrigSetEdge(TrigBlockID, TrigEdge);
```

1.1.18.8 MLTrigWriteDelay

Description

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. This function allows the trigger block to calculate the exact moment at which a signal was triggered by letting you specify the delay introduced by the sensor.

Input

BlockID	Description	Identifier of the trigger block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
Delay	Description	Reaction time of the sensor that the trigger block has to compensate
	Data type	LREAL
	Range	—
	Unit	microseconds
	Default	—

Output

Default (.Q)

Description	Returns TRUE if the delay is successfully set
Data type	BOOL
Unit	n/a

Return Type

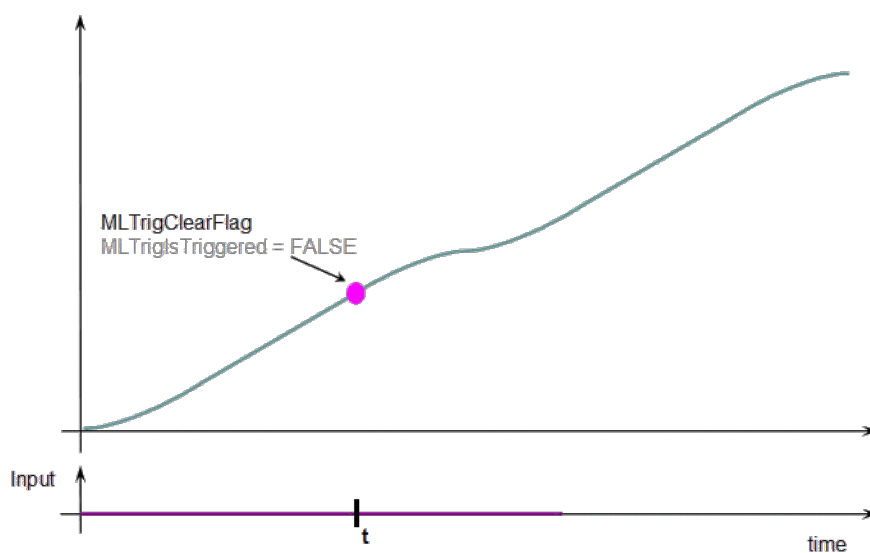
BOOL

Related Functions

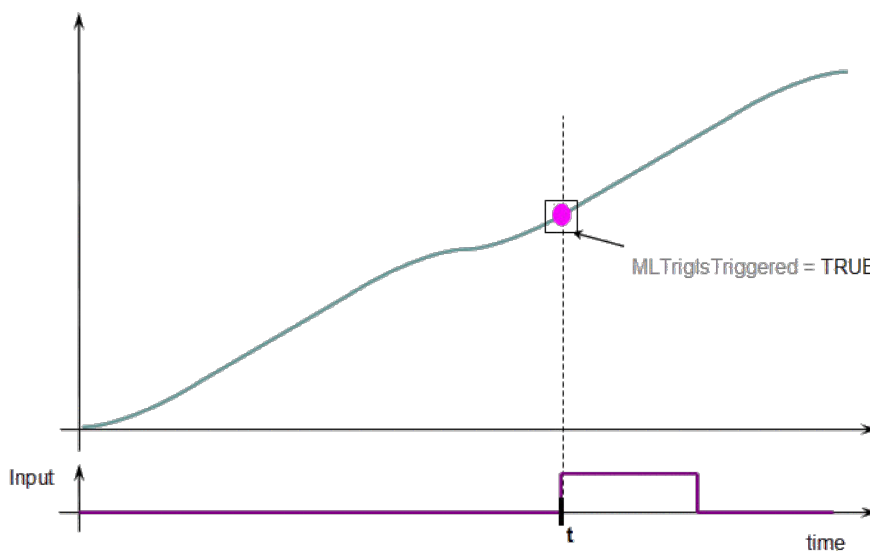
MLTrigReadDelay

1.1.18.9 Usage example of Trigger Functions

When you call the **MLTrigClearFlag** function, the flag for trigger is reset to False.



When a Fast Input is set, the **MLTrigsTriggered** function returns True.



Then you can call the **MLTrigReadPos** and **MLTrigReadTime** functions to get more details.

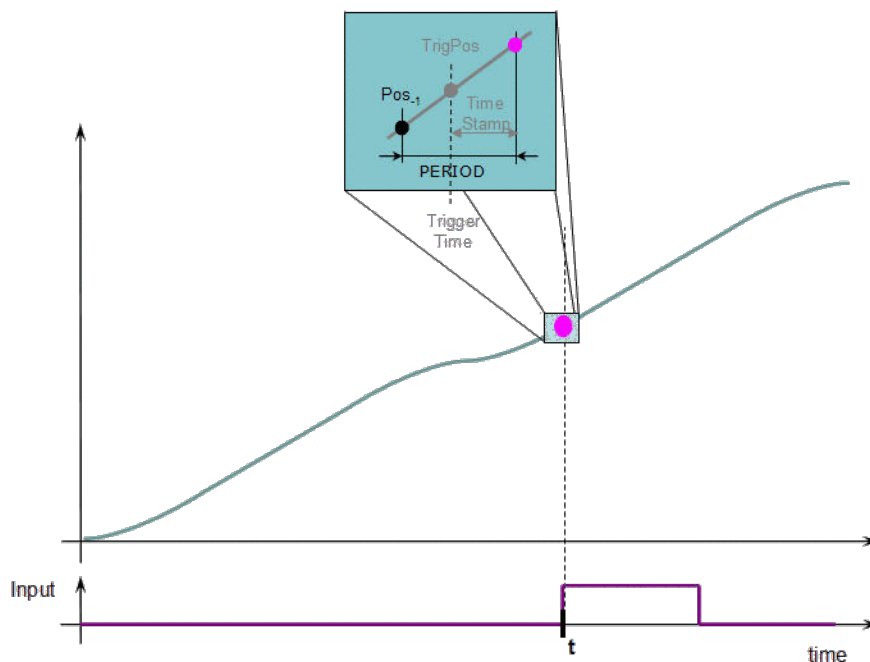


Figure 1-38: Trigger Functions Usage

ⓘ IMPORTANT The trigger delay has to be calculated by **you** and set with the MLTrigWriteDelay function block. This delay belongs to the sensor and it is additional to the MLTrigReadTime / MLTrigReadPos.

1.2 Motion Library / PLCopen

Functions sorted in alphabetical order:

Name	Description
MC_AbortTrigger	Abort MC_TouchProbe
MC_AddSuperAxis	Add an axis to the axis's list of assigned, superimposed axes.
MC_CamIn	Performs a slave axis move based on the Cam Table
MC_CamOut	Disengages the slave axis from a MC_CamIn move
MC_CamTblSelect	Defined to read and initialize the specified profile
MC_ClearFaults	Clear Drive Faults
MC_CreateAxis	Creates a PLCopen Axis
MC_EStop	Performs a Emergency stop
MC_ErrorDescription	Converts the PLCopen error IDs into message strings.
MC_GearIn	Performs a slave axis move based on the ratio
MC_GearInPos	Performs a slave axis move based on the ratio
MC_GearOut	Disengages the slave axis from a MC_GearIn or MC_GearInPos move
MC_Halt	Decelerates an axis to zero velocity
MC_InitAxis	Initializes a PLCopen Servo Axis' data
MC_MachRegist	Runs Mark-to-Machine registration

Name	Description
MC_MarkRegist	Runs Mark-to-Mark registration
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_MoveRelative" (see page 204)	Performs a single-axis move for a specified distance
MC_MoveSuperimp	Performs a single-axis move which is superimposed upon the active move
MC_MoveVelocity	Performs a single-axis non-ending move at a specified velocity
MC_Phasing	Performs a master position phase shift for the slave axis
MC_Power	Requests to enable the drive and close the loop, or disable the drive and open the loop
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_Reference	Defines the position at the reference location for PLCopen Axis
MC_RemSuperAxis	Remove an axis from the axis's list of assigned, superimposed axes.
MC_ResetError	Resets the errors of the specified axis
MC_SetOverride	Writes velocity and acceleration override factors
MC_SetPosition	writes a new axis position
MC_Stop	Aborts the active move, removes the next move from the queue, performs a controlled stop, and switches the axis to Stopping state
MC_StopRegist	Turns off registration for the specified axis
MC_SyncSlaves	Specifies synchronized slaves
MC_TouchProbe	Arm a Fast Input and capture an axis position
MC_WriteBoolPar	Writes the specified axis Boolean parameter
MC_WriteParam	Writes the specified axis parameter

1.2.1 Control

1.2.1.1 MC_ClearFaults (Function)

Description

MC_ClearFaults sends a request to the drive to clear any drive faults that exists.

NOTE

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.

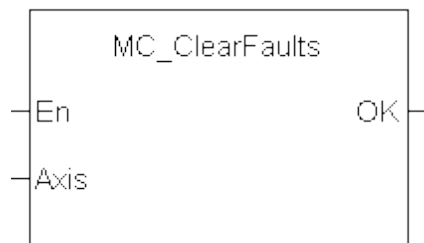


Figure 1-39: MC_ClearFaults

Arguments

Input

En	Description	Function enable – execute function. This Input must be on shot.
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	AXIS_REF.AXIS_NUM is the master axis number
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—

Output

OK	Description	Boolean output to indicate successful request. This output does not indicate that the fault are cleared, but simply indicates the request was made.
	Data type	BOOL

Usage

Upon the 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.

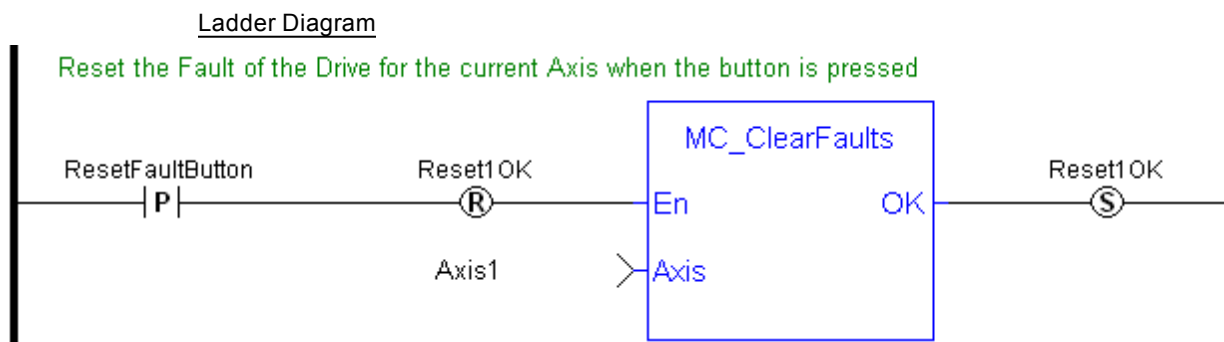
Related Functions

MC_ResetError

Example

Structured Text

```
(* MC_ClearFaults ST example *)
MC_ClearFaults( Axis1); //clear drive faults for Axis 1
```



1.2.1.2 MC_CreateAxis

Description

This function creates a PLCopen Axis. A call to this function is automatically generated when the application is compiled, based on the data entered in the PLCopen Axis Data dialog.

⚠ IMPORTANT MC_CreateAxis must be called between MLMotionInit and MLMotionStart.

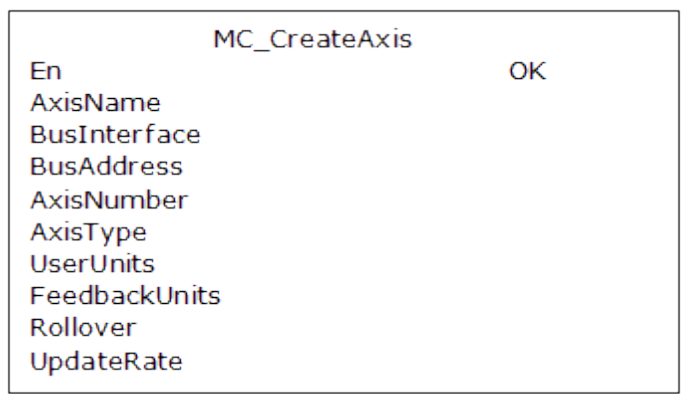


Figure 1-40: MC_CreateAxis

Arguments

Input

En	Description	Requests to create a PLCopen axis
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxisName	Description	Axis name
	Data type	STRING
	Range	—
	Unit	n/a
	Default	—

BusInterface	Description	Bus interface identifier: "EtherCATDriver" = EtherCAT interface "MSBusDriver" = KAS Simulator interface
	Data type	STRING
	Range	—
	Unit	n/a
	Default	—
BusAddress	Description	Address of the drive on the bus
	Data type	DINT
	Range	bus dependent
	Unit	n/a
	Default	—
AxisNumber	Description	Axis number
	Data type	UINT
	Range	[1,256]
	Unit	n/a
	Default	—
AxisType	Description	Axis type: 0 (MC_AXIS_TYPE_SERVO) denotes servo 1 (MC_AXIS_TYPE_DIGITIZING) denotes digitizing 2 (MC_AXIS_TYPE_VIRTUAL_SERVO) denotes virtual servo
	Data type	USINT
	Range	[0,1]
	Unit	n/a
	Default	—
UserUnits	Description	User unit portion of the user unit/feedback unit ratio
	Data type	UDINT
	Range	[1,4294967296]
	Unit	User unit
	Default	—
FeedbackUnits	Description	Feedback unit portion of the user unit/feedback unit ratio
	Data type	UDINT
	Range	[1,4294967296]
	Unit	feedback units. Note: <i>The FeedbackUnits input must be a power of 2. If input FeedbackUnits is not a power of two, the axis will not be created, and the OK output will be FALSE.</i>

Rollover	<p>Default —</p> <p>Description Rollover position (0 = no rollover)</p> <p>Data type UDINT</p> <p>Range [0, 4294967296]</p> <p>Unit User unit</p>
UpdateRate	<p>Default —</p> <p>Description 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</p> <p>Data type UINT</p> <p>Range [3,9]</p> <p>Unit n/a</p> <p>Default —</p>

Output

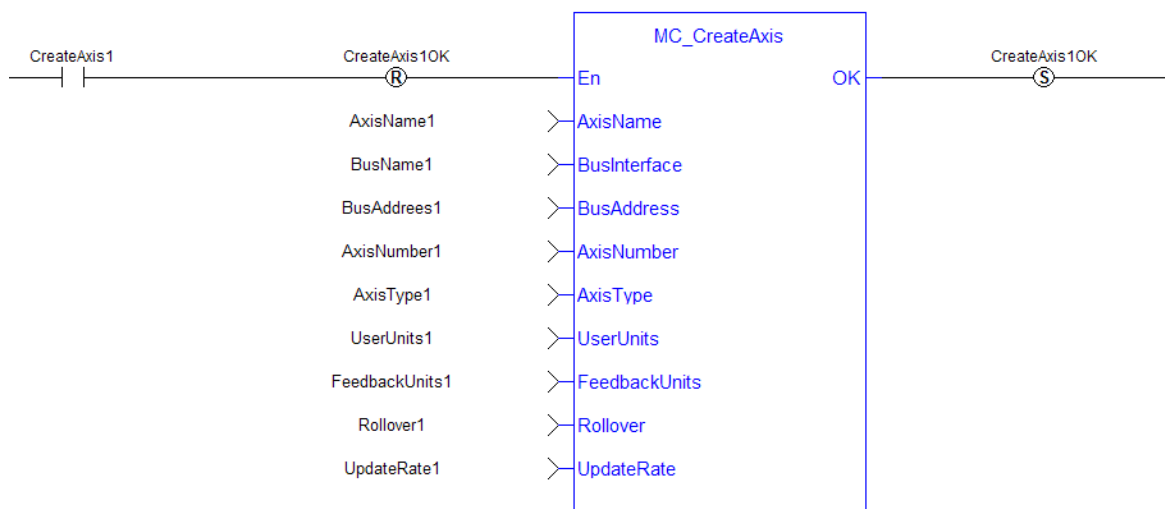
OK	<p>Description Indicates the axis has been created</p> <p>Data type BOOL</p>
-----------	--

Example

Structured Text

```
(* MC_CreateAxis ST example *)
MC_CreateAxis( 'PLCopenAxis1', 'EtherCATDriver', 1001, 1, MC_AXIS_
TYPE_SERVO, 360, 1048576, 0, 3 );
```

Ladder Diagram



1.2.1.3 MC_EStop

Description

This function causes an emergency stop (E-stop). An E-stop stops motion interpolation, clear all moves from the queue (active and next), change the axis state to ErrorStop, and request the drive to open the position loop and disable the drive. The E-stop remains in effect until the application calls MC_ResetError to reset the E-stop.

**Figure 1-41:** MC_EStop**Arguments**Input

En	Description	A positive transition of this input causes an E-stop on the specified axis
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	Axis identifier
	Data type	AXIS_REF
	Range	1-256 The AXIS_NUM element of the AXIS_REF structure must be in the range [1-256]
	Unit	n/a
	Default	—

Output

OK	Description	Indicates the E-stop was executed. If an invalid Axis input was specified, this output is not energized and no E-stop is performed.
	Data type	BOOL

Usage

Call MC_EStop to generate an emergency stop for an axis.

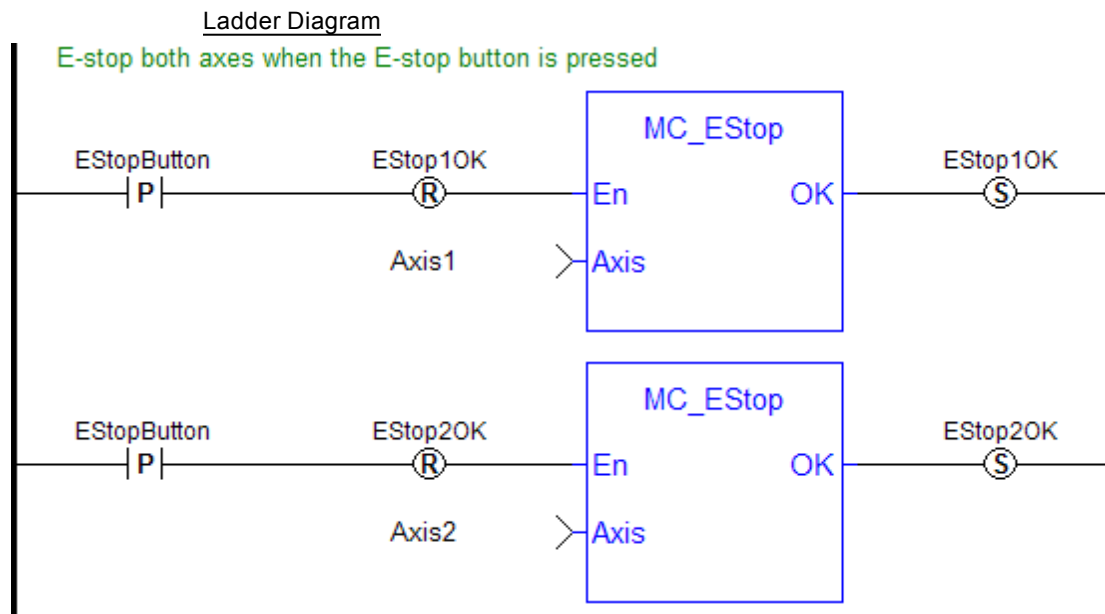
Call MC_ResetError to reset the emergency stop.

Related Functions

MC_ResetError

ExampleStructured Text

```
(* MC_Estop ST example *)
MC_EStop( Axis1 ); //E-Stop Axis 1
```



1.2.1.4 MC_InitAxis (Function)

Description

MC_InitAxis initializes a PLCopen Servo Axis' data. A call to this function is automatically generated when the application is compiled, based on the data entered in the PLCopen Axis Data dialog.

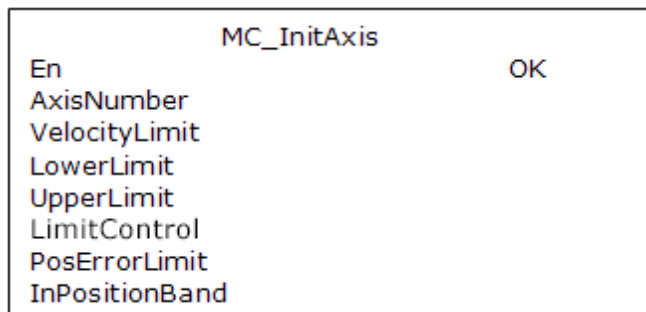


Figure 1-42: MC_InitAxis

Arguments

Input

En	Description Request to initialize a PLCopen servo axis Data type BOOL Range 0, 1 Unit n/a Default —	
AxisNumber	Description Servo axis number Data type UINT Range [1,256] Unit none Default —	

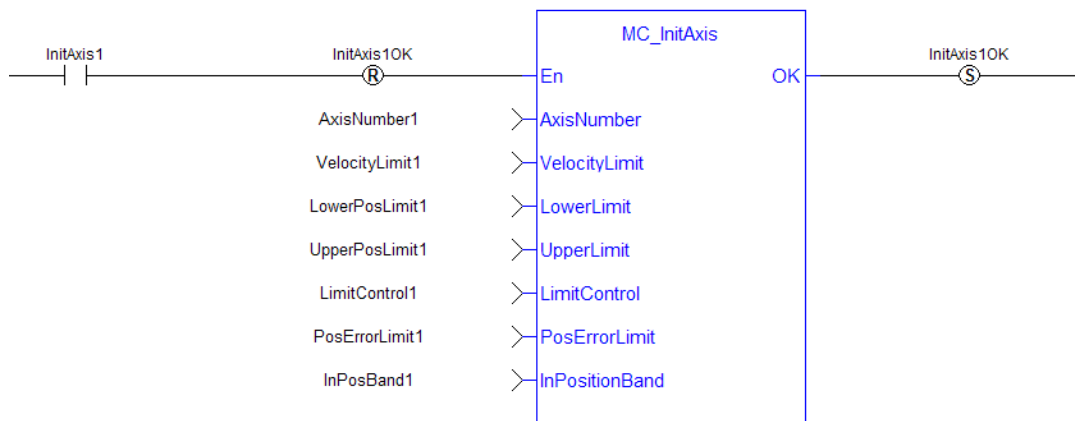
VelocityLimit	Description	Velocity limit
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
LowerLimit	Description	Lower position limit
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
UpperLimit	Description	Upper position limit
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
LimitControl	Description	Establishes how position limits are applied 0 = apply position limits 1 = ignore position limits 2 = ignore limits until referenced
	Data type	UINT
	Range	[0,2]
	Unit	n/a
	Default	—
PosErrorLimit	Description	Position error limit – when the Position Error (command position – actual position) exceeds this value, an E-stop is generated
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
InPositionBand	Description	In-position bandwidth – when the axis actual position is within this distance from its programmed endpoint, the axis is considered “in position”
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
<u>Output</u> OK	Description	Indicates the initialization is complete
	Data type	BOOL

Example

Structured Text

```
(* MC_InitAxis ST example *)
MC_InitAxis( 1, 0, 0, 0, 2, 0, 0 );
```

Ladder Diagram



1.2.1.5 MC_Power

Description

This function block requests to enable the drive and close the position loop, or disable the drive and open the position loop. 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.

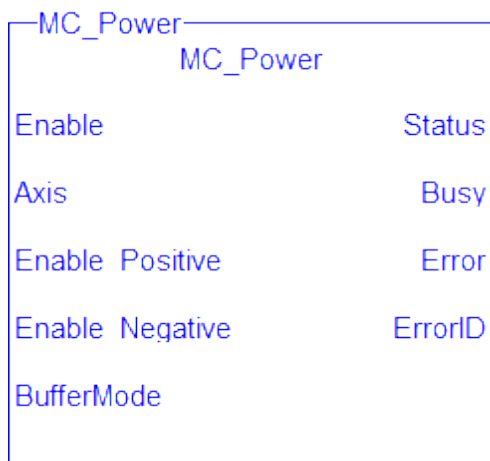


Figure 1-43: MC_Power

NOTE

You must be careful if you have 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 of your active programs.

Arguments

Input

Enable	Description	When this transitions go to high, the control closes the servo loop and sends a command to the drive to enable. When this transitions go to low, the control opens the servo loop and sends a command to the drive to disable.
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	Name of a declared instance of the AXIS_REF library function.
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Enable Positive	Description	<i>for future enhancement</i>
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Enable Negative	Description	<i>for future enhancement</i>
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
BufferMode	Description	Unused
	Data type	SINT
	Range	[0]
	Unit	n/a
	Default	—

Output

Status	Description	Indicates the enabled/disabled state of the drive
	Data type	BOOL
Busy	Description	for future enhancement – always false
	Data type	BOOL
Error	Description	Indicates an invalid input was specified
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT

Example

Structured Text

```
(* MC_Power ST example *)

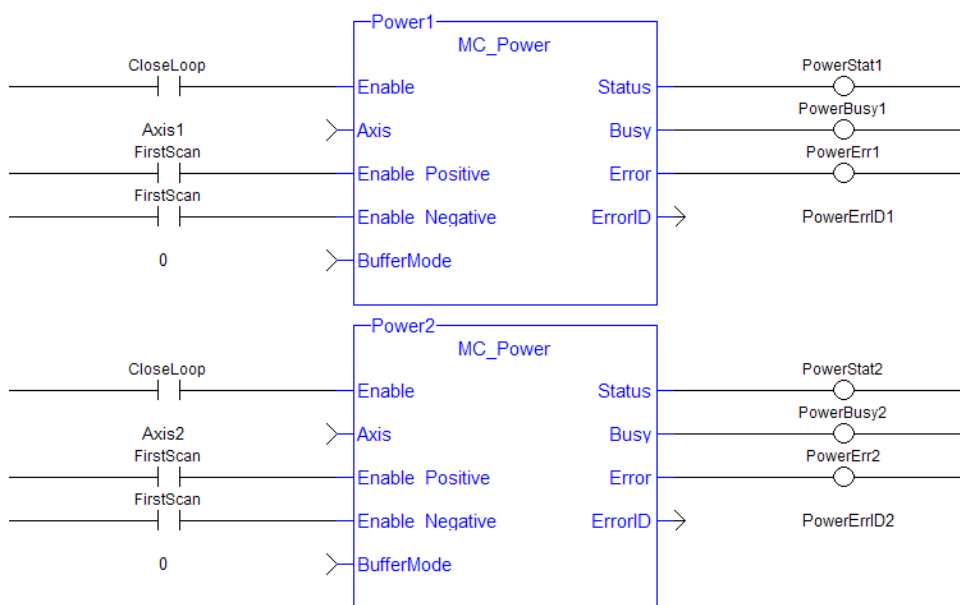
Inst_MC_Power( CloseLoopReq, Axis1, TRUE, TRUE, 0 );

