

# Kollmorgen Automation Suite

## KAS Reference Manual - Motion Library



**Document Edition: J, June 2017**

Valid for KAS Software Revision 2.12

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 © 2009-17 Kollmorgen™

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

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™](#).
- [Kollmorgen™](#) is part of the [Fortive](#) Company.
- EnDat is a registered trademark of Dr. Johannes Heidenhain GmbH
- EtherCAT is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH
- Ethernet/IP is a registered trademark of ODVA, Inc.
- Ethernet/IP Communication Stack: copyright (c) 2009, Rockwell Automation
- HIPERFACE is a registered trademark of Max Stegmann GmbH
- PROFINET is a registered trademark of PROFIBUS and PROFINET International (PI)
- SIMATIC is a registered trademark of SIEMENS AG
- Windows is a registered trademark of Microsoft Corporation
- [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](#).
- SyCon® is a registered trademark of [Hilscher GmbH](#).
- Kollmorgen Automation Suite is based on the work of:
  - [Qwt](#) project (distributed under the [terms](#) of the GNU Lesser General Public License - see also [GPL terms](#))
  - [Zlib](#) software library
  - [Curl](#) software library
  - [Mongoose](#) software (distributed under the MIT License - [see terms](#))
  - JsonCpp software (distributed under the MIT License – [see terms](#))
  - [U-Boot](#), a universal boot loader is used by the PDMM (distributed under the [terms](#) of the GNU General Public License). The U-Boot source files, copyright notice, and readme are available on the distribution disk that is included with the PDMM.

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 J published on 6/22/2017) 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.

# 1 Table of Contents

<b>1 Table of Contents</b> .....	<b>3</b>
<b>2 Motion Library</b> .....	<b>80</b>
<b>2.1 Motion Library / Pipe Network</b> .....	<b>81</b>
2.1.1 Motion Library - Pipe Network .....	81
2.1.1.1 MLPipeAct .....	81
2.1.1.2.1 Description .....	81
2.1.1.3.2 Arguments .....	82
2.1.1.4.3.1 Input .....	82
2.1.1.5.4.2 Output .....	82
2.1.1.6.5.3 Return Type .....	82
2.1.1.7.6 Related Functions .....	82
2.1.1.8.7 Example .....	83
2.1.1.9.8.1 Structured Text .....	83
2.1.1.10.9.2 Ladder Diagram .....	83
2.1.1.11.10.3 Function Block Diagram .....	83
2.1.1.12 MLPipeAddBlock .....	84
2.1.1.13.1 Description .....	84
2.1.1.14.2 Arguments .....	84
2.1.1.15.3.1 Input .....	84
2.1.1.16.4.2 Output .....	85
2.1.1.17.5.3 Return Type .....	85
2.1.1.18.6 Related Functions .....	85
2.