//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
```

Ladder Diagram

Close the servo loop and enable the drive when CloseLoop is high.
Open the servo loop and disable the drive when CloseLoop is low.



1.2.1.6 MC_ResetError

Description

The function MC_ResetError resets the errors of a specified axis.

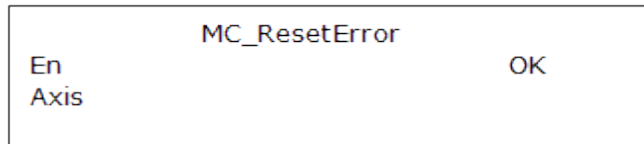
This function performs in sequence the following tasks:

- It sends a request to the drive to clear any drive faults that exists
- Then it 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 also transition 15 in the status machine of the CANopen protocol.

**Figure 1-44:** MC_ResetError**Arguments****Input**

En	Description	Requests to reset the axis errors
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	Name of a declared instance of the AXIS_REF library function)
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—

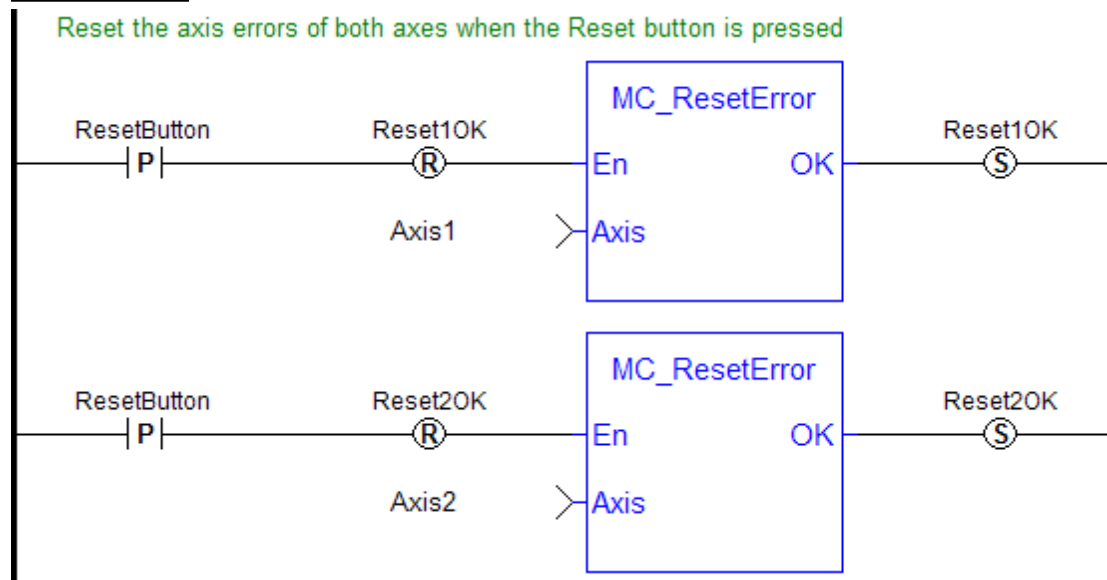
Output

OK	Description	Indicates the function has completed successfully
	Data type	BOOL

Example**Structured Text**

```
//reset the axis and drive errors for Axis 1
MC_ResetError( Axis1 );
```

Ladder Diagram



1.2.1.7 MC_Stop (Function Block)

Description

This function block aborts the active move, removes the next move from the queue, performs a controlled stop at the specified deceleration rate, and switches the axis to Stopping state.

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.

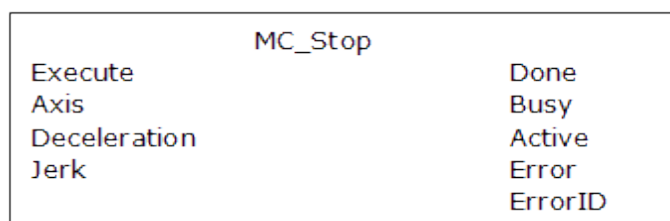
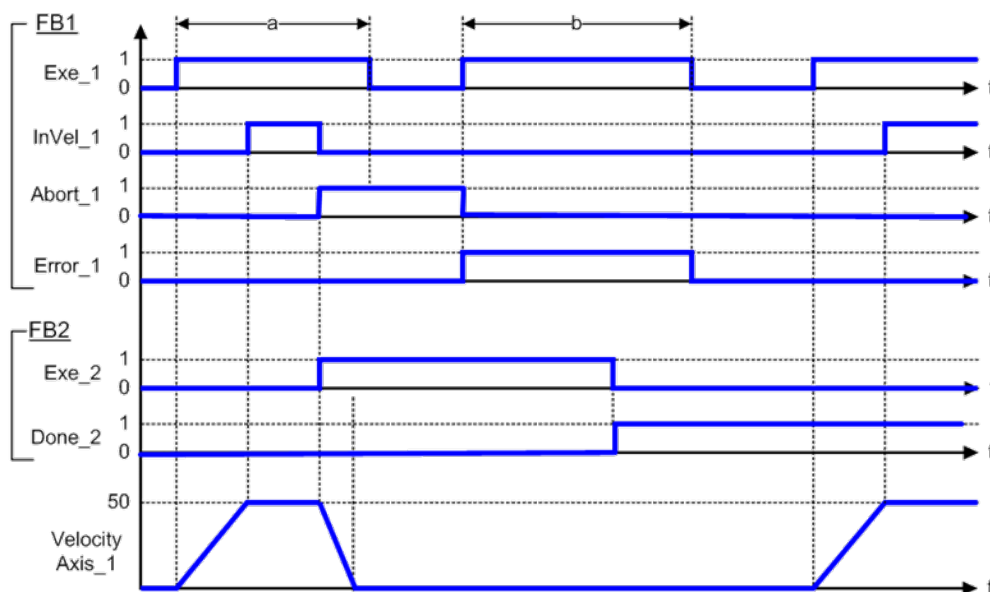
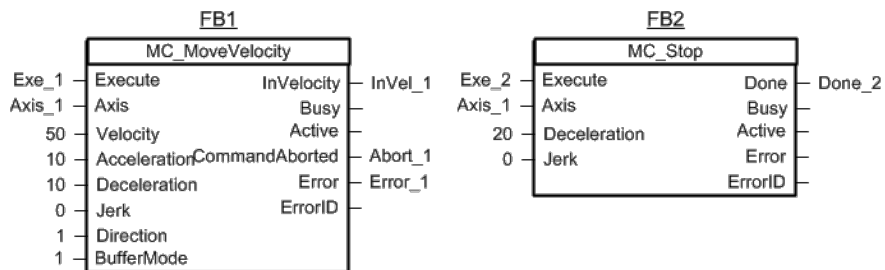


Figure 1-45: MC_Stop

Time Diagram

The example below 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.



Arguments

Input

Execute	Description	Requests to stop the axis. It can be held high to prevent any other moves from being queued
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	Name of a declared instance of the AXIS_REF library function.
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Deceleration	Description	Trapezoidal: Deceleration rate S-curve: Maximum deceleration
	Data type	REAL
	Range	—
	Unit	User unit/sec ²
	Default	—

Jerk	Description	Trapezoidal: 0 S-curve: Constant jerk
	Data type	REAL
	Range	—
	Unit	User unit/sec ³
	Default	—
<u>Output</u>		
Done	Description	Indicates the axis has reached zero velocity AND the Execute input is low
	Data type	BOOL
Busy	Description	High from the time the Execute input goes high until the axis reaches zero velocity AND the Execute input is low
	Data type	BOOL
Active	Description	High from the time the MC_Stop move becomes the active move, until the axis reaches zero velocity AND the Execute input is low
	Data type	BOOL
Error	Description	Indicates an invalid input was specified
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT

Example**Structured Text**

```
(* MC_Stop ST example *)

Inst_MC_Stop( StopRequest , Axis1, 100.0, 100.0 ); //Inst_
MC_Stop is an instance of MC_Stop function block

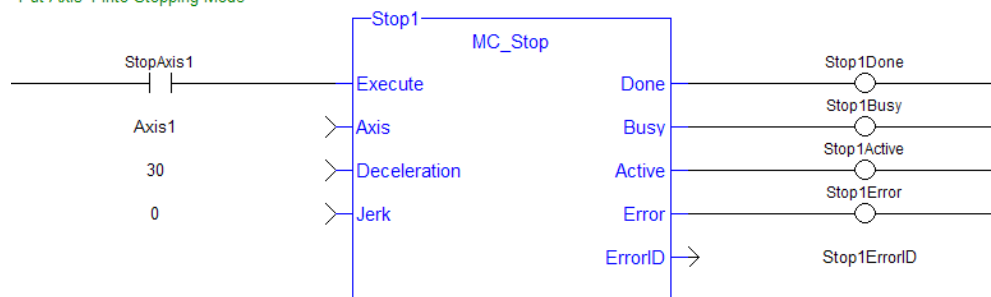
StopComplete := Inst_MC_Stop.Done; //store the Done output
into a user defined variable

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
```

Ladder Diagram

Put Axis 1 into Stopping Mode



1.2.2 I/O

1.2.2.1 MC_AbortTrigger (Function Block)

Description

When the Execute input transitions from low to high, this function block aborts an MC_TouchProbe function block.

Arguments

Input

Execute	Description Data type Range Unit Default	Enables execution BOOL 0, 1 n/a —
Axis	Description Data type Range Unit Default	Specifies the axis that was specified in the MC_TouchProbe function block which is to be aborted AXIS_REF [1,256] n/a —

TriggerInput	Description	<p>Specifies the Fast Input that was specified in the MC_TouchProbe function block which is to be aborted. The elements of TriggerInput are as follows:</p> <p>INT TriggerInput.InputID 0 = Capture Engine 0 1 = Capture Engine 1 Range is [0,1] For information on configuring the capture engines, refer to AKD Capture Engine Configuration.</p> <p>INT TriggerInput.Direction 1 = rising edge 2 = falling edge Range is [1,2]</p> <p>INT TriggerInput.TrigID is the axis number of the input. 0 indicates that the trigger axis is to be the same as Axis.AXIS_NUM. Range is [0,256]</p>
	Data type	TRIGGER_REF
	Range	See Description above
	Unit	n/a
	Default	—
<u>Output</u>		
Done	Description	Function block has completed
	Data type	BOOL
Busy	Description	Indicates the function block is currently executing
	Data type	BOOL
Error	Description	Indicates the function block did not complete due to an error. The ErrorID output indicates the type of error when this output is high
	Data type	BOOL
ErrorID	Description	When the Error output is high, this output indicates the type of error. When the Error output is low, this output is undefined
	Data type	INT

Usage

This function block is used to abort an MC_TouchProbe function block.

Related Functions

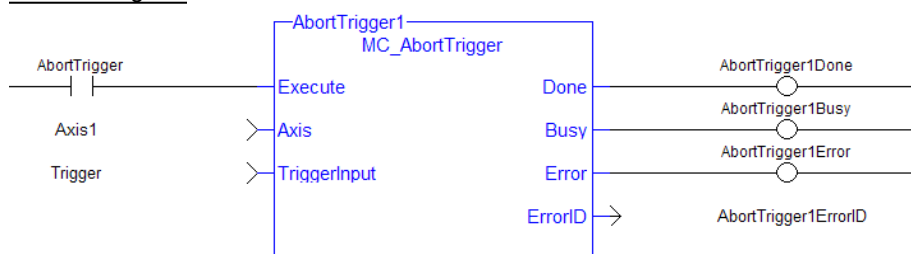
MC_TouchProbe

Example

Structured Text

```
(* MC_AbortTrigger ST example *)
Inst_MC_AbortTrigger( AbortReq, Axis1, TriggerInputRef );
//Inst_MC_AbortTrigger is an instance of MC_AbortTrigger
```

Ladder Diagram



1.2.2.2 MC_TouchProbe (Function Block)

Description

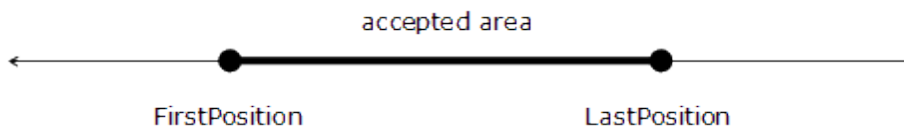
This function block arms a Fast Input and returns the latched position when the Fast Input event occurs. This function block causes no motion.

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.

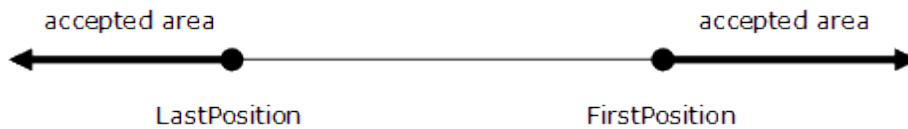
If First Position \leq LastPosition, the window in which a Fast Input is accepted is:

FastInputPosition \geq FirstPosition AND FastInputPosition \leq LastPosition.



If First Position $>$ LastPosition, the window in which a Fast Input is accepted is:

FastInputPosition \geq FirstPosition OR FastInputPosition \leq LastPosition.



The following figure shows the ladder diagram view of the MC_TouchProbe function block:

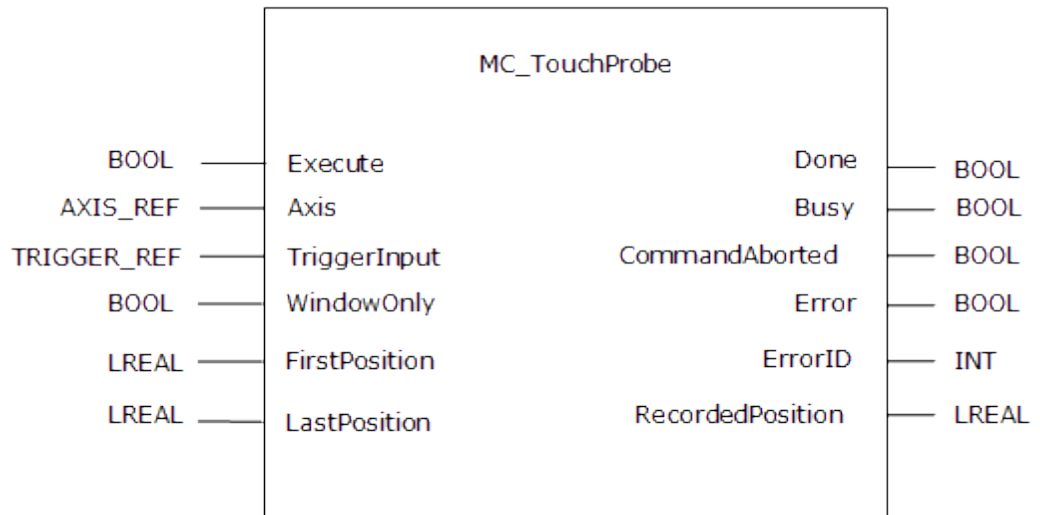


Figure 1-46: MC_TouchProbe

Arguments

Input

Execute	Description Data type Range Unit Default	Enables execution BOOL 0, 1 n/a —
Axis	Description Data type Range Unit Default	Selects the axis for which the position is latched AXIS_REF [1,256] n/a —

TriggerInput	Description	<p>Selects the axis which contains the specified input to be armed. The elements of TriggerInput are as follows:</p> <p>INT TriggerInput.InputID 0 = Capture Engine 0 1 = Capture Engine 1 Range is [0,1] For information on configuring the capture engines, refer to AKD Capture Engine Configuration.</p> <p>INT TriggerInput.Direction 1 = rising edge 2 = falling edge Range is [1,2]</p> <p>INT TriggerInput.TrigID is the axis number of the input. 0 indicates that the trigger axis is to be the same as Axis.AXIS_NUM. Range is [0,256]</p>
	Data type	TRIGGER_REF
	Range	See Description above
	Unit	n/a
	Default	—
WindowOnly	Description	<p>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</p>
	Data type	BOOL
	Range	—
	Unit	n/a
	Default	—
FirstPosition	Description	<p>See the function block Description above for an explanation of 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</p>
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

LastPosition	Description	See the function block Description above for an explanation of 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
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

Done	Description	Function block has completed and the RecordedPosition output is valid
	Data type	BOOL
Busy	Description	Indicates that the specified input is arming or is armed, and waiting for the trigger and recording of the position to occur
	Data type	BOOL
CommandAborted	Description	A TriggerAbort function block has executed and canceled this function
	Data type	BOOL
Error	Description	The function block has not completed successfully due to an error. The ErrorID output indicates the type of error
	Data type	BOOL
ErrorID	Description	When the Error output is high, this output indicates the type of error. When the Error output is low, this output is undefined
	Data type	INT
RecordedPosition	Description	When the Done output goes high, this output returns the latched position. When the Done output is low, this output is undefined
	Data type	LREAL
	Unit	User unit

Usage

This function block can be used to:

- Perform registration
- Determine the position of a product
- Measure product length

Related Functions

MC_AbortTrigger

Example

Structured Text

```
(* MC_TouchProbe ST example *)

TriggerInputRef.InputID := 1; //configure InputID

TriggerInputRef.Direction := 1; //configure Direction

TriggerInputRef.TrigID := 0; //configure TrigID

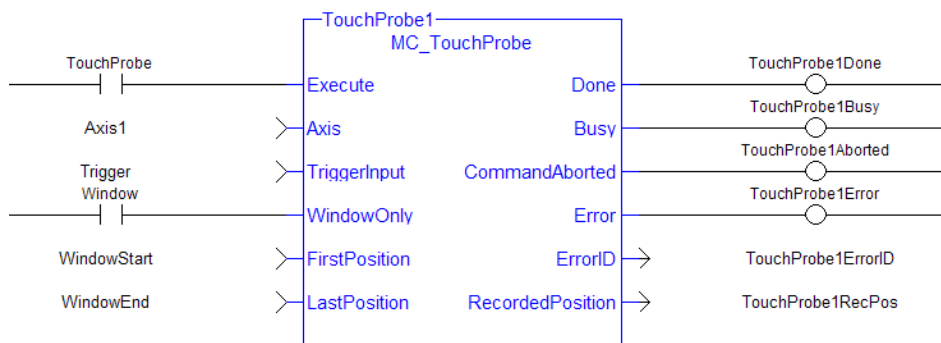
Inst_MC_TouchProbe( ArmProbe, Axis1, TriggerInputRef,
FALSE,0.0, 0.0 );

//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
```

Ladder Diagram



1.2.3 Info

1.2.3.1 MC_ReadActPos (Function Block)

Description

MC_ReadActPos reads the actual position of the axis.

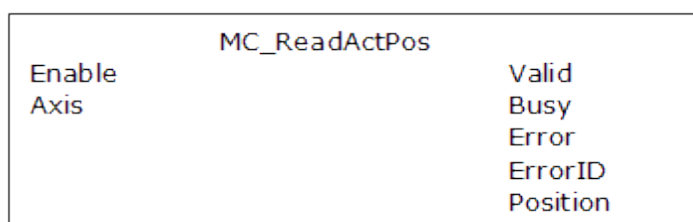


Figure 1-47: MC_ReadActPos

Arguments**Input**

Enable	Description	Request to read the axis's actual position Keeps continuously to read the actual position every PLC cycle, as long as the Enable remains high
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	Name of a declared instance of the AXIS_REF library function.)
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—

Output

Valid	Description	Indicates the value at the Position output is available
	Data type	BOOL
Busy	Description	Indicates this function block is executing
	Data type	BOOL
Error	Description	Indicates an invalid input
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT
Position	Description	Actual position of the axis.
	Unit	User unit
	Data type	LREAL

Example**Structured Text**

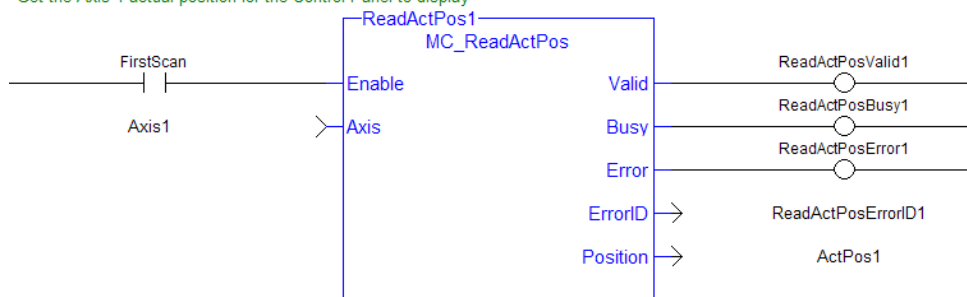
```
(* MC_ReadActPos ST example *)
Inst_MC_ReadActPos( TRUE, Axis1 );

//Inst_MC_ReadActPos is an instance of MC_ReadActPos
function block

ActualPos := Inst_MC_ReadActPos.Position; //store Position
output into a user defined variable
```

Ladder Diagram

Get the Axis 1 actual position for the Control Panel to display



1.2.3.2 MC_ReadActVel (Function Block)

Description

MC_ReadActVel reads the actual velocity of the axis.

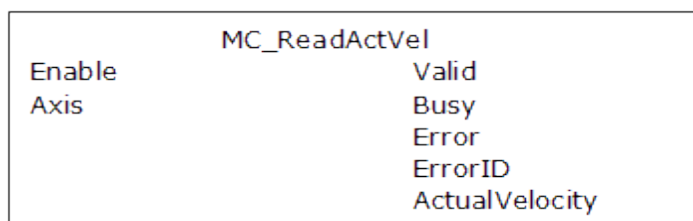


Figure 1-48: MC_ReadActVel

Arguments

Input

Enable	Description	Requests to read the axis's actual velocity
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	Name of a declared instance of the AXIS_REF library function.
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—

Output

Valid	Description	Indicates the value at the ActualVelocity output is available
	Data type	BOOL
Busy	Description	Indicates this function block is executing
	Data type	BOOL
Error	Description	Indicates an invalid input
	Data type	BOOL

ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT
ActualVelocity	Description	Actual velocity ¹ of the axis.
	Unit	User unit/sec
	Data type	LREAL

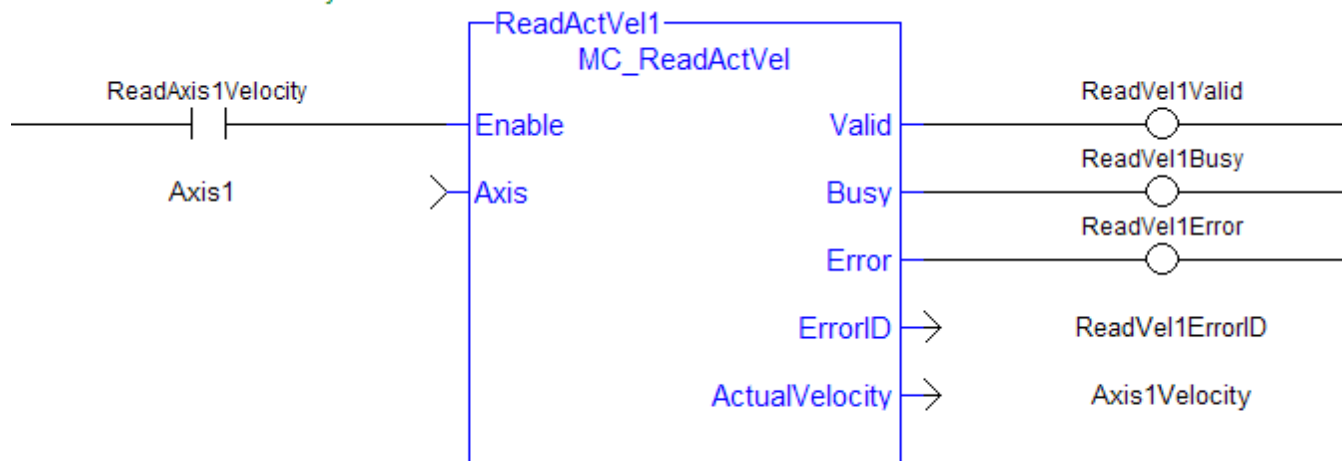
Example

Structured Text

```
(* MC_ReadActVel ST 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
```

Ladder Diagram

Read Axis 1 actual velocity



1.2.3.3 MC_ReadAxisErr

Description

The Function Block MC_ReadAxisErr returns the error status of the specified axis.

¹The measured value is the instant velocity of the axis in RPM*1000. Note that you can see some oscillations because it is an instant velocity, not an average velocity.

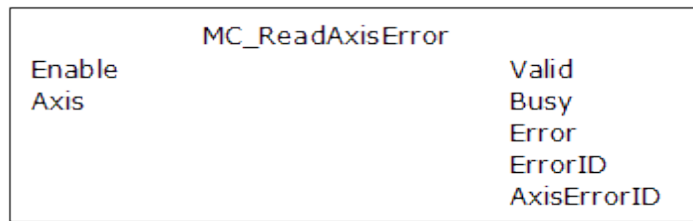


Figure 1-49: MC_ReadAxisErr

Arguments

Input

Enable	Description	requests to read the error status of the axis
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	Name of a declared instance of the AXIS_REF library function.
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—

Output

Valid	Description	Indicates the AxisErrorID output is valid
	Data type	BOOL
Busy	Description	Indicates this function block is executing
	Data type	BOOL
Error	Description	Indicates an invalid input
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT
AxisErrorID	Description	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 table below defines the bits of this output.
	Data type	INT

Hexadecimal	Decimal	Description
0000H	0	No Error
0001H	1	User-set E-stop via MC_EStop, E-stop

Hexadecimal	Decimal	Description
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

NOTE Multiple errors can be active at the same time. For example, if a User-set E-stop and an Excess Position Error E-stop are both active, the value would be 00000011H (17 decimal).

Example

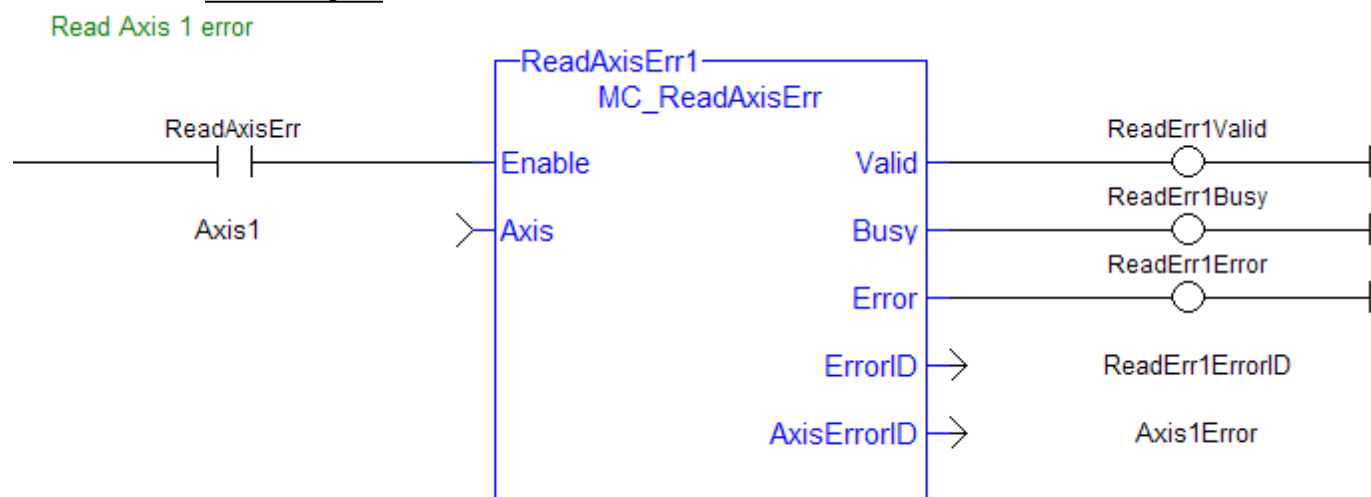
Structured Text

```
(* 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
```

Ladder Diagram



1.2.3.4 MC_ReadBoolPar (Function Block)

Description

MC_ReadBoolPar returns the value of the specified Boolean axis parameter.

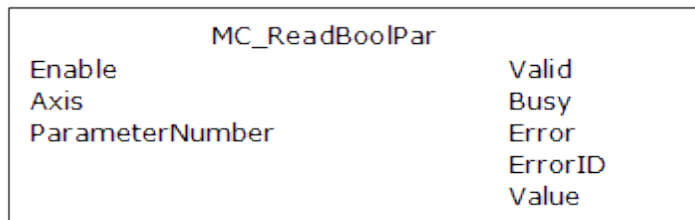


Figure 1-50: MC_ReadBoolPar

Arguments

Input

Enable	Description	Requests to read the Boolean axis parameter
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	Name of a declared instance of the AXIS_REF library function.)
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
ParameterNumber	Description	Parameter number, see table in § "Motion Library (PLCopen)"
	Data type	INT
	Range	—
	Unit	n/a
	Default	—

Output

Valid	Description	Indicates the Value output is valid
	Data type	BOOL
Busy	Description	Indicates this function block is executing
	Data type	BOOL
Error	Description	Indicates an invalid input
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT
Value	Description	State of the Boolean parameter
	Data type	BOOL

Example

Structured Text

```
(* MC_ReadBoolPar ST example *)

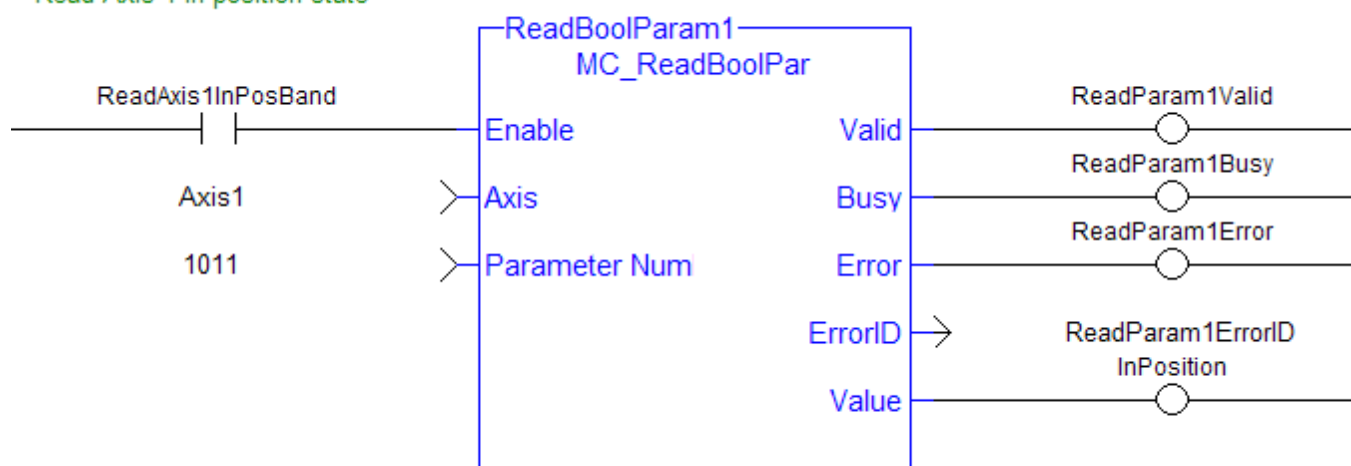
Inst_MC_ReadBoolPar( EnableRead, Axis1, 3 );

//Inst_MC_ReadBoolPar is an instance of MC_ReadBoolPar
function block

BoolParm := Inst_MC_ReadBoolPar.Value; //store the Value
output into a user defined variable
```

Ladder Diagram

Read Axis 1 in-position state



1.2.3.5 MC_ReadParam (Function Block)

Description

MC_ReadParam returns the value of the specified axis parameter.

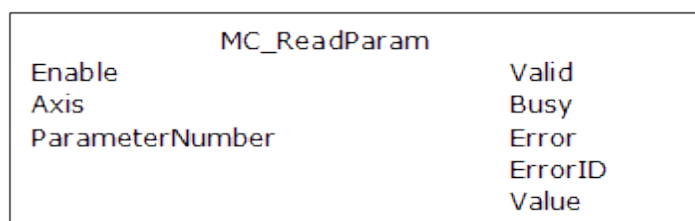


Figure 1-51: MC_ReadParam

Arguments

Input

<p>Enable</p>	<p>Description Requests to read the axis parameter</p> <p>Data type BOOL</p> <p>Range 0, 1</p> <p>Unit n/a</p> <p>Default —</p>
---------------	--

Axis	Description	Name of a declared instance of the AXIS_REF library function.)
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
ParameterNumber	Description	Parameter number, see table in § "Axis Parameters"
	Data type	INT
	Range	—
	Unit	n/a
	Default	—
 <u>Output</u>		
Valid	Description	Indicates the Value output is valid
	Data type	BOOL
Busy	Description	Indicates this function block is executing
	Data type	BOOL
Error	Description	Indicates an invalid input
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT
Value	Description	Value of the parameter
	Data type	LREAL

Example**Structured Text**

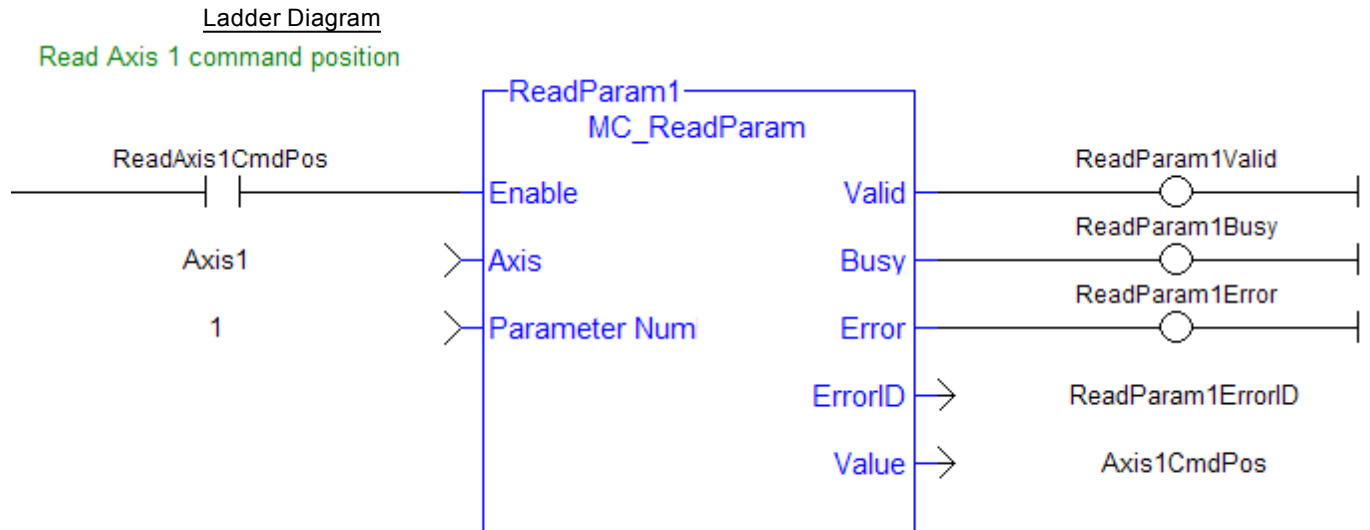
```
(* MC_ReadParam ST example *)

ParameterNumber := 3; //configure the parameter to read

Inst_MC_ReadParam( EnableRead, Axis1, ParameterNumber );

//Inst_MC_ReadParam is an instance of MC_ReadParam
function block

ParmVal := Inst_MC_ReadParam.Value; //store the Value
output into a user defined variable
```



1.2.3.6 MC_ReadStatus

Description

The function block MC_ReadStatus returns the state of the specified axis.

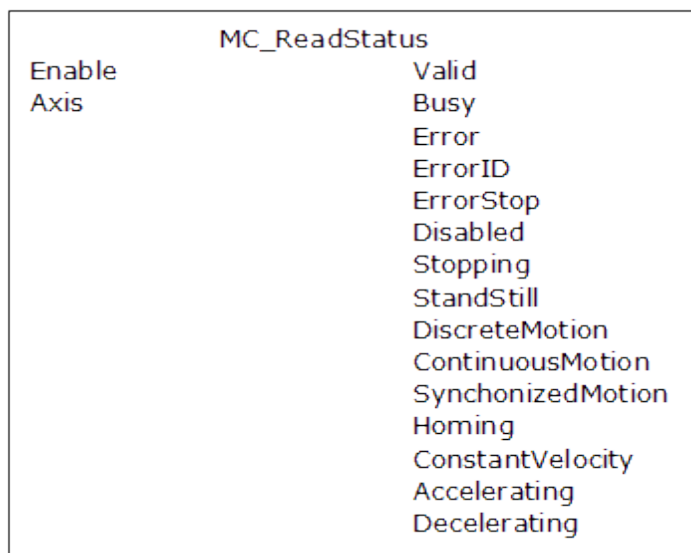


Figure 1-52: MC_ReadStatus

Arguments

Input

Argument	Description
Enable	Requests to read and return the axis status
	Data type: BOOL
	Range: 0, 1
	Unit: n/a
	Default: —

Axis	Description	Name of a declared instance of the AXIS_REF library function. click here...
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
 <u>Output</u>		
Valid	Description	Indicates the outputs are valid
	Data type	BOOL
Busy	Description	Indicates this function block is executing
	Data type	BOOL
Error	Description	Indicates an invalid input
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT
ErrorStop	Description	Indicates Error Stop state – E-stop or C-stop
	Data type	BOOL
Disabled	Description	Indicates Disabled state – open loop and drive is disabled
	Data type	BOOL
Stopping	Description	Indicates Stopping state – MC_Stop command
	Data type	BOOL
StandStill	Description	Indicates Stand Still state – no move, closed loop, drive enabled
	Data type	BOOL
DiscreteMotion	Description	Indicates Discrete Motion state – programmed endpoint move is active
	Data type	BOOL
ContinuousMotion	Description	Indicates Continuous Motion state – unending, single-axis move is active
	Data type	BOOL
SynchronizedMotion	Description	Indicates Synchronized Motion state – slave move is active
	Data type	BOOL
Homing	Description	Indicates Homing state – a homing cycle is currently executing
	Data type	BOOL
ConstantVelocity	Description	Indicates the axis is moving at a constant velocity
	Data type	BOOL
Accelerating	Description	Indicates the axis is accelerating
	Data type	BOOL

Decelerating	Description Data type	Indicates the axis is decelerating BOOL
--------------	--------------------------	--

Example

Structured Text

```
(* MC_ReadStatus ST example *)

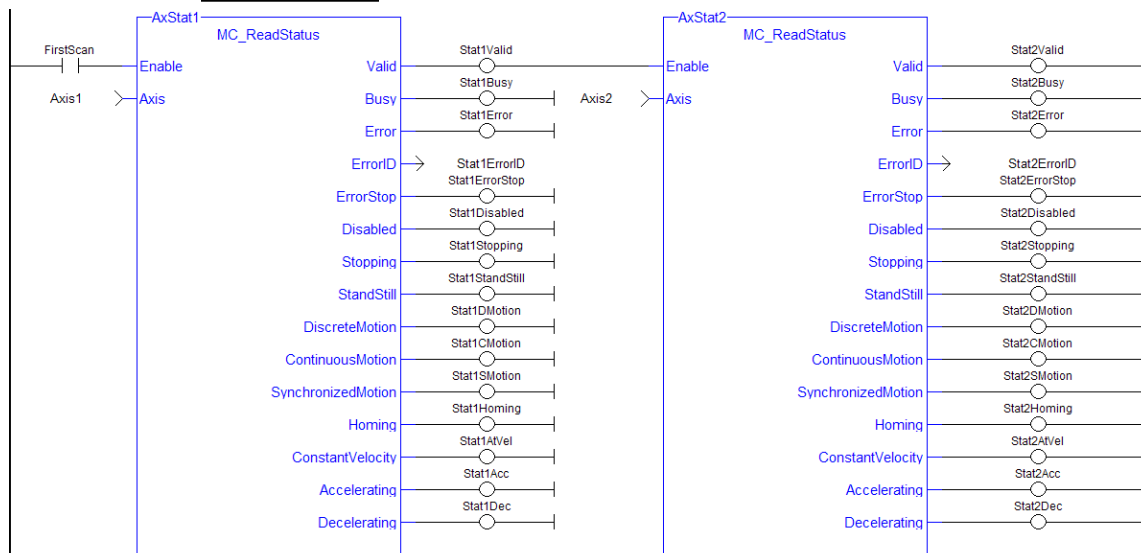
Inst_MC_ReadStatus( EnableRead, Axis1 );

//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
```

Ladder Diagram



1.2.3.7 MC_WriteBoolPar (Function Block)

Description

MC_WriteBoolPar writes the specified axis Boolean parameter.

Arguments

Input

Execute	Description	Requests to write a Boolean axis parameter
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—

Axis	Description	Name of a declared instance of the AXIS_REF library function.
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
ParameterNumber	Description	Parameter number, see table in § "Motion Library (PLCopen)"
	Data type	INT
	Range	—
	Unit	n/a
	Default	—
Value	Description	State to write
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
<u>Output</u>		
Done	Description	Indicates the Boolean parameter has been written
	Data type	BOOL
Busy	Description	Indicates this function block is executing
	Data type	BOOL
Error	Description	Indicates an invalid input
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT

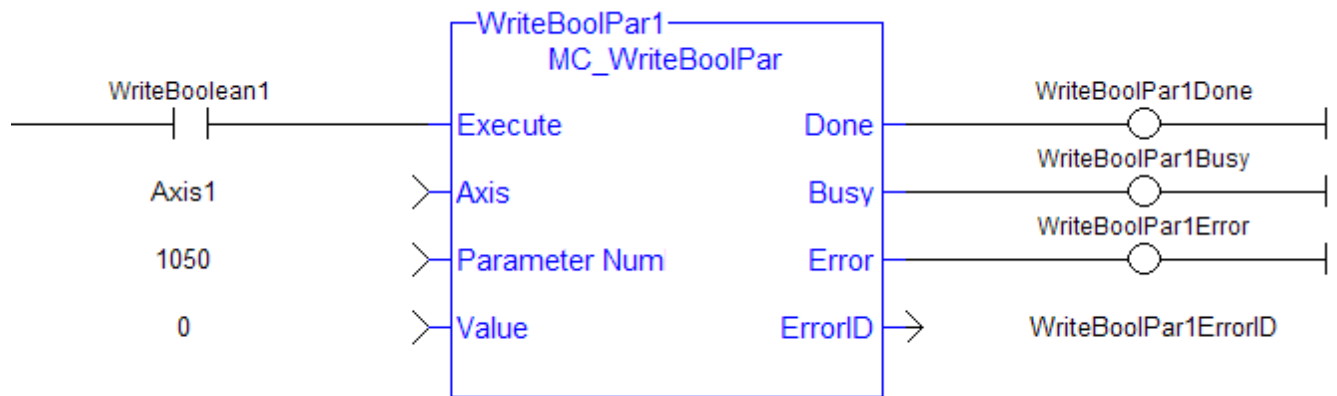
Example**Structured Text**

```
(* MC_WriteBoolPar ST example *)

WriteBool := TRUE; //value to write to the boolean
parameter #1

Inst_MC_WriteBoolPar( WriteReq, Axis1, 1, WriteBool );
//Inst_MC_WriteBoolPar is an instance of MC_WriteBoolPar
```

Ladder Diagram



NOTE Currently, MC_WriteBoolPar does not support any parameters (1050 is an arbitrary number chosen for example)

1.2.3.8 MC_WriteParam (Function Block)

Description

MC_WriteParam writes the specified axis parameter.

Arguments

Input

Argument	Description	Request
Execute	Requests to write the axis parameter	
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	Name of a declared instance of the AXIS_REF library function.	
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
ParameterNumber	Parameter number, see table in § "Motion Library (PLCopen)"	
	Data type	INT
	Range	—
	Unit	n/a
	Default	—
Value	Value to write	
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—

Output

Done	Description	Indicates the parameter has been written
	Data type	BOOL
Busy	Description	Indicates this function block is executing
	Data type	BOOL
Error	Description	Indicates an invalid input
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT

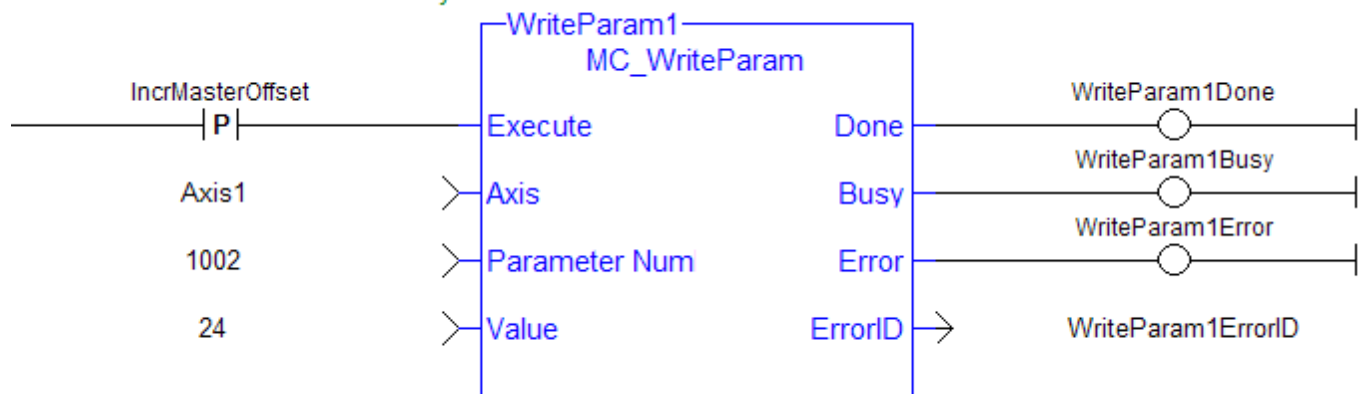
Example

Structured Text

```
(* MC_WriteParam ST example *)
WriteValue := 1234.2; //value to write to parameter 1002
Inst_MC_WriteParam( WriteReq, Axis1, 1002, WriteValue);
//Inst_MC_WriteParam is an instance of MC_WriteParam
```

Ladder Diagram

Increment the master offset delta by 24



1.2.4 PLCopenMotion

1.2.4.1 MC_Halt (Function Block)

Description

This function block decelerates an axis to zero velocity. 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. To execute a stop that cannot be aborted, see MC_Stop.

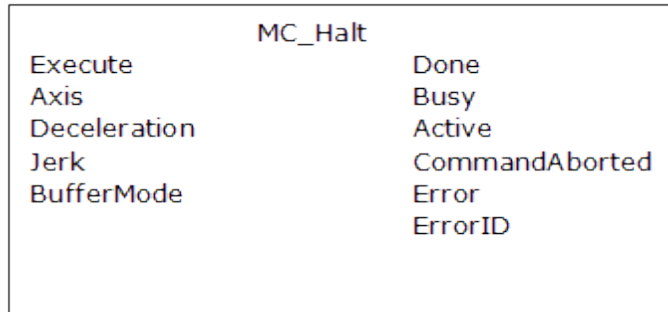


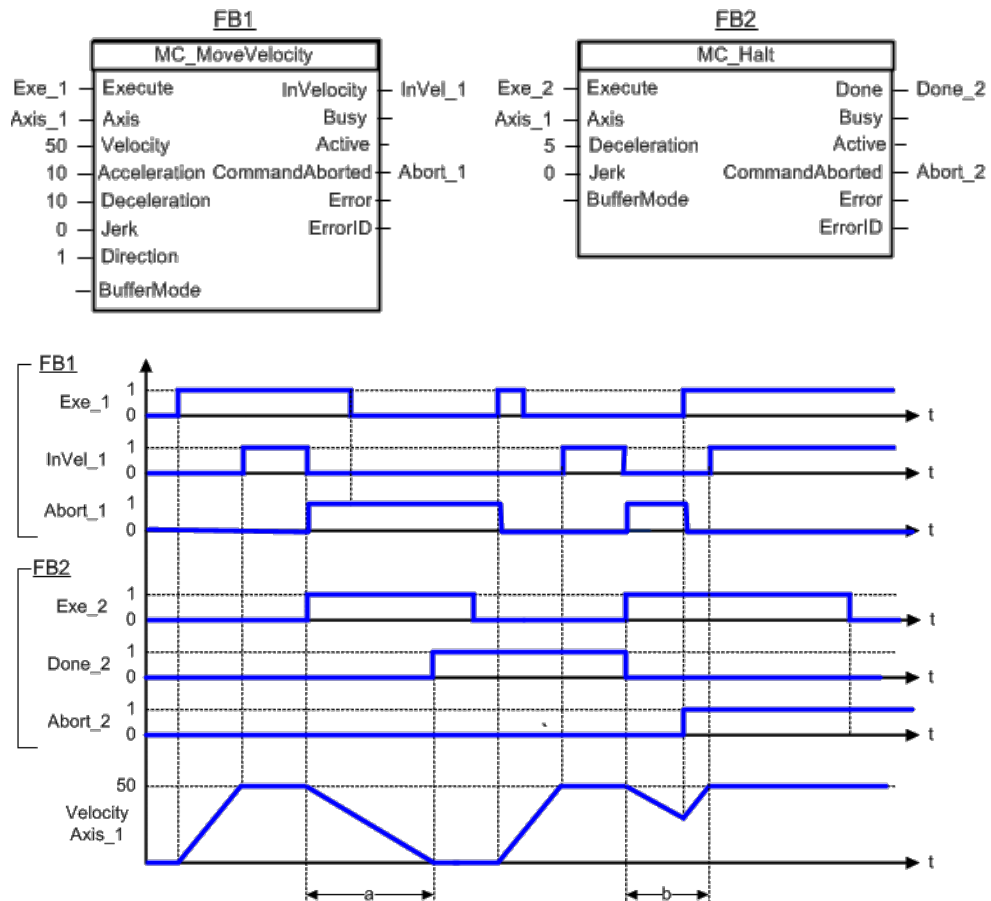
Figure 1-53: MC_Halt

Time Diagram

The example below shows the behavior in combination with a MC_MoveVelocity.

- 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.



Arguments

Input

Execute	Description Data type	Requests to queue the move BOOL
---------	--------------------------	------------------------------------

	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Deceleration	Description	Trapezoidal: Deceleration rate S-curve: Maximum deceleration
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Jerk	Description	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec ³
	Default	—
BufferMode	Description	0 = abort 1 = buffer 2 = blend to active 3 = blend to next 4 = blend to low velocity 5 = blend to high velocity
	Data type	SINT
	Range	[0,5]
	Unit	n/a
	Default	—
<u>Output</u>		
Done	Description	Indicates the move completed successfully. The Command Position has reached the endpoint.
	Data type	BOOL
Busy	Description	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
Active	Description	Indicates this move is the active move
	Data type	BOOL
CommandAborted	Description	Indicates this move was aborted
	Data type	BOOL
Error	Description	Indicates an invalid input was specified or the move was terminated due to an error

	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT

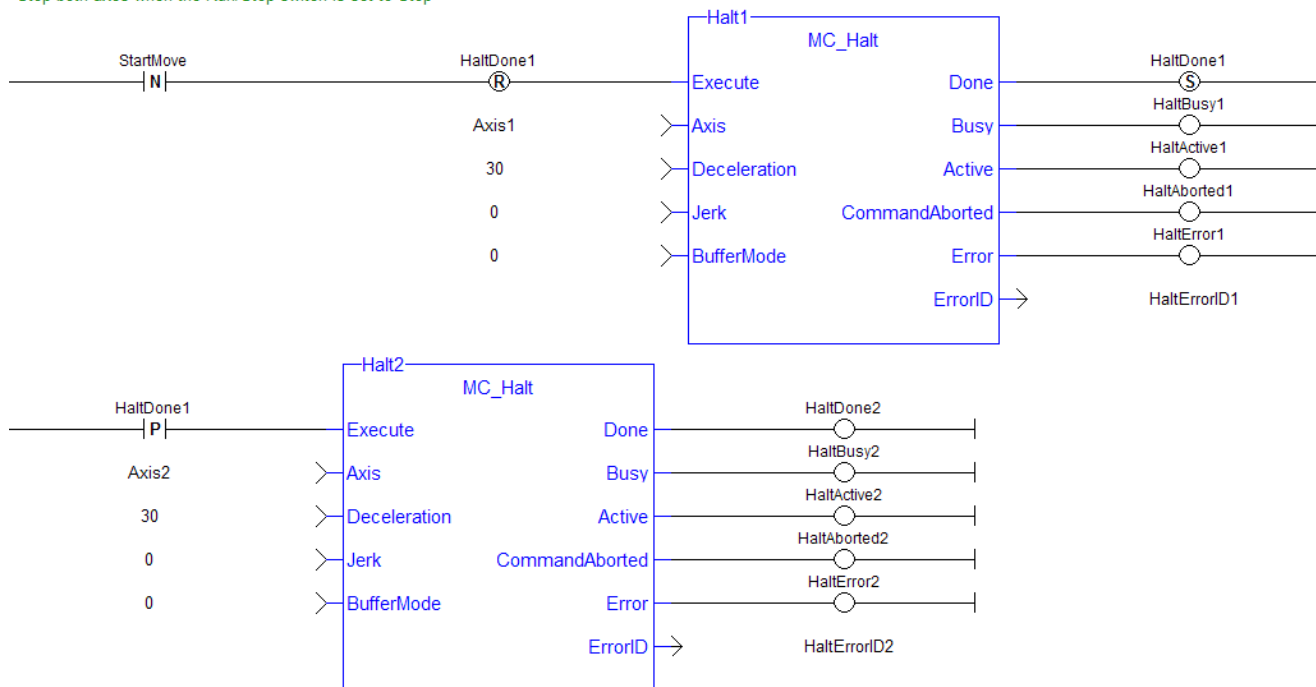
Example

Structured Text

```
(* 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
```

Ladder Diagram

Stop both axes when the Run/Stop switch is set to Stop



1.2.4.2 MC_MoveAbsolute (Function Block)

Description

This function block performs a single-axis move to a specified endpoint position based on Axis, Position, Velocity, Acceleration, Deceleration, Jerk, and Direction parameters.

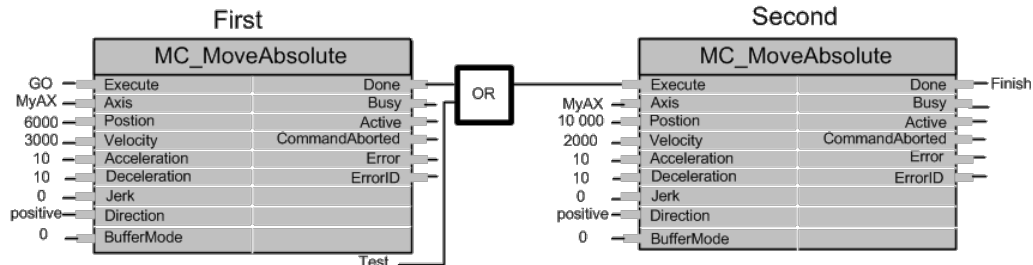
MC_MoveAbsolute	
Execute	Done
Axis	Busy
Position	Active
Velocity	CommandAborted
Acceleration	Error
Deceleration	ErrorID
Jerk	
Direction	
BufferMode	

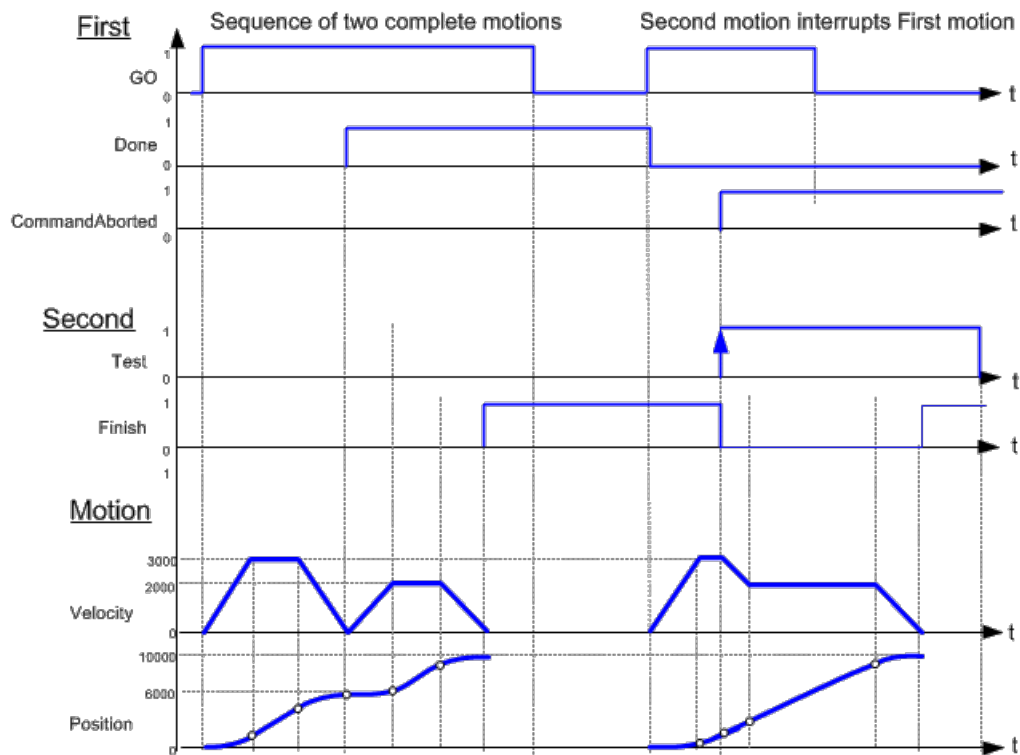
Figure 1-54: MC_MoveAbsolute

Time Diagram

The following figure shows two examples of the combination of two absolute move Function Blocks:

- The left part of timing diagram illustrates the case if the Second Function Block is called **after** the First one. If First reaches the commanded position of 6000 (and the velocity is 0) then the output Done causes the Second FB to move to the position 10000
- The right part of the timing diagram illustrates the case if the Second move Function Block starts the execution **while** the First FB is still executing. In this case 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





Arguments

Input

Execute	Description Data type Range Unit Default	Requests to queue the move BOOL 0, 1 n/a —
Axis	Description Data type Range Unit Default	Name of a declared instance of the AXIS_REF library function AXIS_REF [1,256] n/a —
Position	Description Data type Range Unit Default	Endpoint position. If Rollover Position is nonzero, this value must be in the range $0 \leq \text{Position} < \text{Rollover Position}$ When not in Rollover mode, the input accepts a 64-bit floating point value. When converted to feedback units, the range is $[-2^{51}, 2^{51}-1]$ feedback units. LREAL [see Description] User unit —
Velocity	Description Data type Range	Velocity setpoint LREAL —

	Unit	User unit/sec
	Default	—
Acceleration	Description	Trapezoidal: Acceleration rate S-curve: Maximum acceleration See also "Selection of Acceleration and Jerk Parameters for Function Blocks"
		NOTE If Acceleration is not valid, ErrorID is set to 21
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Deceleration	Description	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Jerk	Description	Trapezoidal: 0 S-curve: Constant jerk See also "Selection of Acceleration and Jerk Parameters for Function Blocks"
		NOTE If Jerk is not valid, ErrorID is set to 21
	Data type	LREAL
	Range	—
	Unit	User unit/sec ³
	Default	—

Direction

Description

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.

Value	Description
0	no direction specification
1	positive direction. The axis travels in the positive direction to the endpoint
2	shortest distance. The axis travels in the direction that provides the shortest distance to the endpoint
3	negative direction. The axis travels in the negative direction to the endpoint
4	last direction. The axis travels to the endpoint in the same direction as its previous move

 **NOTE**

If the Position input is the same as the axis's current position, then:

- when Direction = **2** (shortest distance), the axis does not move and the Done output goes high indicating that the move has been completed.
- when Direction = **1, 3, or 4**, the axis travels in the specified direction, through one rollover cycle, and arrives back at the same

 **NOTE** position.

	Data type	SINT
	Range	[0,4]
	Unit	n/a
	Default	—
BufferMode	Description	0 = abort 1 = buffer 2 = blend to active 3 = blend to next 4 = blend to low velocity 5 = blend to high velocity
	Data type	SINT
	Range	[0,5]
	Unit	n/a
	Default	—
<u>Output</u>		
Done	Description	Indicates the move completed successfully. The Command Position has reached the endpoint.
	Data type	BOOL
Busy	Description	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
Active	Description	Indicates this move is the active move
	Data type	BOOL
CommandAborted	Description	Indicates the move was aborted
	Data type	BOOL
Error	Description	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT

Example

Structured Text

```
(* MC_MoveAbsolute ST 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;

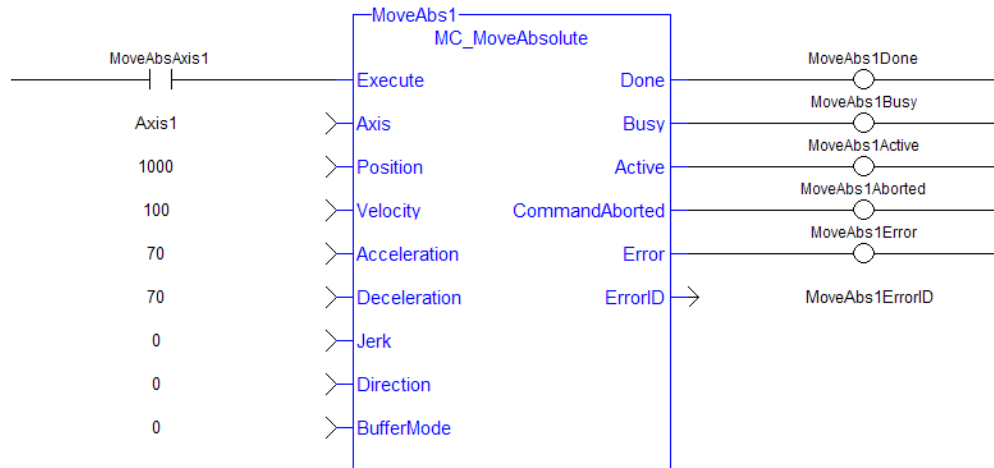
MovAbsAborted := Inst_MC_MoveAbsolute.CommandAborted;

MovAbsError := Inst_MC_MoveAbsolute.Error;

MovAbsErrID := Inst_MC_MoveAbsolute.ErrorID;
```

Ladder Diagram

Move Axis 1 to 1000



1.2.4.3 MC_MoveAdditive (Function Block)

Description

This function block performs a single-axis move for a specified distance from the endpoint of the previous move. It is typically used with Abort specified at the BufferMode input. If BufferMode is not Abort, this move is identical to an MC_MoveRelative.

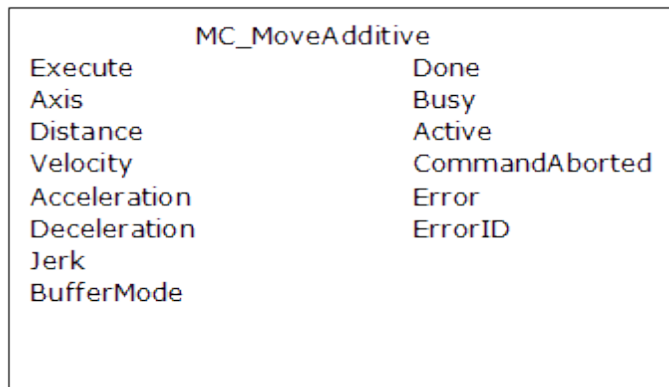
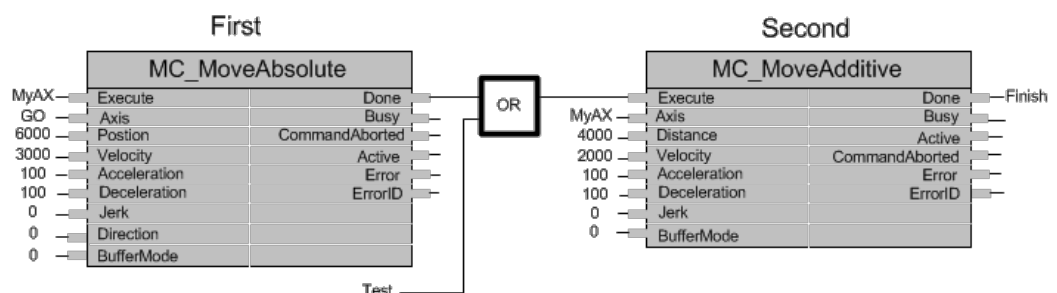


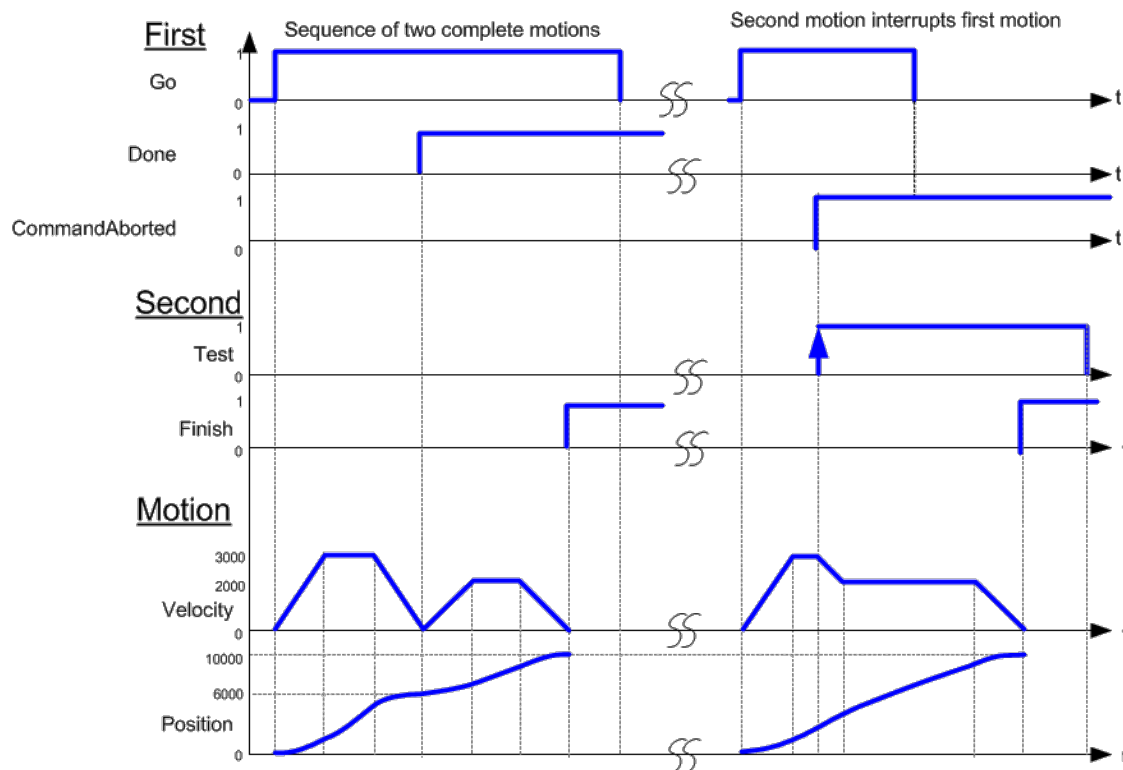
Figure 1-55: MC_MoveAdditive

Time Diagram

The following figure shows two examples of the combination of two Function Blocks while the axis is in Discrete Motion state:

- The left part of timing diagram illustrates the case if the Second Function Block is called **after** the First one. If First reaches the commanded distance 6000 (and the velocity is 0) then the output **Done** causes the Second FB to move to the distance 10000
- The right part of the timing diagram illustrates the case if the Second move Function Blocks starts the execution **while** the First FB is still executing. In this case 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 and moves the axis to the resulting position of 10000





Arguments

Input

Execute	Description	Requests to queue the move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	Name of a declared instance of the AXIS_REF library function.)
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Distance	Description	Distance to add to the endpoint of the previous move
	Data type	REAL
	Range	—
	Unit	User unit
	Default	—
Velocity	Description	Velocity setpoint
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	Description	Trapezoidal: Acceleration rate S-curve: Maximum acceleration

	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Deceleration	Description	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Jerk	Description	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec ³
	Default	—
BufferMode	Description	0 = abort 1 = buffer 2 = blend to active 3 = blend to next 4 = blend to low velocity 5 = blend to high velocity
	Data type	SINT
	Range	[0,5]
	Unit	n/a
	Default	—
<u>Output</u>		
Done	Description	Indicates the move completed successfully. The Command Position has reached the endpoint.
	Data type	BOOL
Busy	Description	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
Active	Description	Indicates this move is the active move
	Data type	BOOL
CommandAborted	Description	Indicates the move was aborted
	Data type	BOOL
Error	Description	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT

Example

Structured Text

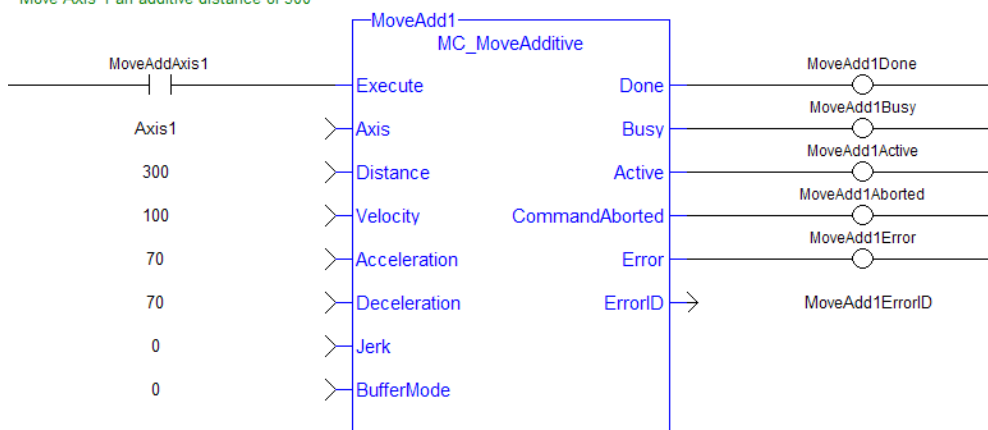
```
(* MC_MoveAdditive ST example *) //Inst_MC_MoveAdditive is
an instance of MC_MoveAdditive function block

Inst_MC_MoveAdditive( MovAddReq, Axis1, 123.456, 100.0,
100.0, 100.0, 0, 0 );

MovAddDone := Inst_MC_MoveAdditive.Done; //store Done
output into user defined variable
```

Ladder Diagram

Move Axis 1 an additive distance of 300



1.2.4.4 MC_MoveRelative

Description

This function block executes a single-axis move for a specified distance to perform incremental motion.

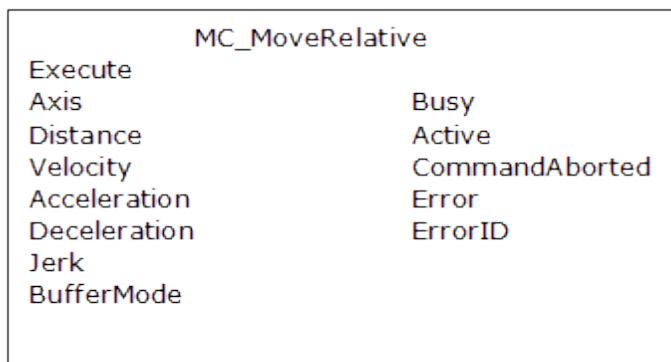
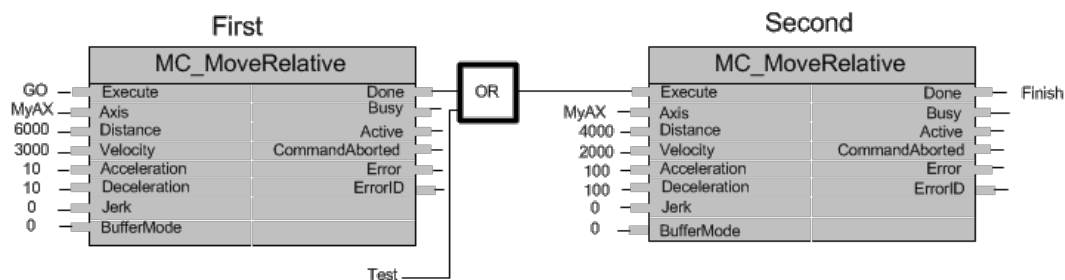


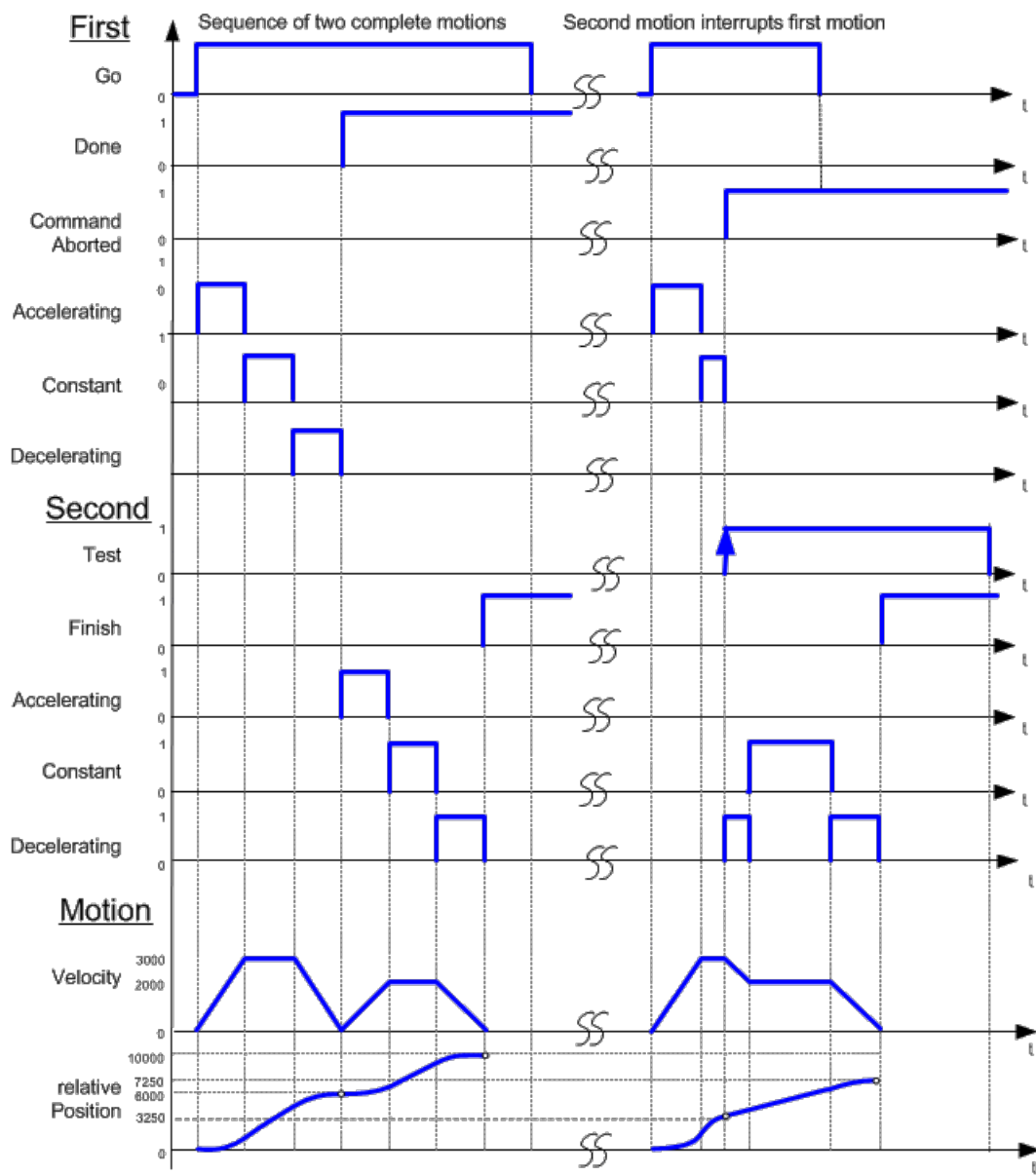
Figure 1-56: MC_MoveRelative

Time Diagram

The following figure shows the example of the combination of two relative move Function Blocks:

- The left part of timing diagram illustrates the case if the Second Function Block is called **after** the First one. If First reaches the commanded distance 6000 (and the velocity is 0) then the output **Done** causes the Second FB to move to the distance 10000
- The right part of the timing diagram illustrates the case if the Second move Function Blocks starts the execution **while** the First FB is still executing. In this case 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





Arguments

Input

Execute

Description Requests to queue the move
 Data type BOOL
 Range 0, 1
 Unit n/a
 Default —

Axis

Description Name of a declared instance of the AXIS_REF library function.
 Data type AXIS_REF
 Range [1,256]
 Unit n/a
 Default —

Distance

Description Distance

	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
Velocity	Description	Velocity setpoint
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	Description	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Deceleration	Description	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Jerk	Description	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec ³
	Default	—
BufferMode	Description	0 = abort 1 = buffer 2 = blend to active 3 = blend to next 4 = blend to low velocity 5 = blend to high velocity
	Data type	SINT
	Range	[0,5]
	Unit	n/a
	Default	—
<u>Output</u>		
Done	Description	Indicates the move completed successfully. The Command Position has reached the endpoint.
	Data type	BOOL
Busy	Description	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
Active	Description	Indicates this move is the active move

	Data type	BOOL
CommandAborted	Description	Indicates the move was aborted
	Data type	BOOL
Error	Description	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT

Example

Structured Text

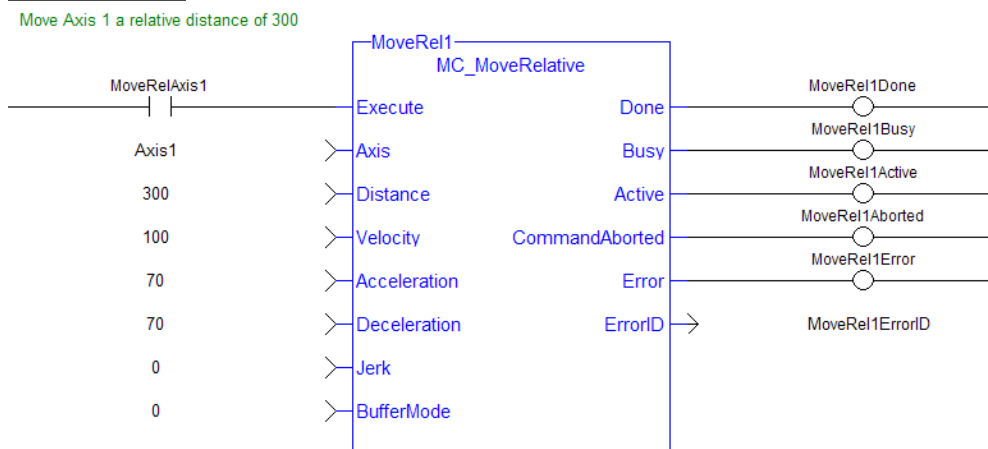
```
(* MC_MoveRelative ST example *)

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 how this function is used in the Hole punch project here

Ladder Diagram



1.2.4.5 MC_MoveSuperimp (Function Block)

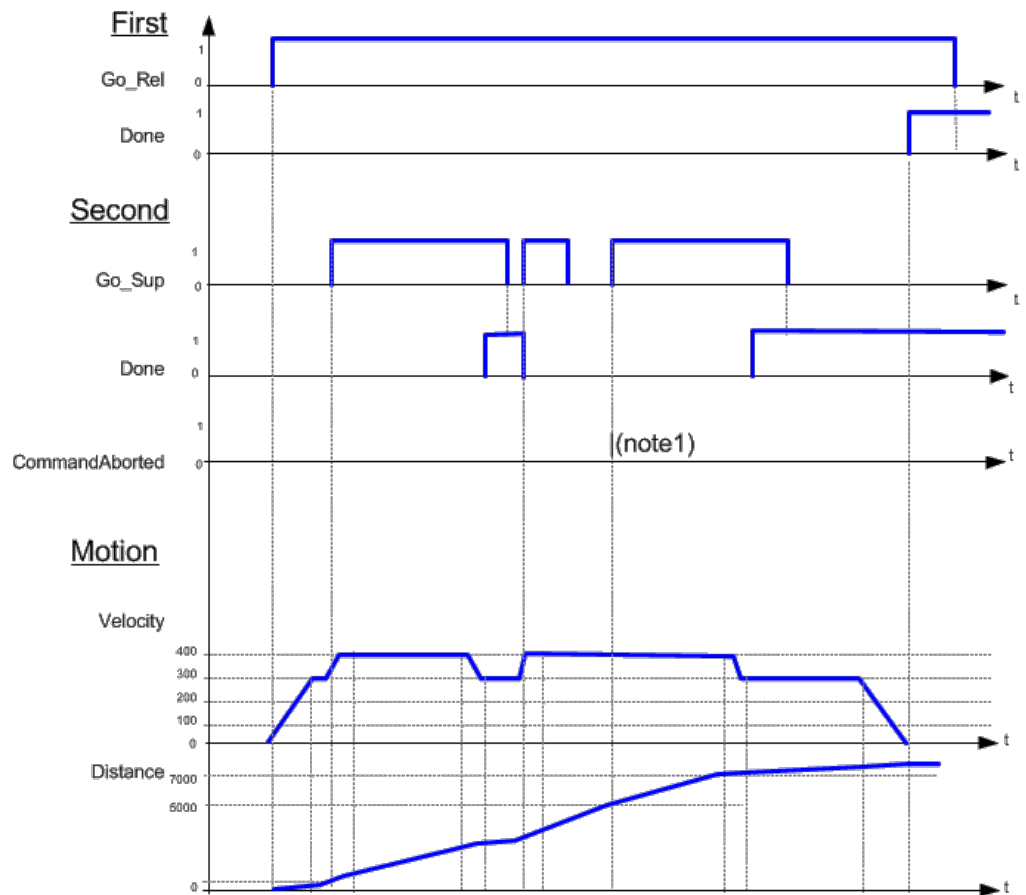
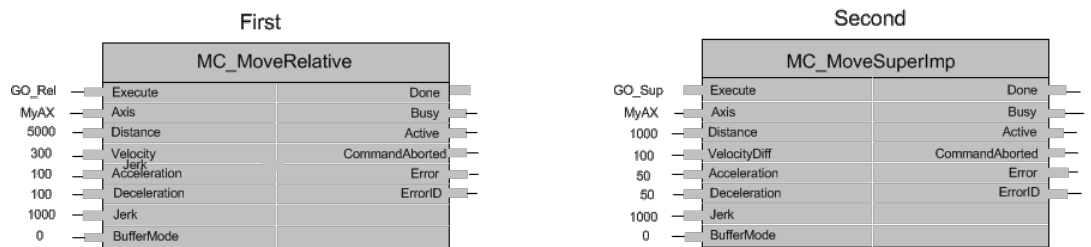
Description

This function block performs a relative single-axis move which is superimposed upon the active move. Superimposed moves have their own Profile Generator and queue. Superimposed moves can be aborted by and blended with other superimposed moves. The distance of the superimposed move is an addition to the existing motion.

MC_MoveSuperimp	
Execute	Done
Axis	Busy
Distance	Active
VelocityDiff	CommandAborted
Acceleration	Error
Deceleration	ErrorID
Jerk	
BufferMode	

Figure 1-57: MC_MoveSuperimp

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

ArgumentsInput

Execute	Description	Requests to queue the superimposed move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	Name of a declared instance of the AXIS_REF library function.
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Distance	Description	Distance
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
Velocity	Description	Velocity rate
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	Description	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Deceleration	Description	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Jerk	Description	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec ³
	Default	—

BufferMode	Description	0 = abort 1 = buffer 2 = blend to active 3 = blend to next 4 = blend to low velocity 5 = blend to high velocity
	Data type	SINT
	Range	[0,5]
	Unit	n/a
	Default	—

Output

Done	Description	Indicates the move completed successfully. The Command Position has reached the endpoint.
	Data type	BOOL
Busy	Description	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
Active	Description	Indicates this move is the active superimposed move
	Data type	BOOL
CommandAborted	Description	Indicates the move was aborted
	Data type	BOOL
Error	Description	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT

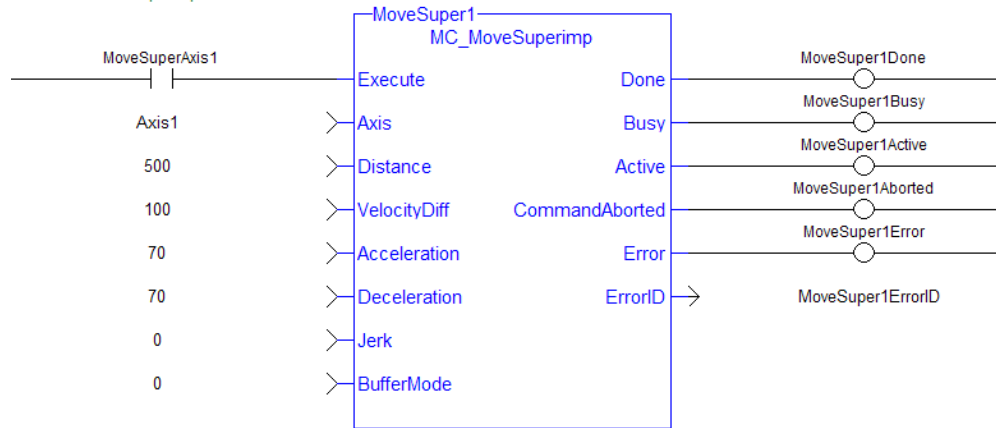
Example**Structured Text**

```
(* 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
```

Ladder Diagram

Move Axis 1 a superimposed distance of 500



1.2.4.6 MC_MoveVelocity (Function Block)

Description

This function block performs a single-axis non-ending move at a specified velocity. This type of move can be terminated with the MC_Halt function block or by aborting it with another move.

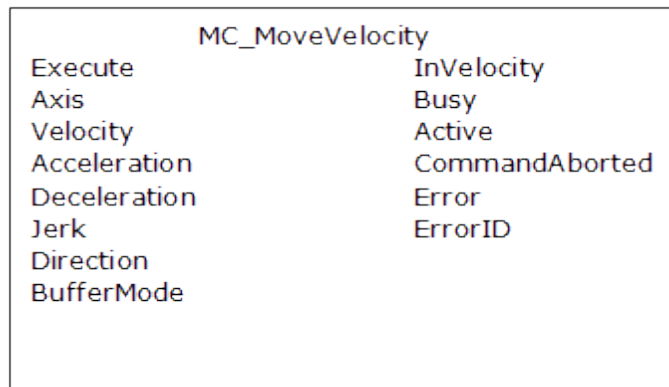
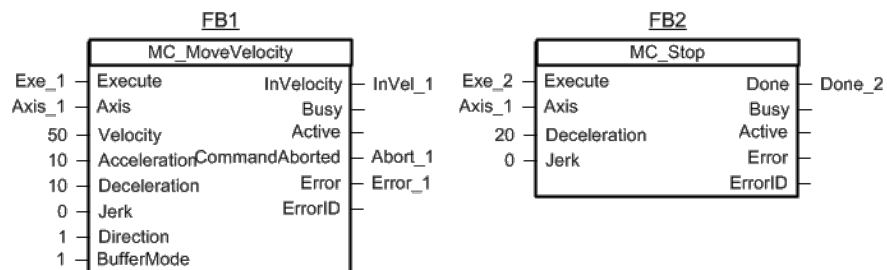


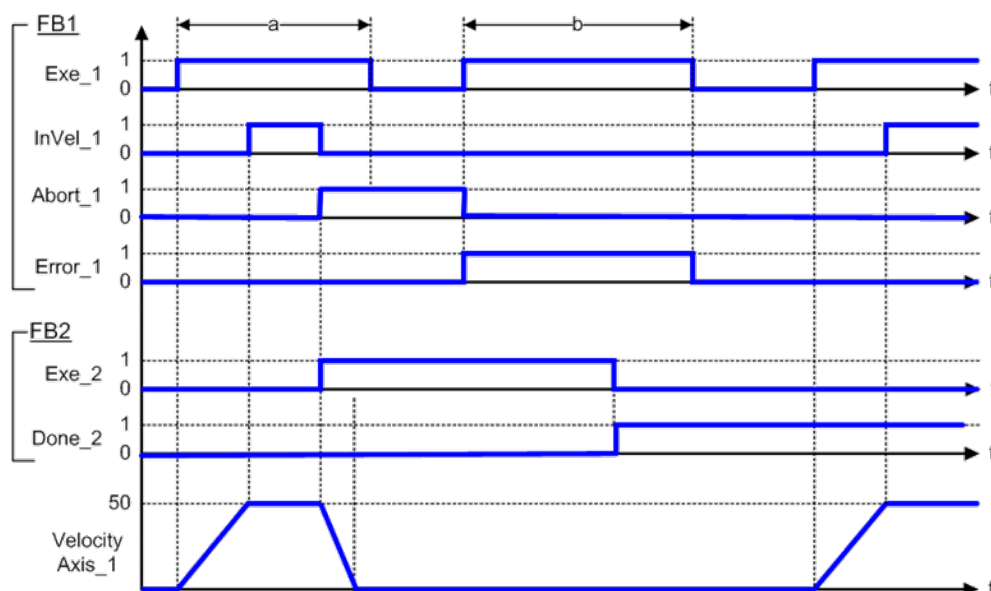
Figure 1-58: MC_MoveVelocity

Time Diagram

The example below 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.





Arguments

Input

Execute	Description	Requests to queue the move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Velocity	Description	Velocity rate
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	Description	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Deceleration	Description	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—

Jerk	Description	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec ³
	Default	—
Direction	Description	0 = positive direction 1 = negative direction
	Data type	SINT
	Range	[0,1]
	Unit	n/a
	Default	—
BufferMode	Description	0 = abort 1 = buffer 2 = blend to active 3 = blend to next 4 = blend to low velocity 5 = blend to high velocity
	Data type	SINT
	Range	[0,5]
	Unit	n/a
	Default	—
Output		
InVelocity	Description	Indicates the command velocity has reached the programmed velocity
	Data type	BOOL
Busy	Description	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
Active	Description	Indicates this move is the active move
	Data type	BOOL
CommandAborted	Description	Indicates the move was aborted
	Data type	BOOL
Error	Description	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT

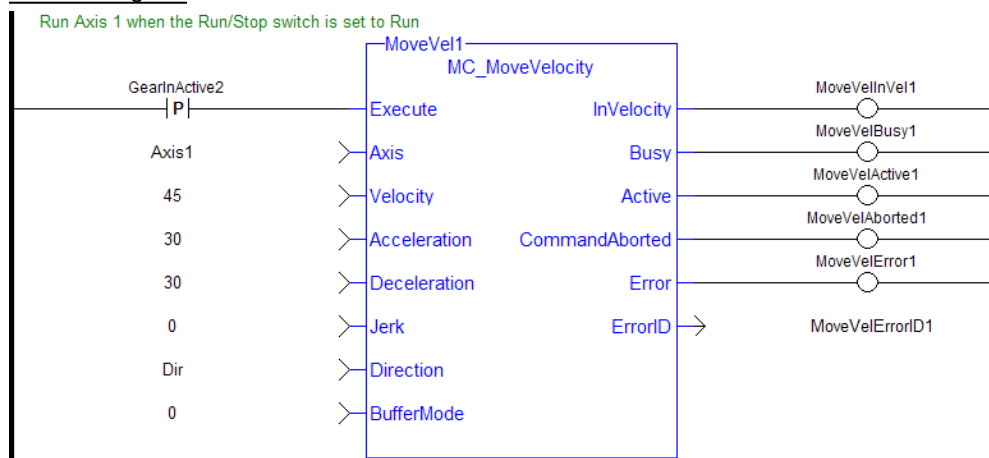
Example

Structured Text

```
(* MC_MoveVelocity ST example *)

Inst_MC_MoveVelocity( MovVelReq , Axis1, 200.0,
100.0,100.0, 0, 0, 0 );
```

Ladder Diagram



1.2.4.7 MC_SetOverride (Function Block)

Description

This function block writes the velocity override factor. A change in the velocity override factor takes effect immediately on the active move.

The velocity override factor is applied to the programmed velocity to determine the command velocity:

$$\text{command velocity} = \text{programmed velocity} * \text{VelFactor}$$

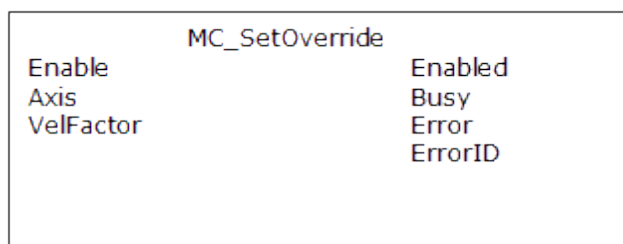


Figure 1-59: MC_SetOverride

Arguments

Input

Argument	Description	Data type
Enable	Request to write the override factors	BOOL

	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	Name of a declared instance of the AXIS_REF library function.
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
VelFactor	Description	Velocity override factor
	Data type	REAL
	Range	[0.0, 2.0]
	Unit	n/a
	Default	—
Output		
Enabled	Description	Indicates the override values have been written
	Data type	BOOL
Busy	Description	Indicates the function block is executing
	Data type	BOOL
Error	Description	Indicates an invalid input is specified
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT

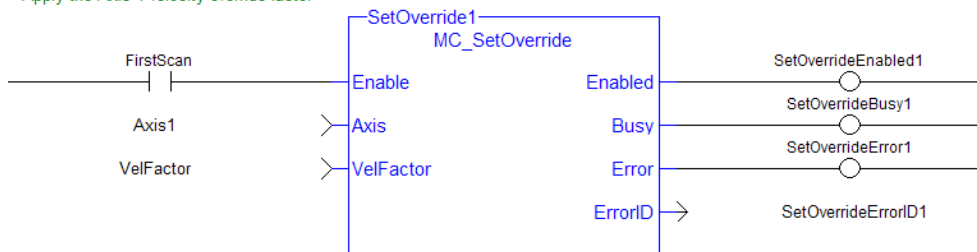
Example

Structured Text

```
(* 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
```

Ladder Diagram

Apply the Axis 1 velocity override factor



1.2.5 Profile

1.2.5.1 MC_CamIn

Description

This function block performs a slave axis move which follows the master axis based on the Cam Table specified by CamTableID.

Arguments

Input

Execute	Description	Requests to queue the CamIn move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Master	Description	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	1 - 256
	Unit	n/a
	Default	—
Slave	Description	AXIS_REF.AXIS_NUM is the slave axis number
	Data type	AXIS_REF
	Range	1-256
	Unit	n/a
	Default	—
MasterOffset	Description	Profile shift along the master axis
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
Slave	Description	AXIS_REF.AXIS_NUM is the slave axis number
	Data type	AXIS_REF
	Range	1-256
	Unit	n/a
	Default	—
SlaveOffset	Description	Profile shift along the slave axis
	Data type	LREAL
	Range	—

	Unit	User unit
	Default	—
MasterScaling	Description	Master axis profile range
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
SlaveScaling	Description	Slave axis profile range
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
Startmode	Description	(future addition, must be set to zero)
	Data type	INT
	Range	0
	Unit	n/a
	Default	—
CamTableID	Description	ID number of the profile to be used with MC_CamIn
	Data type	INT
	Range	—
	Unit	n/a
	Default	—
BufferMode	Description	Buffer mode for CamIn block
	Data type	SINT
	Range	0-5
	Unit	n/a
	Default	—
SlaveScaling	Description	Slave axis profile range
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
CamTableID	Description	ID number of the profile to be used with MC_CamIn
	Data type	INT
	Range	—
	Unit	n/a
	Default	—

<u>Output</u>		
InSync	Description	Indicates the slave axis is in sync with the profile
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Busy	Description	Indicates this function block is executing
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Active	Description	Indicates this move is the Active move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
CommandAborted	Description	Indicates the move was aborted
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Error	Description	Indicates an invalid input, or the move was terminated due to an error
	Data type	BOOL
	Range	0, 1
	Unit	n/a
ErrorID	Description	Indicates the error if the Error output is high
	Data type	INT
	Range	—
	Unit	n/a
EndOfProfile	Description	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.
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Active	Description	Indicates this move is the Active move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Error	Description	Indicates an invalid input, or the move was terminated due to an error

	Data type	BOOL
	Range	0, 1
	Unit	n/a
EndOfProfile	Description	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.
	Data type	BOOL
	Range	0, 1
	Unit	n/a

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.

If the profile is periodic, when the end of profile 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.

Adjustments computation is done as follows:

When cam is first started, offsets are adjusted if necessary

- If slave is not absolute, then slave offset = slave offset + starting position
- If master is not absolute, then master offset = master 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

Related Functions

MC_CamTblSelect

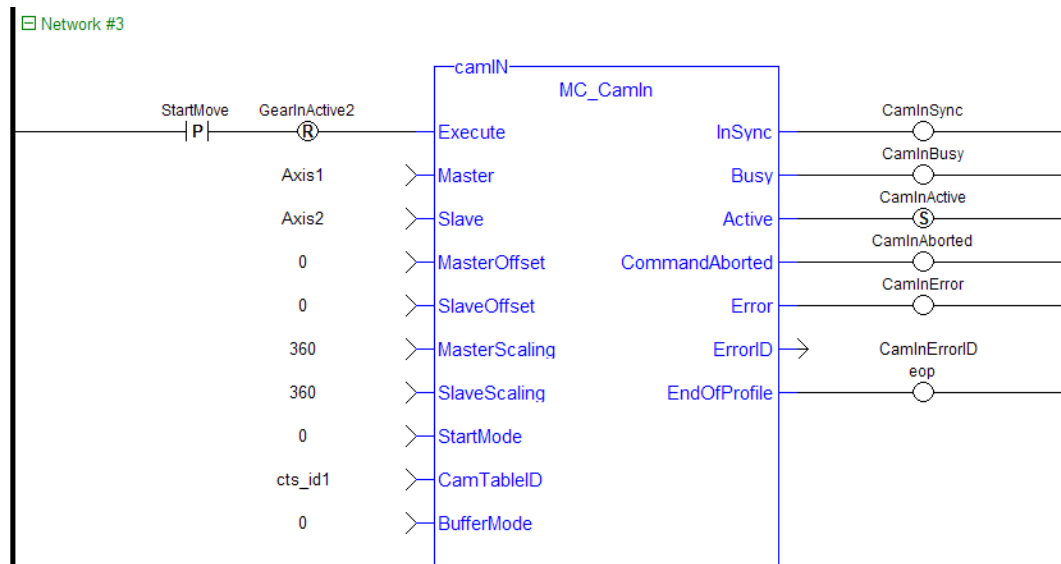
MC_CamOut

Examples

Structured Text

```
(* 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 );
```

Ladder Diagram



The three following examples utilizes the screen shot below showing the cam profile “MyProfile”

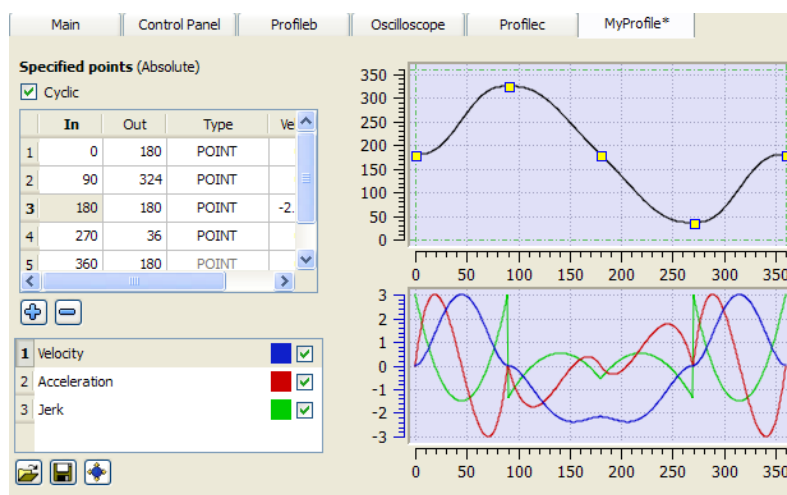


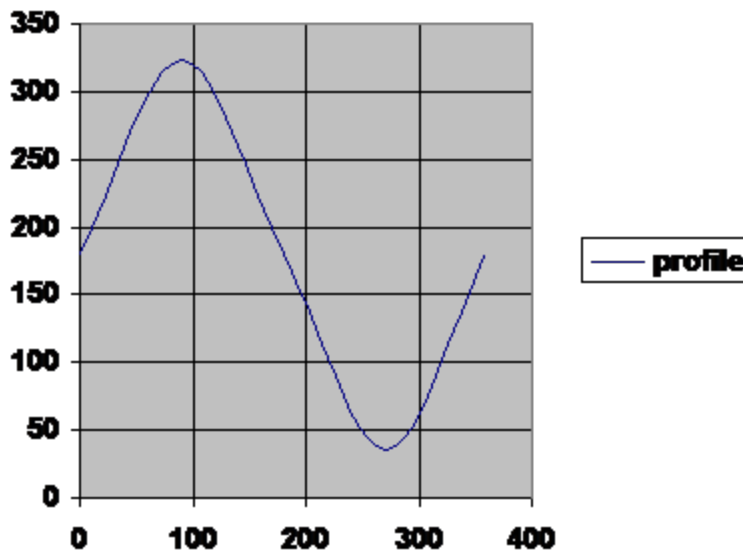
Figure 1-60: MC_CamIn examples

Example 1

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 above parameters, the slave axis is locked on to the profile. Since both have zero offsets, the profile is not shifted in either axis. The initial condition of the master axis at position 0, 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, then the slave follows the profile again.

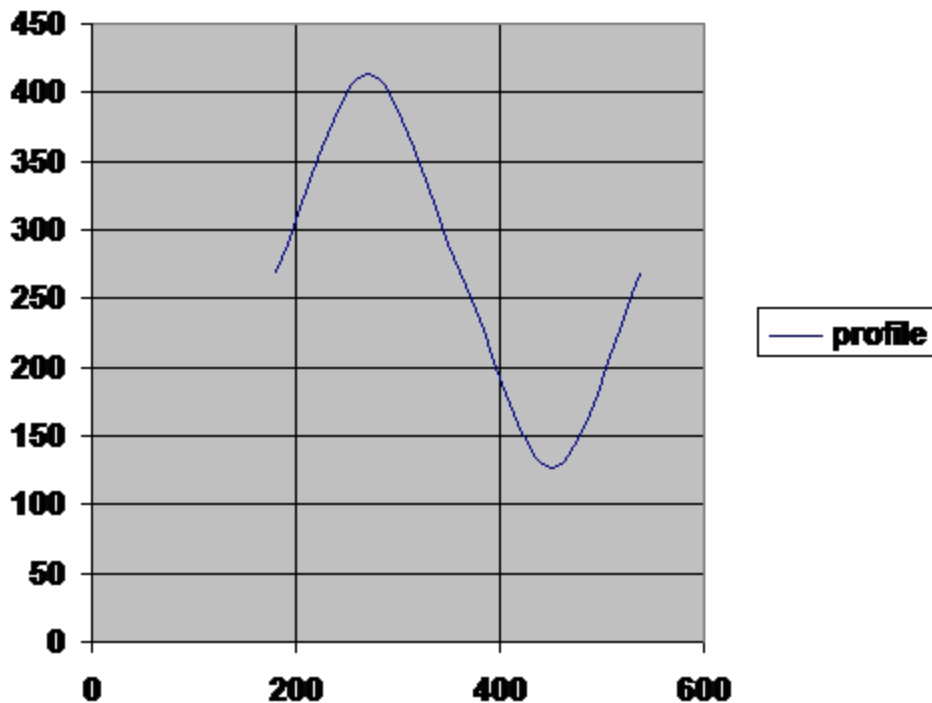


Example 2

Profile	MyProfile
Cycle	YES
MasterAbsolute	NO
SlaveAbsolute	NO
MasterOffset	0.0
SlaveOffset	0.0
MasterScaling	360.0
SlaveScaling	360.0
Initial Master position	180
Initial Slave position	90.0

After `MC_CamTblSelect` and `MC_CamIn` are programmed with the above parameters, the slave axis is locked on to the profile. Since the 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).



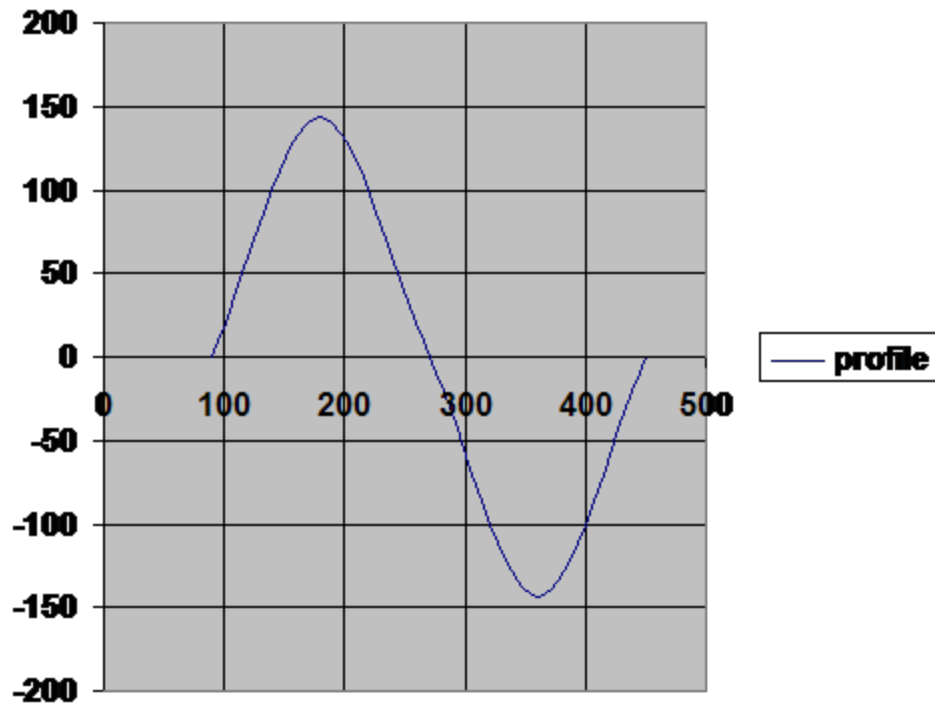
Example 3

Profile	MyProfile
Cycle	NO
MasterAbsolute	YES
SlaveAbsolute	YES
MasterOffset	90
SlaveOffset	-180
MasterScaling	360.0
SlaveScaling	360.0
Initial Master position	180
Initial Slave position	144

After `MC_CamTblSelect` and `MC_CamIn` are programmed with the above 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), which 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 (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 since the Cyclic input is false.



1.2.5.2 MC_CamOut

Description

This function block:

- aborts the active MC_CamIn move
- disengages the axis from its master
- and commands the axis to continue at its current velocity

Like a MC_MoveVelocity move, the control continues to command the axis to move at this velocity until this MC_CamOut move is aborted. 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.

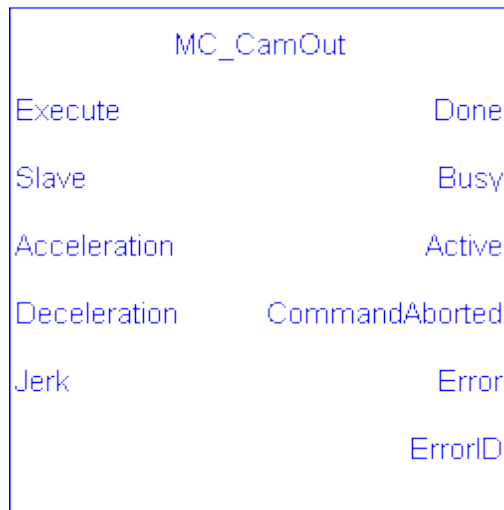


Figure 1-61: MC_CamOut

ArgumentsInput

Execute	Description	Requests to queue the CamOut move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Slave	Description	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	1 – 256
	Unit	n/a
	Default	—
Acceleration	Description	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Deceleration	Description	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Jerk	Description	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec ³
	Default	—
<u>Output</u>		
Done	Description	Indicates the axis is disengaged from its master
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Busy	Description	Indicates this function block is executing
	Data type	BOOL
	Range	0, 1

Active	Unit	n/a
	Description	Indicates this move is the Active move
	Data type	BOOL
	Range	0, 1
CommandAborted	Unit	n/a
	Description	Indicates the move was aborted
	Data type	BOOL
	Range	0, 1
Error	Unit	n/a
	Description	Indicates an invalid input was specified or no MC_CamIn move was active
	Data type	BOOL
	Range	0, 1
ErrorID	Unit	n/a
	Description	Indicates the error if the Error output is high
	Data type	INT
	Range	—
	Unit	n/a

Usage

This function block disengages the slave axis from a MC_CamIn move and then leaves the axis running at its current velocity. The axis continues to run at this velocity until this move is aborted.

Related Functions

MC_CamIn

MC_CamTblSelect

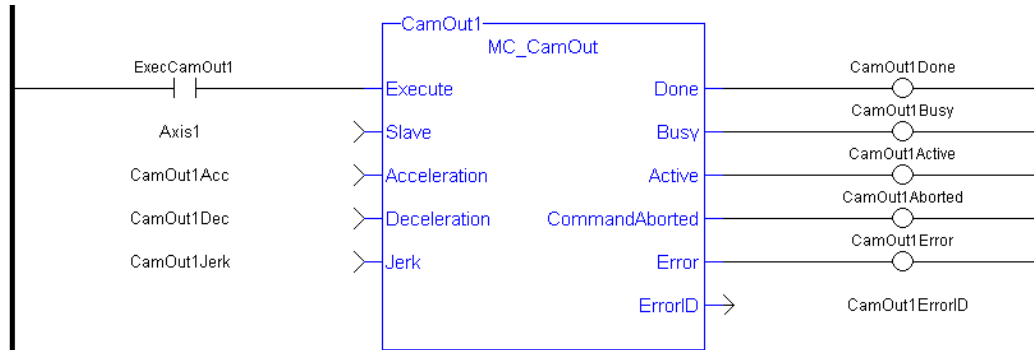
Example**Structured Text**

```
(* MC_CamOut ST example *)

Inst_MC_CamOut
(ExecCamOut1,Axis1,CamOut1Acc,CamOut1Dec,CamOut1Jerk);

//Inst_MC_CamOut is an instance of MC_CamOut
```

Ladder Diagram



See also MC_CamIn for examples.

1.2.5.3 MC_CamTblSelect

Description

This Function Block is defined to read and initialize the specified profile, returning an ID to be used with MC_CamIn function block.

Arguments

Input

Execute	Description	Requests to queue the slave gear ratio move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
CamTable	Description	Profile name as defined in the CAM Profile Properties dialog
	Data type	STRING
	Range	—
	Unit	n/a
	Default	—
Periodic	Description	Selects if the profile is periodic (see also Usage section)
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
MasterAbsolute	Description	Selects if master profile is absolute or relative (see also Usage section)
	Data type	BOOL
	Range	0, 1

	Unit	n/a
	Default	—
SlaveAbsolute	Description	Selects if Slave profile is absolute or relative (see also Usage section)
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
<u>Output</u>		
Done	Description	Indicates the function block has completed successfully
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Busy	Description	Indicates this function block is executing
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Error	Description	Indicates an invalid input was specified
	Data type	BOOL
	Range	0, 1
	Unit	n/a
ErrorID	Description	Indicates the error if the Error output is high
	Data type	INT
	Range	—
	Unit	n/a
CamTableID	Description	Indicates the ID number of the profile to be used with MC_CamIn
	Data type	INT
	Range	0 - 255
	Unit	n/a

Usage

- Each positive transition of the **Enable** input will create 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 will be true and the **ErrorID** output will be 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.

NOTE

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.
- Similarly, 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.

TIP If the SlaveAbsolute input is set to TRUE, the axis jumps back to the starting position. If you set this input to FALSE, the axis will no longer jump back; but rather, as the profile repeats, the slave moves relative to the start of each period.

Related Functions

MC_CamIn
MC_CamOut

Example

Structured Text

```
(* 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

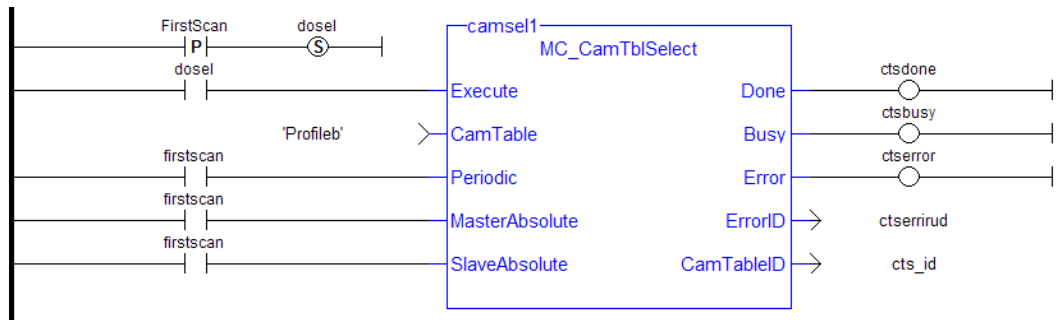
IF CamSelDone = TRUE THEN//when function block is "done"
store

CamTableID := Inst_MC_CamTblSelect.CamTableID; //CamTableID
in user defined variable

END_IF;
```

See also how this function is used in the Hole punch project here

Ladder Diagram



See also MC_CamIn for examples.

1.2.5.4 MC_GearIn

Description

This function block performs a slave axis move which follows the master axis based on the ratio specified by RatioNumerator and RatioDenominator.

$$\text{SlaveCommandPosition} = \frac{\text{MasterActualPosition} * \text{RatioNumerator}}{\text{RatioDenominator}}$$

When this command is executed, the slave axis accelerates or decelerates (using the Acceleration, Deceleration, and Jerk inputs) to the target velocity determined by the master axis velocity and the ratio. When the slave axis reaches that target velocity, it locks on to the master and the InGear output goes high. The slave axis continues to follow the master axis until this move is aborted.

Time to Reach the Target Velocity

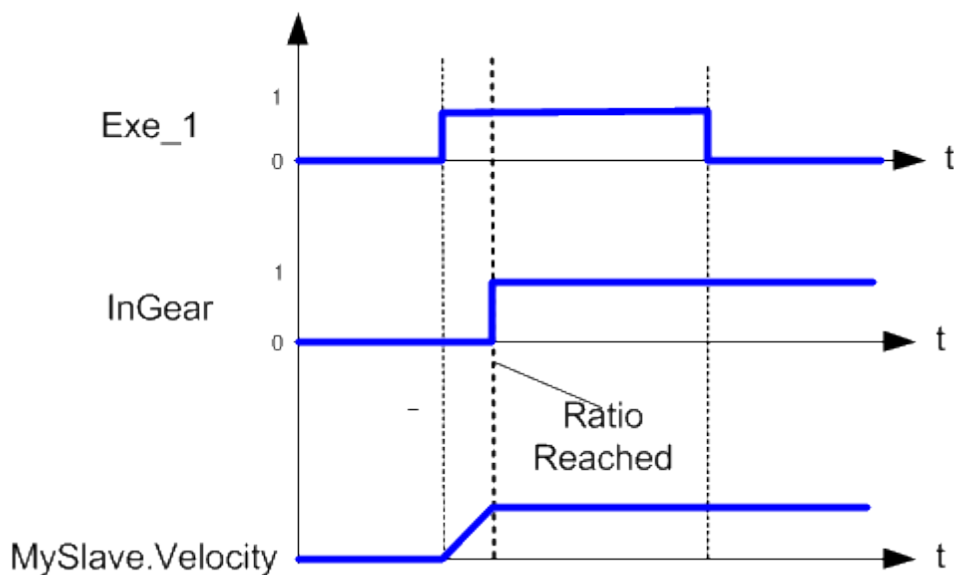
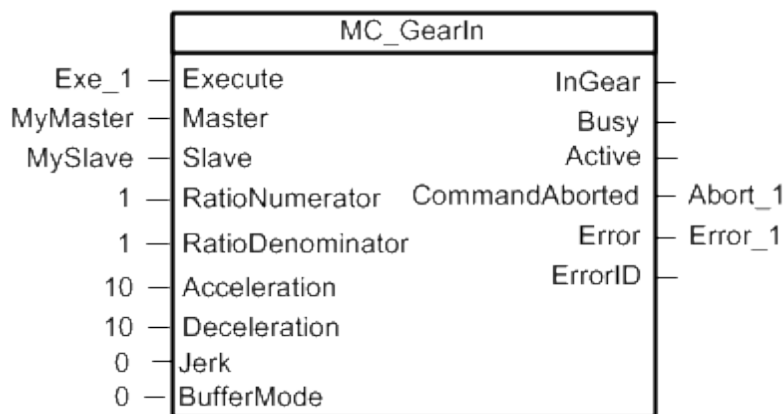
While following the master, gearing functions can generate large accelerations. If the gearing function is aborted while the axis is currently accelerating, and the aborting function block has small non-zero Jerk or small acceleration values, it can take a long time to reach the target velocity, or position 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:

- Abort the gearing function with an MC_GearOut with higher accelerations and/or Jerk values (or zero jerk value),
- Execute the next MC motion function block.

MC_GearIn	
Execute	InGear
Master	Busy
Slave	Active
RatioNumerator	CommandAborted
RatioDenominator	Error
Acceleration	ErrorID
Deceleration	
Jerk	
BufferMode	

Figure 1-62: MC_GearIn

Time Diagram



Arguments

Input

Execute	Description	Requests to queue the slave gear ratio move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Master	Description	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Slave	Description	AXIS_REF.AXIS_NUM is the slave axis number

	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
RatioNumerator	Description	Numerator of master/slave ratio
	Data type	DINT
	Range	[-2147483648, 2147483647]
	Unit	n/a
	Default	—
RatioDenominator	Description	Denominator of master/slave ratio
	Data type	DINT
	Range	[-2147483648, 2147483647]
	Unit	n/a
	Default	—
Acceleration	Description	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Deceleration	Description	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Jerk	Description	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec ³
	Default	—
BufferMode	Description	0 = abort 1 = buffer
	Data type	SINT
	Range	[0,1]
	Unit	n/a
	Default	—
<u>Output</u>		
InGear	Description	Indicates the slave axis is locked on to the master axis
	Data type	BOOL
Busy	Description	High from the moment the Execute input goes high until the time the move is ended
	Data type	BOOL
Active	Description	Indicates this move is the Active move

	Data type	BOOL
<code>CommandAborted</code>	Description	Indicates the move was aborted
	Data type	BOOL
<code>Error</code>	Description	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
<code>ErrorID</code>	Description	Indicates the error if Error output is set to TRUE
	Data type	INT

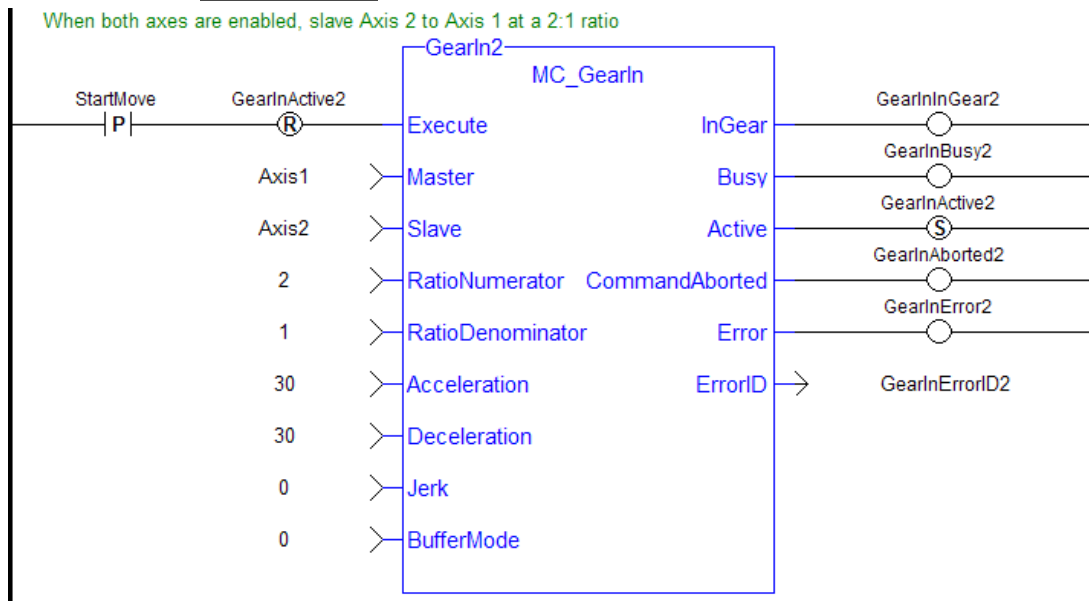
Example

Structured Text

```
(* 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
```

See also how this function is used in the Hole punch project here

Ladder Diagram



1.2.5.5 MC_GearInPos

Description

This function block performs a slave axis move which follows the master axis based on the ratio specified by RatioNumerator and RatioDenominator.

$$\text{SlaveCommandPosition} = \frac{\text{MasterActualPosition} * \text{RatioNumerator}}{\text{RatioDenominator}}$$

This function block also 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.

Time to Reach the Target Velocity

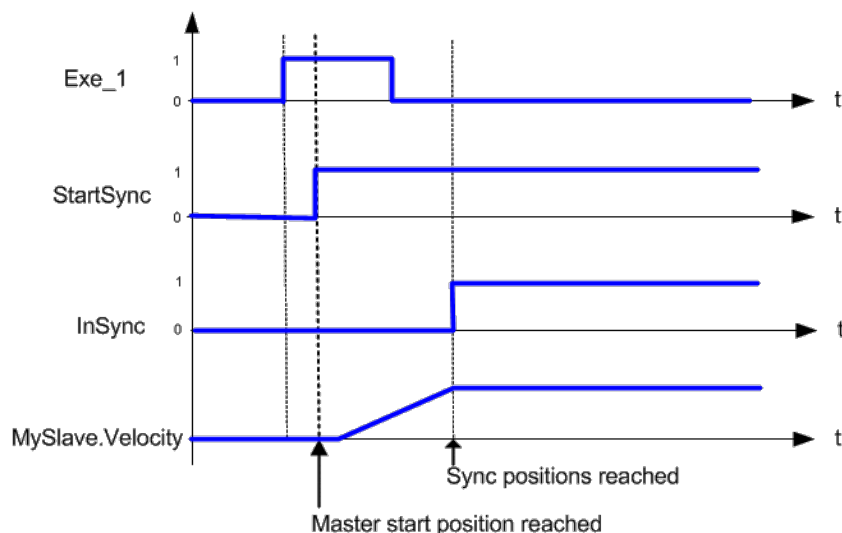
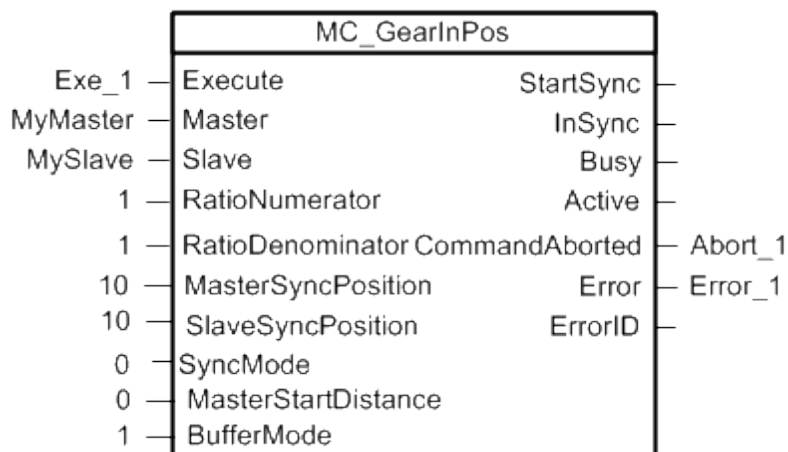
While following the master, gearing functions can generate large accelerations. If the gearing function is aborted while the axis is currently accelerating, and the aborting function block has small non-zero Jerk or small acceleration values, it can take a long time to reach the target velocity, or position 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:

- Abort the gearing function with an MC_GearOut with higher accelerations and/or Jerk values (or zero jerk value),
- Execute the next MC motion function block.

MC_GearInPos	
Execute	StartSync
Master	InSync
Slave	Busy
RatioNumerator	Active
RatioDenominator	CommandAborted
MasterSyncPosition	Error
SlaveSyncPosition	ErrorID
SyncMode	
MasterStartDistance	
BufferMode	

Figure 1-63: MC_GearInPos

Time Diagram



Arguments

Input

Execute	Description	Requests to queue the slave gear ratio move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Master	Description	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Slave	Description	AXIS_REF.AXIS_NUM is the slave axis number

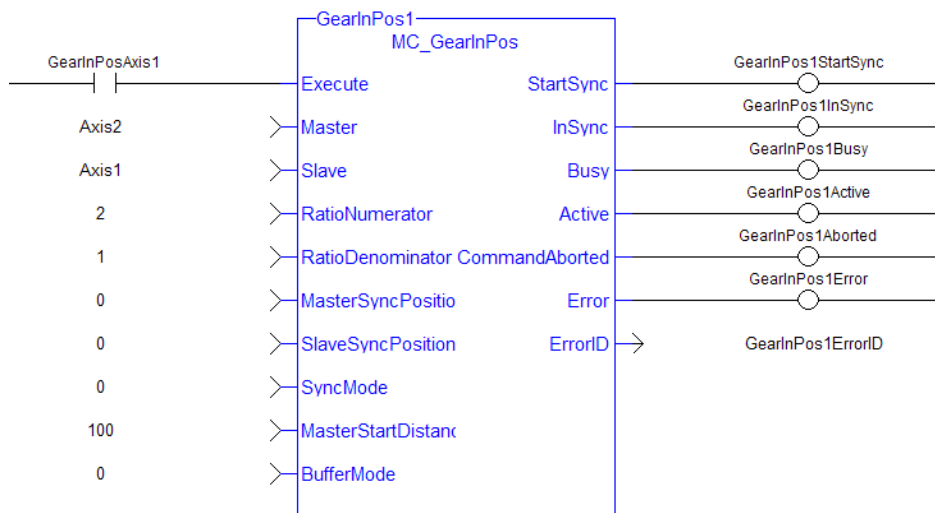
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
RatioNumerator	Description	Numerator of master/slave ratio
	Data type	DINT
	Range	[-2147483648, 2147483647]
	Unit	n/a
	Default	—
RatioDenominator	Description	Denominator of master/slave ratio
	Data type	DINT
	Range	[-2147483648, 2147483647]
	Unit	n/a
	Default	—
MasterSyncPosition	Description	Master axis sync position
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—
SlaveSyncPosition	Description	Slave axis sync position
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—
SyncMode	Description	for future enhancements
	Data type	INT
	Range	—
	Unit	n/a
	Default	—
MasterStartDistance	Description	When the master axis reaches this distance before MasterSyncPosition, the slave axis begins its lock-on process
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
BufferMode	Description	1 = buffer
	Data type	SINT
	Range	[1]
	Unit	n/a
	Default	—
<u>Output</u>		
StartSync	Description	Indicates that the master axis has reached the MasterStartDistance from the MasterSyncPosition and the lock-on process has begun
	Data type	BOOL

InSync	Description	Indicated the slave axis is locked on to the master axis
	Data type	BOOL
Busy	Description	High from the moment the Execute input goes high until the time the move is ended
	Data type	BOOL
Active	Description	Indicates this move is the Active move
	Data type	BOOL
CommandAborted	Description	Indicates the move was aborted
		<div style="background-color: #e6f2ff; padding: 5px; border: 1px solid #000080;"> <p>NOTE If the abort arises because the inputs cause inconsistent motion, then this FB:</p> <ul style="list-style-type: none"> performs no motion sets an error flag set the ErrorID to 13 </div>
	Data type	BOOL
Error	Description	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT

ExampleStructured Text

```
(* MC_GearInPos ST example *)
Inst_MC_GearInPos( GearInPosReq, Axis1, Axis2, 2, 1, 0, 0,
0, 100.0, 0 );
//Inst_MC_GearInPos is instance of MC_GearInPos
GearInPosSync:= Inst_MC_GearInPos.InSync; //store InSync
output into user defined variable
```

Ladder Diagram



1.2.5.6 MC_GearOut

Description

This function block:

- aborts the active MC_GearIn or MC_GearInPos move,
- disengages the axis from its master,
- and commands the axis to continue at its current velocity.

Like a MC_MoveVelocity move, 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.

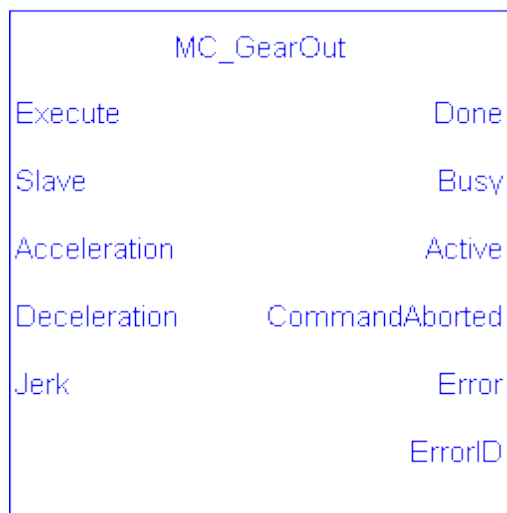


Figure 1-64: MC_GearOut

Arguments**Input**

Execute	Description	Requests to disengage the slave axis from a MC_GearIn or MC_GearInPos move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Slave	Description	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Acceleration	Description	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Deceleration	Description	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Jerk	Description	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec ³
	Default	—
<u>Output</u>		
Done	Description	Indicates the axis is disengaged from its master
	Data type	BOOL
Busy	Description	Indicates the function is executing
	Data type	BOOL
Active	Description	Indicates this move is the Active move
	Data type	BOOL
CommandAborted	Description	Indicates the move was aborted
	Data type	BOOL
Error	Description	Indicates an invalid input was specified or no MC_GearIn or MC_GearInPos move is active
	Data type	BOOL

ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT

Example

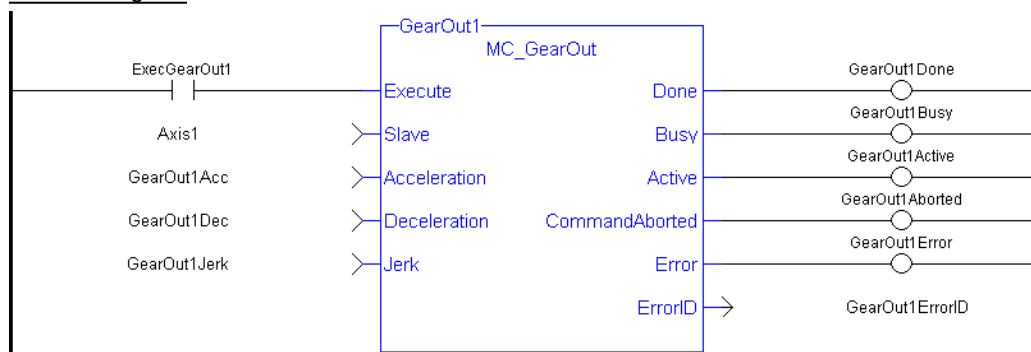
Structured Text

```
(* MC_GearOut ST example *)

Inst_MC_GearOut
(ExecGearOut1,Axis1,GearOut1Acc,GearOut1Dec,GearOut1Jerk);

//Inst_MC_GearOut is instance of MC_GearOut
```

Ladder Diagram



1.2.5.7 MC_Phasing

Description

This function block performs a master position phase shift for the slave axis. The phase shift is applied like a traditional single-axis move with a velocity setpoint and acceleration and deceleration rates. Phasing has its own Profile Generator and its own queue. Phase shifts can be aborted and blended with additional phase shifts. The amount of phase shift is added to the total master offset as the phase shift is executing.

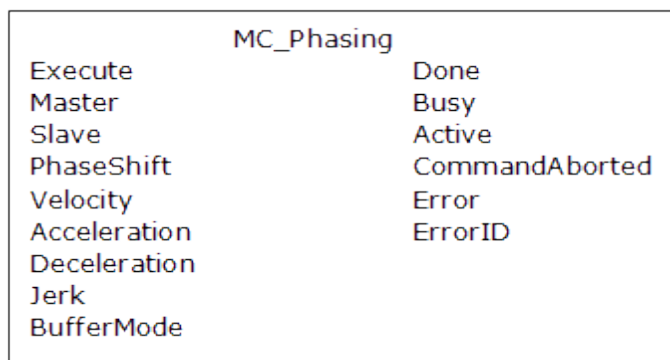


Figure 1-65: MC_Phasing

Arguments**Input**

Execute	Description	Requests to queue the phase shift
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Master	Description	Name of a declared instance of the AXIS_REF library function.)
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Slave	Description	AXIS_REF.AXIS_NUM is the slave axis number
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
PhaseShift	Description	Amount of phase shift
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
Velocity	Description	Velocity setpoint
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	Description	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Deceleration	Description	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Jerk	Description	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec ³
	Default	—

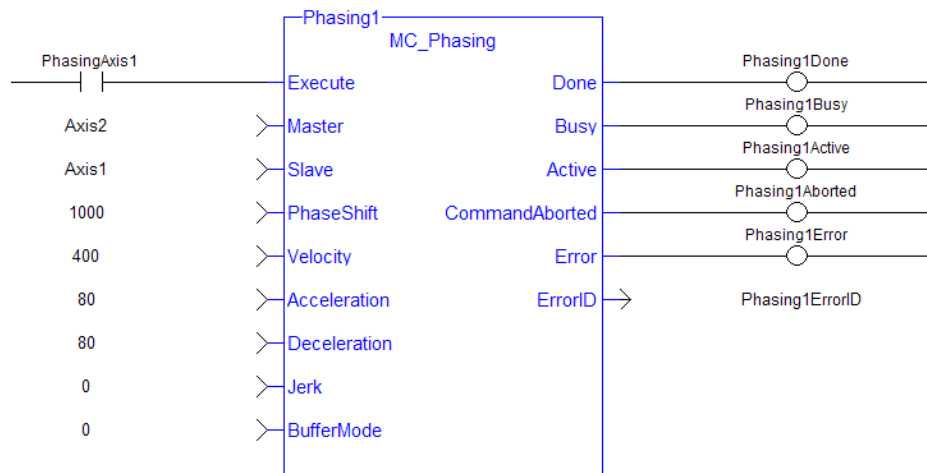
BufferMode	Description	0 = abort 1 = buffer 2 = blend to active 3 = blend to next 4 = blend to low velocity 5 = blend to high velocity
	Data type	SINT
	Range	[0,5]
	Unit	n/a
	Default	—
<u>Output</u>		
Done	Description	Indicates the phase shift has been completely applied
	Data type	BOOL
Busy	Description	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
Active	Description	Indicates this phase shift is the active phase shift
	Data type	BOOL
CommandAborted	Description	Indicates the move was aborted
	Data type	BOOL
Error	Description	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT

Example**Structured Text**

```
(* 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 );
```

Ladder Diagram



1.2.5.8 MC_SyncSlaves

Description

This function block allows the application to specify what slave axes are to be synchronized and which master they follow. After this function block is executed successfully, all the slave axes specified at the SlaveList input start their slave moves (i.e. MC_GearIn, MC_CamIn, 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.

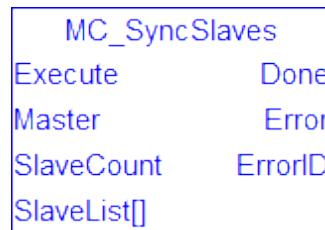


Figure 1-66: MC_SyncSlaves

Arguments

Input

Argument	Description	Value/Range
Execute	A positive transition of this input causes the function block to execute	BOOL 0, 1 n/a —
Master	Master axis identifier	AXIS_REF 1 - 256 n/a —

SlaveCount	Description	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, the list of synchronized slaves for the specified Master axis is cleared.
	Data type	AXIS_REF
	Range	1-256 The AXIS_NUM element of the AXIS_REF structure must be in the range [1-256]
	Unit	n/a
	Default	—
SlaveList	Description	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.
	Data type	UINT
	Range	1-32
	Unit	n/a
	Default	—
<u>Output</u>		
Done	Description	Indicates the synchronized slave assignments were completed without error
	Data type	BOOL
Error	Description	Indicates an invalid input was specified
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT

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.

Related Functions

MC_GearIn

MC_GearInPos

MC_CamIn

Example

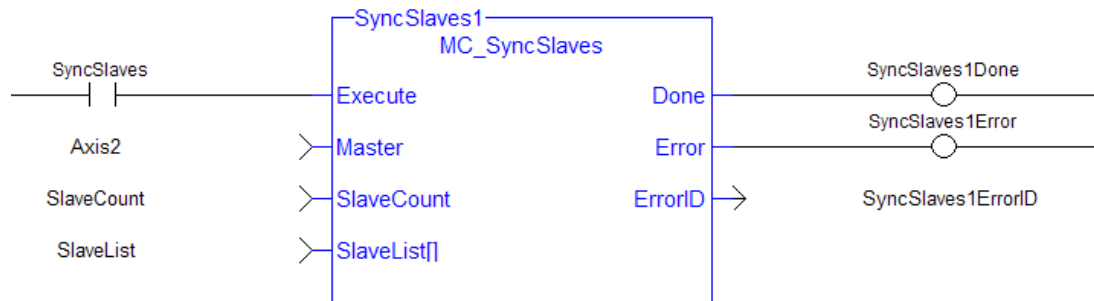
Structured Text

```
(* MC_SyncSlaves ST example *)

// Inst_MC_SyncSlaves is an instance of MC_SyncSlaves
function block

Inst_MC_SyncSlaves( SyncSlaves, Axis1, SlaveCount,
SlaveList );
```

Ladder Diagram



1.2.6 Reference

1.2.6.1 MC_Reference (Function Block)

Description

This function block is used to execute a fast home to a switch. 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.**

NOTE

At this time, position capture is not available for PLCopen axes assigned to the secondary feedback input (digitizing axes). Therefore, MC_Reference cannot be used to home digitizing axes at this time.

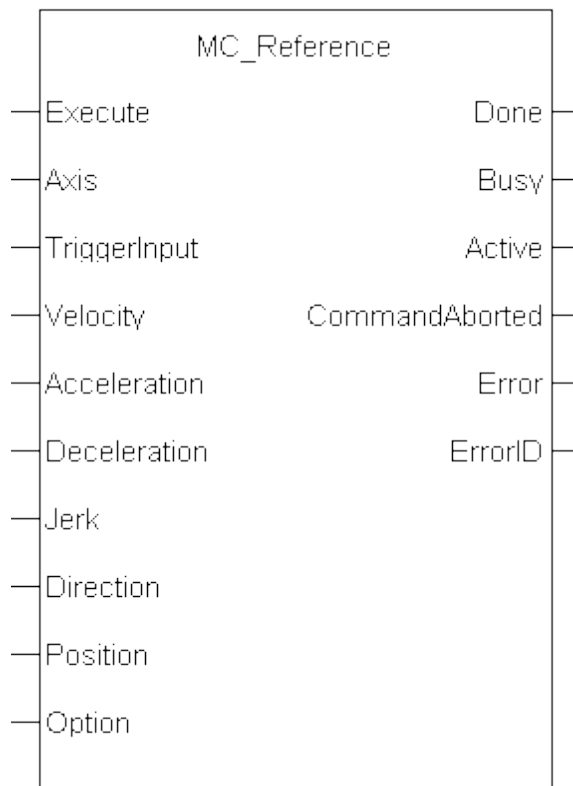


Figure 1-67: MC_Reference

Arguments

Input

Execute	Description	Requests to queue the MC_Reference move and arms reference trigger events
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	Name of a declared instance of the AXIS_REF library function.)
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
TriggerInput	Description	TRIGGER_REF structure defines the trigger INT InputID: 0 = Capture Engine 0 1 = Capture Engine 1 Range is [0,1] For information on configuring the capture engines, refer to AKD Capture Engine Configuration. INT Direction; 1 = rising edge of trigger, 2 = falling edge of trigger INT Trigid; must be zero

	Data type	TRIGGER_REF
	Range	See Description above
	Unit	n/a
	Default	—
Velocity	Description	Commanded velocity for the reference move
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	Description	Commanded acceleration for the reference move
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Deceleration	Description	Commanded deceleration for the reference move
	Data type	LREAL
	Range	—
	Unit	User unit/sec ²
	Default	—
Jerk	Description	Commanded jerk for the reference move (if zero, then trapezoidal acc/dec is used)
	Data type	LREAL
	Range	—
	Unit	User unit/sec ³
	Default	—
Direction	Description	Commanded Direction of the reference
	Data type	SINT
	Range	[0,1]
	Unit	n/a
	Default	—
Position	Description	Position of the axis at the reference location
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Option	Description	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
	Data type	SINT
	Range	[0,15]
	Unit	n/a
	Default	—
<u>Output</u>		
Done	Description	Indicates the reference move and position adjustment is complete
	Data type	BOOL
Busy	Description	Indicates this function block is executing
	Data type	BOOL
Active	Description	Indicates this move is the Active move
	Data type	BOOL
CommandAborted	Description	Indicates the move was aborted
	Data type	BOOL
Error	Description	Indicates an invalid input, or the move was terminated due to an error
	Data type	BOOL
ErrorID	Description	Indicates the error if the Error output is high
	Data type	INT

Usage

The following lists the 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.

The sequence of events of a PLCopen homing cycle consists of the following steps:

- Ensure Axis is not on Reference switch.
If a switch is used in the homing cycle for the event or precondition to the event, check to ensure the axis is not already tripping the switches that trigger the event and precondition. If it is, move the axis off the switches.
- Configure AKD capture engine
Configuration of the AKD capture engine is performed by writing drive CAN objects via SDO. It is accomplished with the ECATWriteSdo function. **The AKD Capture mode must be set to POSITION CAPTURE.**
The available configurations are discussed in paragraph "**AKD Capture Engine Configuration**". Example AKD capture engine configurations and reference examples are discussed in paragraph "**PLCopen Homing Methods**".
- Call the MC_REFERENCE function to initiate optional homing motion and to arm the AKD capture engine
The MC_Reference function block selects the trigger edge (rising or falling edge) and arm the capture. Then, it optionally moves the axis to the reference location as directed by inputs to this function. When the AKD indicates that the capture event has occurred, the coordinate system is shifted so that the reference position input to this function block is set to the reference location. Then, the reference motion is stopped.
- 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.
- Upon completion of the MC_Reference function block, the axis can be moved to the home position with a MC_MoveAbsolute function block.

TIP

Once the MC_Reference block is queued, but before it is completed, the cycle can be aborted with a MC_Halt or MC_Stop function block or by queuing a new motion function block with the Abort selected for buffer mode.

Related Functions

ECATWriteSdo

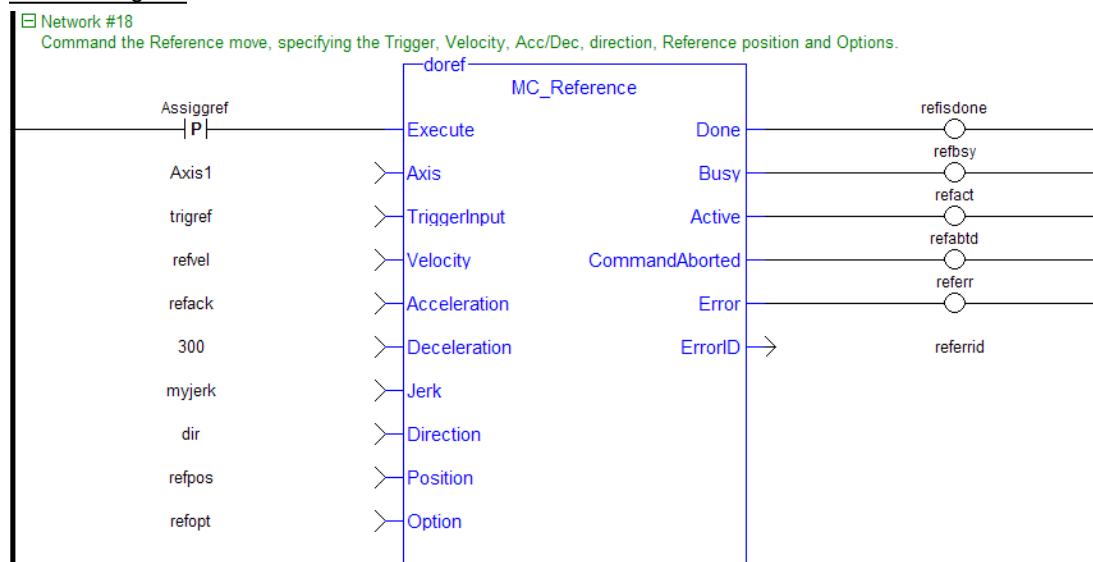
MC_MoveAbsolute

Example

Structured Text

```
(* 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 );
```

Ladder Diagram



1.2.6.2 MC_SetPosition (Function)

Description

This Function sets the axis position to the position specified at the Position input. It is a no-motion reference.

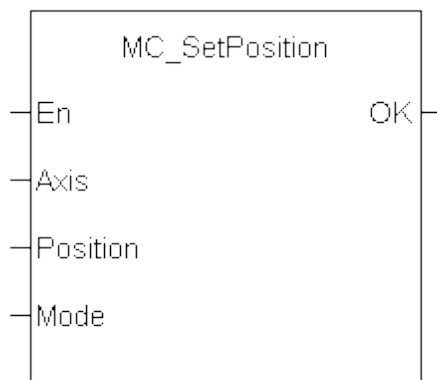


Figure 1-68: MC_SetPosition

Arguments

Input

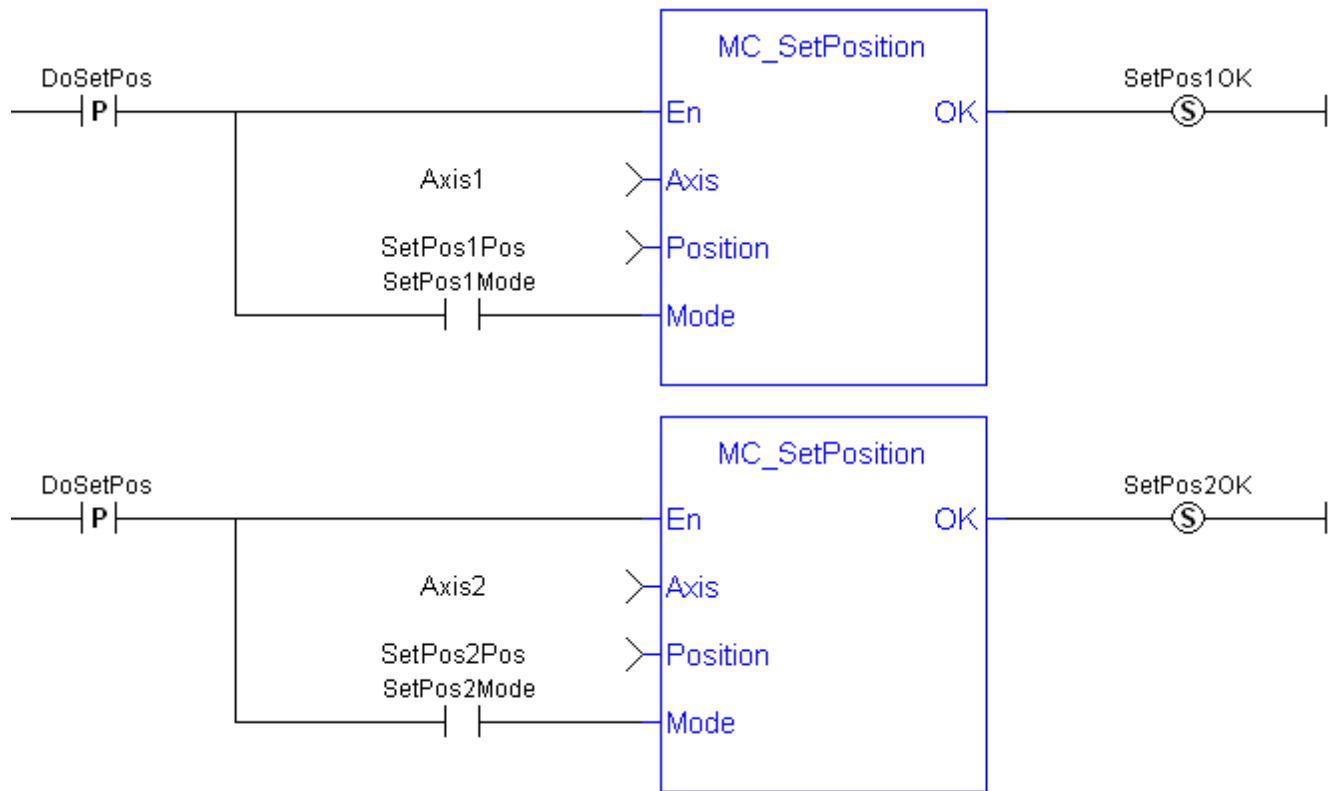
En	Description	Requests to change the axis position
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	Name of a declared instance of the AXIS_REF library function.)
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—

<u>Position</u>	Description	New axis position (absolute or relative)
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—
<u>Mode</u>	Description	LOW = value at Position is an absolute position HIGH = value at Position is a relative position
	Data type	BOOL
	Range	—
	Unit	n/a
	Default	—
<u>Output</u>		
<u>OK</u>	Description	HIGH = The function completed successfully LOW = The Axis input is invalid or Rollover position is non-zero and the Position input is outside the range [0,RolloverPosition]
	Data type	BOOL

Example**Structured Text**

```
(* MC_SetPosition ST example *)
Inst_MC_SetPosition( Axis1 , 0, 0 );
//Inst_MC_SetPosition is an instance of MC_SetPosition
function
```

Ladder Diagram



2 Fieldbus Library

2.1 EtherCAT Library	254
----------------------------	-----

2.1 EtherCAT Library

Name	Object Type	Description
DriveParamRead	SDO	Reads a drive parameter (ASCII format)
DriveParamWrite	SDO	Writes a drive parameter (ASCII format)
ECATGetObjVal	PDO	Reads cyclic drive parameter (String format) by returning the value of an EtherCAT PDO element
ECATGetStatus	PDO	Reads cyclic status word (Index 6041)
ECATReadData	PDO	Reads cyclic parameter (byte offset format)
ECATReadSdo	SDO	Reads parameter (32 bit format) using SDO command
ECATSetControl	PDO	Manipulates the state of a drive by setting its control word (Index 6040)
ECATWriteData	PDO	Writes cyclic parameter (byte offset format)
ECATWriteSdo	SDO	Writes parameter (32 bit format) using SDO command

Table 2-1: List of EtherCAT FB

The four 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.

Why use ECATReadSdo and ECATWriteSdo FBs?

The ECATReadSdo and ECATWriteSdo response time is faster and therefore is typically preferred over the DriveParamRead and DriveParamWrite.

Why use the DriveParam FBs?

The two reasons to prefer the DriveParam FBs are:

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

See some stats about the CPU load

Increase of CPU load when calling SDO function blocks

	Mid-range IPC (Celeron 1.2GHz)	High-range IPC (Core 2 Duo 1.86GHz)
Mean	60 μs	30 μs
Min	48 μ s	24 μ s
Max	64 μ s	38 μ s

(these values have been computed with the TraceTimes command)

2.1.1 EtherCAT Library - Drive

These function blocks are used to work with drive parameters that are 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.

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).

Reason

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

Result

The PLC code is overrunning the cycle duration. as explained in paragraph "**Tasking Model / Scheduling**".

As a consequence, you can see the following message in the Controller Log window:

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

Solution

When this happens we recommend to:

- 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.

See some stats about the FB execution time

- **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. Therefore the time to execute multiple commands is:
 $Number\ of\ commands \times Execution\ time\ of\ a\ single\ command$

2.1.1.1 DriveParamRead (Function Block)

Description

This function block reads a drive parameter by sending an ASCII command to a drive.

See also some **stats** about the execution time here.

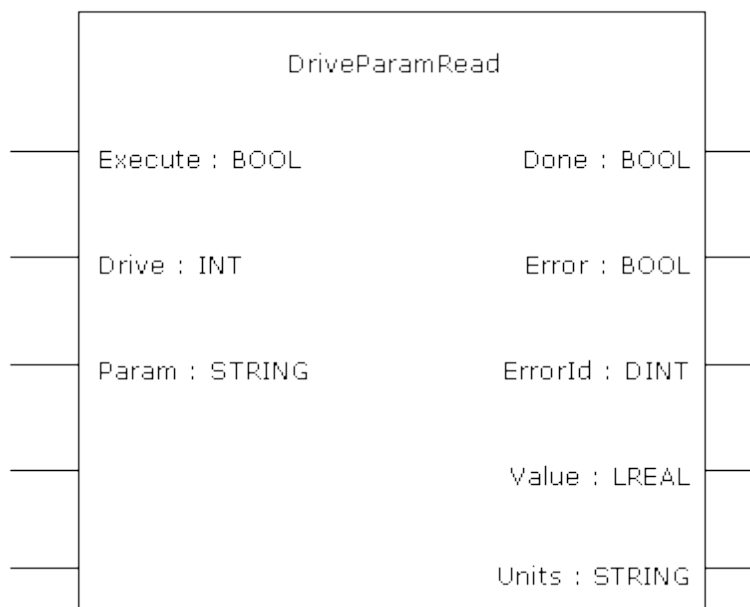


Figure 2-1: DriveParamRead

NOTE This function block uses *and reserves* the EtherCAT SDO Channel. The SDO Channel will remain reserved until the done output is "true". Therefore, 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 (the AKD GUI Views, for example) will be blocked.

Using this FB in SFC P0 or P1 steps is not recommended as 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.

Arguments

Input

Execute

Description

On the rising edge of Execute, a drive parameter is read.

NOTE The function block only handles 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.

Data type

BOOL

Range

0, 1

Unit

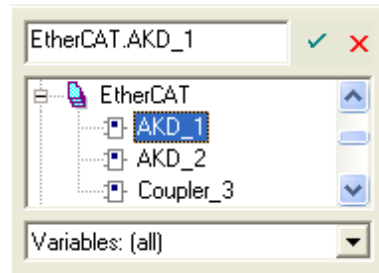
n/a

Default

—

Drive

Description The address of the drive from which data is read.
 The first node usually has the value '1001'. The second node usually has the value '1002'.
 Alternately, you can use the members of the EtherCAT structure to specify a drive's address when you create the variable.



Param

Data type INT
Range —
Unit n/a
Default —
Description The parameter to read.
Data type STRING
Range —
Unit n/a
Default —

Output

Done

Description Indicates whether the DriveParamRead function block has completed without error.

Data type BOOL
Unit n/a

Error

Description Indicates whether the DriveParamRead function block call has completed with error:

Data type BOOL
Unit n/a

ErrorID

Description The DriveParamRead error result if Error is TRUE (see list of Error Codes in table below). Upon success, Error is set to zero.

Data type DINT
Unit n/a

Error Code	Value, dec (hex)	Description
ECERR_OK	0	The SDO call succeeded
ECERR_DEVICE_INVALIDINDEX	1795 (0x703)	An invalid value for the Index input was specified
ECERR_DEVICE_INVALIDACCESS	1796 (0x704)	Reading of the variable is not permitted

Error Code	Value, dec (hex)	Description
ECERR_DEVICE_INVALIDDATA	1798 (0x706)	Invalid parameter value(s) in SDO index and/or sub-index
ECERR_DEVICE_NOTREADY	1799 (0x707)	device is not in a ready state, network is not in operational
ECERR_DEVICE_NOTFOUND	1804 (0x70C)	EtherCAT device not found
ECERR_DEVICE_SYNTAX	1805 (0x70D)	An unexpected error occurred
ECERR_DEVICE_INVALIDSTATE	1810 (0x712)	The EtherCAT device is in an invalid state
ECERR_DEVICE_TIMEOUT	1817 (0x719)	The EtherCAT device failed to respond, timing out
ECERR_DEVICE_INSERTMAILBOX	1826 (0x722)	Error while inserting the mailbox command into internal FIFO
ECERR_DEVICE_UNKNOWNMAILBOXCMD	1828 (0x724)	The master sent an unknown mailbox command to the slave
ECERR_DEVICE_INVALIDADDR	1832 (0x728)	Can't send a mailbox command to the specified slave
ECERR_DEVICE_INVALIDOFFSET	1827 (0x723)	An invalid value for the SubIndex input was specified
ECERR_DEVICE_PARAM_ACCESS_ERROR	1920 (0x780)	Unknown error occurred while accessing parameter
ECERR_DEVICE_PARAM_NOT_FOUND	1921 (0x781)	Parameter was not found
ECERR_DEVICE_PARAM_NOT_INTEGER	1922 (0x782)	Parameter is a floating-point value. Integer value required.
ECERR_DEVICE_VALUE_IS_NEGATIVE	1923 (0x783)	No negative values allowed. Value specified was negative.
ECERR_DEVICE_VALUE_OUT_OF_RANGE	1924 (0x784)	Value is out of data-range
ECERR_DEVICE_VALUE_GREATER_THAN_MAX	1925 (0x785)	Value bigger than maximum
ECERR_DEVICE_VALUE_LOWER_THAN_MIN	1926 (0x786)	Value lower than minimum
ECERR_CLIENT_ERROR	2048 (0x800)	Error in Mailbox response to a previously sent mailbox command
ECERR_CLIENT_TIMEOUT	2049 (0x801)	The SDO command timed out
ECERR_CLIENT_INVALIDPARM	2050 (0x802)	An invalid value was specified
ECERR_CLIENT_INVALIDSIZE	2051 (0x803)	An invalid value for the size input was specified

Table 2-2: List of EtherCAT Error Codes

Value	Description	The value of the drive parameter. Value is only set when the function block has successfully completed.
	Data type	LREAL
	Unit	n/a
Units	Description	The units of the drive parameter. Value is only set when the function block has successfully completed.
	Data type	STRING
	Unit	n/a

Usage

Use this FB to read drive parameters that are not supported by other function blocks. Examples would be motor temperature, drive bus voltage, Present drive limit settings, present regen loading, drive display, and fault history.

Related Functions

DriveParamWrite

Example

Structured Text

```
(* 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 example with animation

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

On Inst_DriveParamRead1.Done TRUE do
  Inst_DriveParamRead1( 0, 1001, 'PL.KP' );
  PositionProportionalGain 94.999000 := Inst_DriveParamRead1.Value 94.999000 ;
  ReadPropGain FALSE := 0;
End_DO;
```

2.1.1.2 DriveParamWrite (Function Block)

Description

This function block writes a drive parameter by sending an ASCII command to a drive.

See also some **stats** about the execution time here.

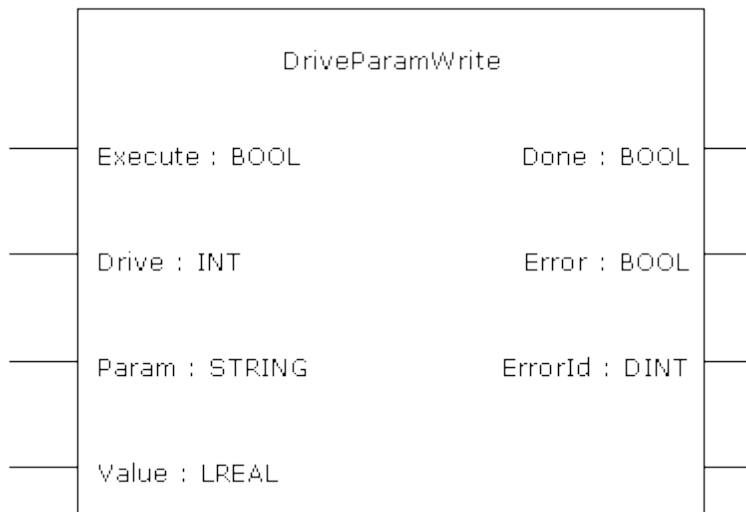


Figure 2-2: DriveParamWrite

NOTE

This function block uses *and reserves* the EtherCAT SDO Channel. The SDO Channel will remain reserved until the done output is "true". Therefore, 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 (the AKD GUI Views, for example) will be blocked.

Using this FB in SFC P0 or P1 steps is not recommended as 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.

Arguments

Input

Execute

Description

On the rising edge of Execute, a drive parameter is set.

NOTE

The function block only handles 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.

Data type

BOOL

Range

0, 1

Unit

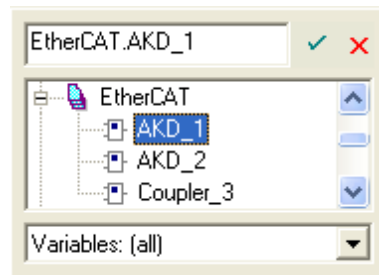
n/a

Default

—

Drive

Description The address of the drive to which data is written to.
 The first node usually has the value '1001'. The second node usually has the value '1002'.
 Alternately, you can use the members of the EtherCAT structure to specify a drive's address when you create the variable.



Data type INT
Range —
Unit n/a
Default —

Param

Description The parameter to write.
Data type STRING
Range —
Unit n/a
Default —

Value

Description The value to set the drive parameter to.
Data type LREAL
Range —
Unit n/a
Default —

Output

Done

Description Indicates whether the DriveParamWrite function block has completed without error.
Data type BOOL
Unit n/a

Error

Description Indicates whether the DriveParamWrite function block call has completed with error.
Data type BOOL
Unit n/a

ErrorID

Description The DriveParamWrite error result if Error is TRUE (see "Table 2-2: List of EtherCAT Error Codes " on page 259)

Upon success, Error is set to zero.

Data type DINT
Unit n/a

Usage

The function block can be used to change drive parameters. Common examples include tuning parameters and changing drive limits such as peak current.

Related Functions

DriveParamRead

Example

Structured Text

```
(* Write 58.000 to PL.KP of first AKD Drive on EtherCAT
network *)
```

```
Inst_DriveParamWrite( TRUE, 1001, 'PL.KP', 58 );
```

2.1.2 EtherCAT Library - SDO

These function blocks 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).

See some stats about the FB execution time

- **Max** time to consider when executing a single SDO command (i.e. before the Done output becomes True): **45 ms**

	Read SDO	Write SDO
Mean	10 ms	15 ms
Min	5 ms	10 ms
Max	45 ms	45 ms

- When sending multiple commands to a single drive, only one command can be sent at a time. Therefore 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. So you can be confident the function finishes execution in the same max time as to one drive

2.1.2.1 ECATReadSdo (Function Block)

Description

This function block reads a 32-bit word from I/O nodes using a CANopen SDO read command. It is typically used to query the status of inputs.

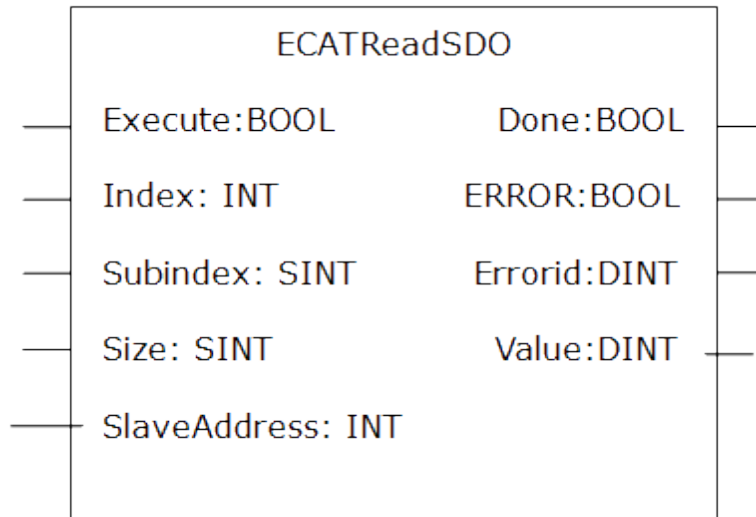


Figure 2-3: ECATReadSdo

State Diagram

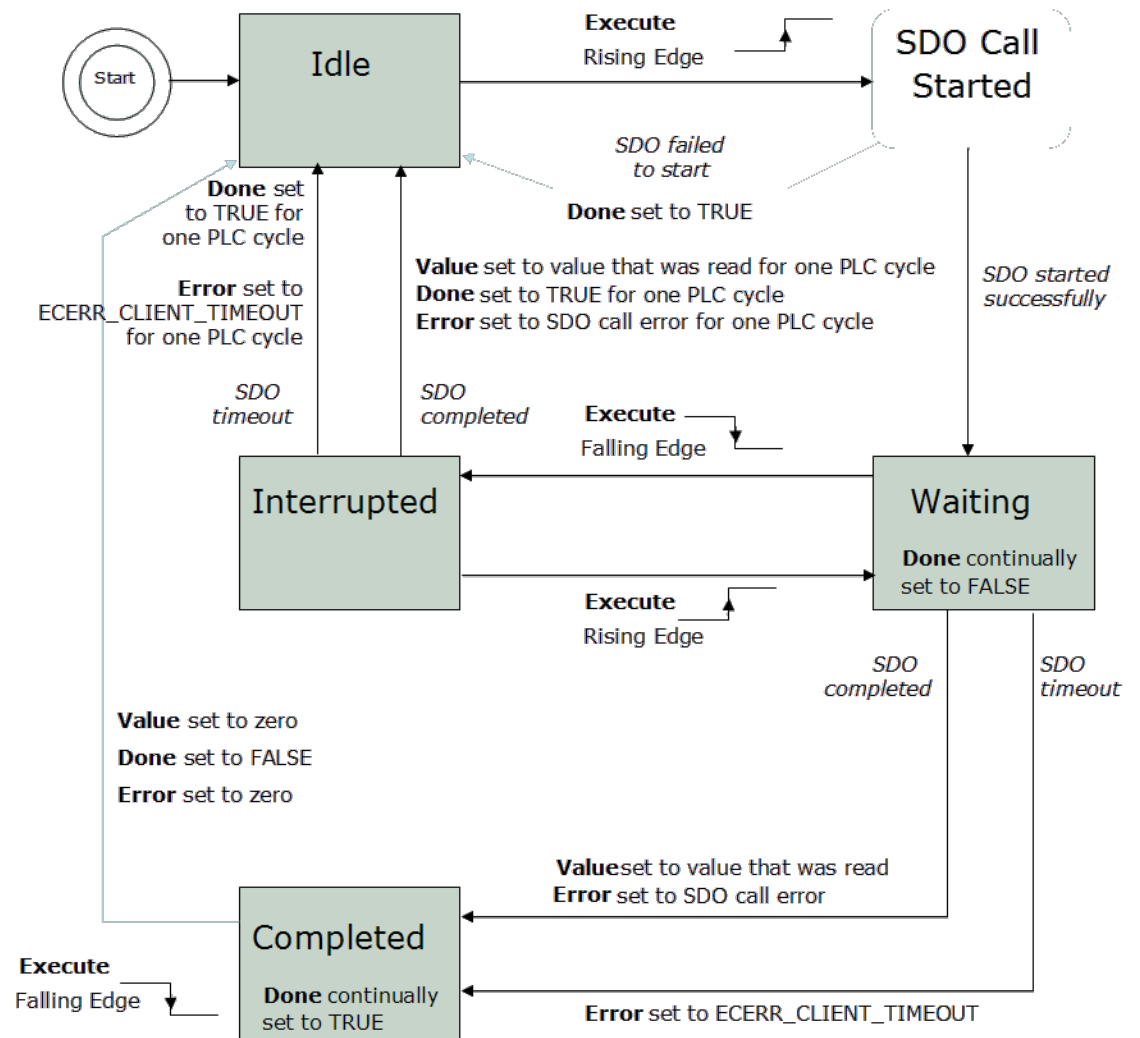


Figure 2-4: ECATReadSdo State Diagram

NOTE This function block uses *and reserves* the EtherCAT SDO Channel. The SDO Channel will remain reserved until the done output is "true". Therefore, 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 (the AKD GUI Views, for example) will be blocked.

Using this FB in SFC P0 or P1 steps is not recommended as 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.

Arguments

Input

Execute

Description On the rising edge of Execute, an SDO read command is issued.

NOTE 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.

Data type BOOL
 Range 0, 1
 Unit n/a
 Default —

Index

Description The object directory index of the data to be read.

For more details, refer to:

- Communication SDOs
- Manufacturer specific SDOs
- Profile specific SDOs

Data type INT
 Range —
 Unit n/a
 Default —

Subindex

Description The sub-index of the object directory variable to be read.

For more details, refer to:

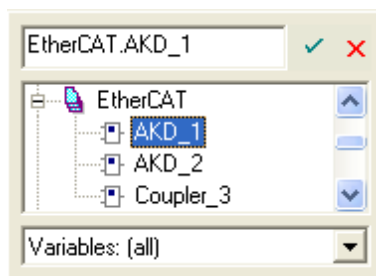
- Communication SDOs
- Manufacturer specific SDOs
- Profile specific SDOs

Data type SINT
 Range —
 Unit n/a
 Default —

Size

Description The size (number of bytes) to write.

	Data type	SINT
	Range	1 - 4
	Unit	n/a
	Default	—
SlaveAddress	Description	<p>The EtherCAT address of the slave from which data is written to.</p> <p>The first node usually has the value '1001'. The second node usually has the value '1002'.</p> <p>Alternately, you can use the members of the EtherCAT structure to specify a drive's address when you create the variable.</p>



	Data type	INT
	Range	—
	Unit	n/a
	Default	—
<u>Output</u>		
Done	Description	Indicates whether the SDO call has completed without error.
	Data type	BOOL
	Unit	n/a
Error	Description	Indicates whether the SDO call has completed with error:
	Data type	BOOL
	Unit	n/a
ErrorID	Description	<p>The SDO call error result, if Error is TRUE (see).</p> <p>Upon success, Error is set to zero.</p>
	Data type	DINT
	Unit	n/a

Value	Description	The value of the object directory variable being read.
		Value is only set when an SDO read command has successfully completed.
	Data type	DINT
	Unit	n/a

Related Functions

ECATWriteSDO

Example

Structured Text

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

2.1.2.2 ECATWriteSdo (Function Block)

Description

This function block writes a 32-bit word to I/O nodes using a CANopen SDO write command.

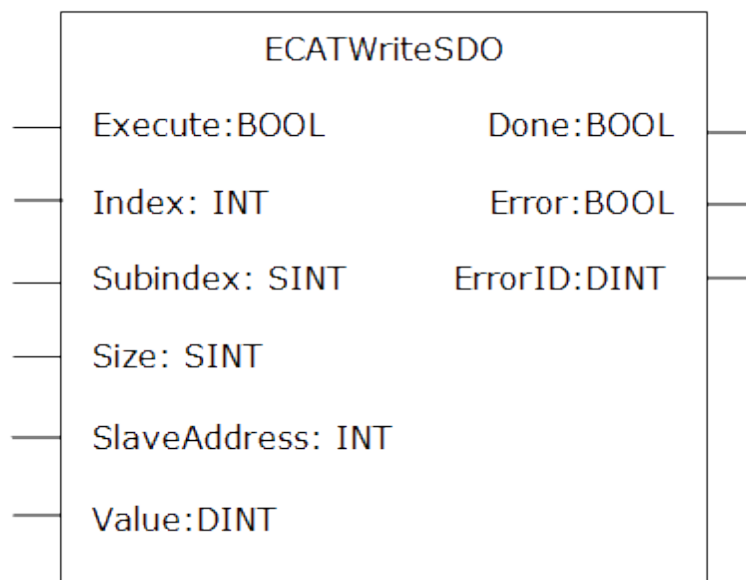


Figure 2-5: ECATWriteSdo

Arguments

Input

Execute

Description On the rising edge of Execute, an SDO write command will be issued.

NOTE

The function block will only handle 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 will not issue a second SDO command.

Data type BOOL
 Range 0, 1
 Unit n/a
 Default —

Index

Description The object directory index of the data to be written to.

For more details, refer to:

- Communication SDOs
- Manufacturer specific SDOs
- Profile specific SDOs

Data type INT
 Range —
 Unit n/a
 Default —

Subindex

Description The sub-index of the object directory variable to be written to.

For more details, refer to:

- Communication SDOs
- Manufacturer specific SDOs
- Profile specific SDOs

Data type SINT
 Range —
 Unit n/a
 Default —

Size

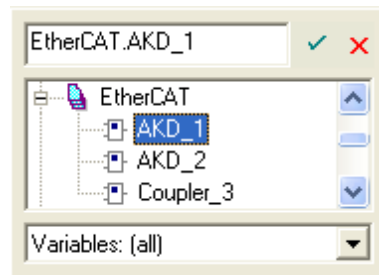
Description The size (number of bytes) to write.

Data type SINT
 Range 1 - 4
 Unit n/a
 Default —

SlaveAddress

Description The EtherCAT address of the slave from which data will be written to. The first node usually has the value '1001'. The second node usually has the value '1002'.

Alternately, you can use the members of the EtherCAT structure to specify a drive's address when you create the variable.



Data type INT
Range —
Unit n/a
Default —

Value

Description The value to write to the object directory variable.

Data type DINT
Range [-2147483648, 2147483648]
Unit n/a
Default —

Output

Done

Description Indicates whether the SDO call has completed without error.

Data type BOOL
Unit n/a

Error

Description Indicates whether the SDO call has completed with error:

Data type BOOL
Unit n/a

ErrorID

Description The SDO call error result, if Error is TRUE (see). Upon success, Error is set to zero.

Data type DINT
Unit n/a

Related Functions

ECATReadSDO

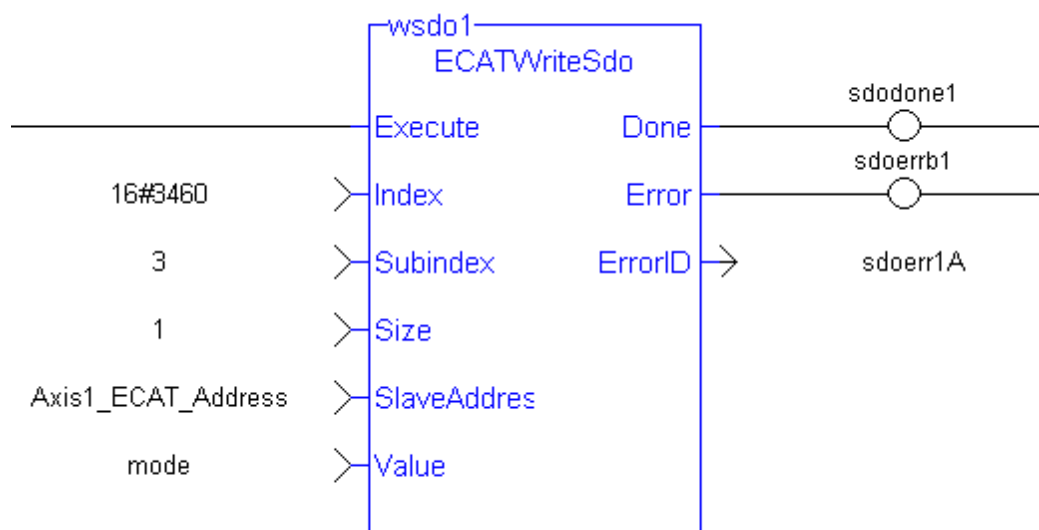
Example

Structured Text

```
(* Write 58.000 to PL.KP of first AKD Drive on EtherCAT
network *)
```

```
Inst_ECATWriteSdo( TRUE, 16#3542, 0, 4, 1001, 58000 );
```

Ladder Diagram



2.1.3 EtherCAT Library - Debug

The following function blocks support advanced functionality typically used for diagnostic support.

Most information available in these function blocks is also available in a ML and MC function block.

2.1.3.1 ECATReadData (Function)

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

Description

This function allows a direct access to the memory image of the EtherCAT frame which is sent or received when you need to debug your application. You 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, ECATReadData is used for more than just debug. It is used to get the status of the module (e.g. Stepper I/O slice).

Arguments

Input

<u>Offset</u>	Description	Offset from the beginning of the frame
	Data type	UINT
	Range	0- <i>size of frame</i> (maximum size of an Ethernet frame is 1500)
	Unit	n/a
	Default	—
<u>Nbytes</u>	Description	Number of bytes to read
	Data type	SINT
	Range	1, 2 or 4
	Unit	bytes
	Default	—
<u>Direction</u>	Description	Direction of the frame (true = output image, false = input image).
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—

Output

<u>Value</u>	Description	Value of the EtherCAT frame
	Data type	DINT
	Unit	n/a

Related Functions

ECATGetObjVal

Example

Structured Text

```
// Read 4 bytes starting at offset 26 of the output
image
```

```
Position := ECATReadData(26, 4, true);
```

2.1.3.2 ECATWriteData (Function)

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

Description

Modify the EtherCAT process image by directly writing values in it.

If you have a device other than the drive, ECATWriteData 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 your project is based on an external XML file because it contains unsupported EtherCAT Device.

Arguments

Input

Offset	Description	Byte offset from the beginning of the process image where data is to be written
	Data type	UINT
	Range	0 - 1500
	Unit	n/a
	Default	—
Nbytes	Description	Number of bytes to write
	Data type	SINT
	Range	1, 2 or 4
	Unit	bytes
	Default	—
Value	Description	Value to be written in the image. Only the number of bytes specified by Nbytes is copied.
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Default (.Q)	Description	True if data was written
	Data type	BOOL
	Unit	n/a

Related Functions

ECATReadData

2.1.3.3 ECATGetObjVal (Function)



This function block is deprecated as of KAS v2.7. The recommended best practice is to map a PLC variable to a PDO object.

2.1.4 EtherCAT Library - Status

The following function blocks support advanced functionality typically used for diagnostic support.

Most information available in these function blocks is also available in ML and MC function blocks.

2.1.4.1 ECATGetStatus (Function)

Description

Return the status word of the designated drive (SDO 0x6041).

The status machine for the status word corresponds to the CANopen status machine.

The Function Block receives the status word through the cyclic EtherCAT PDO communications. The status word is captured in every instance of fixed PDO mapping.

Arguments**Input**

Address	Description	EtherCAT address of the drive
	Data type	DINT
	Range	[0, 65535]
	Unit	n/a
	Default	—

Output

Status	Description	Status word of the drive as defined in the EtherCAT profile for the S300/S400/S600/S700. Compatible with CiA 402 definition of status word (CANopen object 0x6041).
	Data type	UINT
	Unit	n/a

Related Functions

ECATSetControl

Example**Structured Text**

```
(*****)
(* read EtherCAT axis status (Bit3: Fault, Bit7: Warning) *)
(*****)
ECATStatus := ECATGetStatus(AxisAddress); //Read the ECAT
Status Word (SDO 6041) of the Axis

IF AxisAddress > 1000 THEN
(*****)
(* timer to read cyclically SDOs *)
(*****)
```

2.1.4.2 ECATSetControl (Function)**Description**

Manipulate the state of a drive by setting its control word (SDO 06040).

The status machine for the control word corresponds to the CANopen status machine.

The Function Block transmits the control word through the cyclic EtherCAT PDO communications. The control word is captured in every instance of fixed PDO mapping.

Arguments

Input

Address	Description	EtherCAT address of the drive
	Data type	DINT
	Range	[0, 65535]
	Unit	n/a
	Default	—
Control	Description	Control word of the drive as defined in the EtherCAT profile for the S300/S400/S600/S700. Compatible with CiA 402 definition of control word (CANopen object 0x6040).
	Data type	UINT
	Range	—
	Unit	n/a
	Default	—

Output

Default (.Q)	Description	Returns true when function successfully executes (i.e. if the control word was successfully set)
	Data type	BOOL
	Unit	n/a

Related Functions

ECATGetStatus

This page intentionally left blank.

3 System Library

3.1 PrintMessage (Function)	278
-----------------------------------	-----

Name	Description
PrintMessage	Generate an output message in the log windows.
GetCtrlErrors	Get a list of the active errors and alarms on the controller.
ClearCtrlErrors	Clears the list of active errors and alarms on the controller.

Table 3-1: List of System FB

3.1 PrintMessage (Function)

3.1.1 Description

The PrintMessage block is used to generate a log message with any wanted strings in the log message window.

3.1.1.1 About the Source

PrintMessage use the PLC message type. So, to display all messages generated by PrintMessage, go to the log configuration and select the DEBUG level for the PLC source.

3.1.1.2 About the 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).

Keep in mind that only Error and Critical messages a generated by default. If you want to force the system to generate every message level, go into the log configuration and 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.

See at the configuration settings for more details.

3.1.2 Arguments

3.1.2.1 Input

Level	Description	Level of the logged message. In other words, the importance of it. Keep in mind that not all messages are displayed in the log windows by default. Only Error and Critical messages a displayed. So, in order to show lower level, it is needed to change the log settings. PrintMessage logs PLC messages.
	Data type	DINT
	Range	[0 , 4] Defines are: LEVEL_DEBUG, LEVEL_INFO, LEVEL_WARNING, LEVEL_ERROR, LEVEL_CRITICAL
	Unit	n/a
	Default	—
	Message	Description
	Data type	String
	Range	1 to 255 characters
	Unit	n/a
	Default	—

3.1.2.2 Output

Default (.Q)	Description	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a

3.1.3 Usage

```
PrintMessage( LEVEL_DEBUG, 'Message string to be logged'
);
```

3.1.4 Example

3.1.4.1 Structured Text

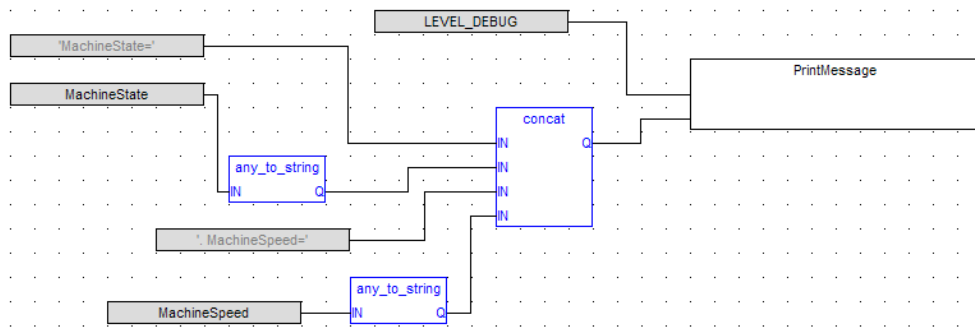
```
// It's possible to create a temporary variable with the
message.

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) ) );
```

3.1.4.2 Function Block Diagram



Index

A

actual velocity	180
ASCII	256

C

copyrights	2
curve	
synchronizer	142
cycle	
missed	256

D

debug	
EtherCAT	271
delay compensation	148
disclaimer	3
DriveParamRead	256
DriveParamWrite	260

E

ECATGetObjVal	273
ECATGetStatus	273
ECATReadData	271
ECATReadSdo	263
ECATSetControl	274
ECATWriteData	272
ECATWriteSdo	267
EtherCAT	
image	271-272
timeout	259
EtherCAT library	254
EtherCAT library function	
DriveParamRead	256
DriveParamWrite	260
ECATGetObjVal	273
ECATGetStatus	273
ECATReadData	271
ECATReadSdo	263
ECATSetControl	274
ECATWriteData	272
ECATWriteSdo	267

F

falling edge	145
fast input	149
feed-forward	
torque-	85
feedback position	69

H

homing 248

I

image EtherCAT 271-272

M

MC_AbortTrigger 171
 MC_CamIn 217
 MC_CamOut 224
 MC_CamTbiSelect 227
 MC_ClearFaults 156
 MC_CreateAxis 158
 MC_EStop 160
 MC_GearIn 229
 MC_GearInPos 233
 MC_GearOut 238
 MC_Halt 191
 MC_InitAxis 162
 MC_MoveAbsolute 194
 MC_MoveAdditive 200
 MC_MoveRelative 204
 MC_MoveSuperimp 208
 MC_MoveVelocity 212
 MC_Phasing 240
 MC_Power 164
 MC_ReadActPos 177
 MC_ReadActVel 179
 MC_ReadAxisErr 180
 MC_ReadBoolPar 182
 MC_ReadParam 184
 MC_ReadStatus 186
 MC_Reference 245
 MC_ResetError 166
 MC_SetOverride 215
 MC_SetPosition 250
 MC_Stop 168
 MC_SyncSlaves 243
 MC_TouchProbe 173
 MC_WriteBoolPar 188
 MC_WriteParam 190
 missing PLC cycles 256
 MLTrigGetPos 150
 MLTrigGetTime 152
 motion library 37-38
 adder 52
 Axis 69
 Block 45
 CAM Profile 71
 Comparator 71
 Convertor 82
 Delay 90
 Derivator 91
 Gear 96
 Integrator 98

Master	102
Phaser	125, 256, 263, 271, 273
Pipe	38
PMP	127
Sampler	128
State Machine	38
Synchronizer	134
Trigger	143
motion library function	
MC_AbortTrigger	171
MC_CamIn	217
MC_CamOut	224
MC_CamTblSelect	227
MC_ClearFaults	156
MC_CreateAxis	158
MC_EStop	160
MC_GearIn	229
MC_GearInPos	233
MC_GearOut	238
MC_Halt	191
MC_InitAxis	162
MC_MoveAbsolute	194
MC_MoveAdditive	200
MC_MoveRelative	204
MC_MoveSuperimp	208
MC_MoveVelocity	212
MC_Phasing	240
MC_Power	164
MC_ReadActPos	177
MC_ReadActVel	179
MC_ReadAxisErr	180
MC_ReadBoolPar	182
MC_ReadParam	184
MC_ReadStatus	186
MC_Reference	245
MC_ResetError	166
MC_SetOverride	215
MC_SetPosition	250
MC_Stop	168
MC_SyncSlaves	243
MC_TouchProbe	173
MC_WriteBoolPar	188
MC_WriteParam	190
P	
phasing	
PLCopen	240
synchronizer pipe block	134
position	
feedback position	69
reference position	70-71
PrintMessage	278
R	
reference position	70-71
registration	168
rising edge	145

S

servo axis	162
stats	254, 256, 263

T

timeout	
EtherCAT	259
torque feed-forward	85
trademarks	2

Global Support Contacts

North America

KOLLMORGEN

203A 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

Pempelfurtstraße 1
40880 Ratingen, Germany

Web: www.kollmorgen.com

Mail: technik@kollmorgen.com

Tel.: +49 - 2102 - 9394 - 0

Fax: +49 - 2102 - 9394 - 3155

Asia

KOLLMORGEN

Rm 2205, Scitech Tower, China
22 Jianguomen Wai Street

Web: www.kollmorgen.com

Mail: sales.asia@kollmorgen.com

Tel.: +86 - 400 666 1802

Fax: +86 - 10 6515 0263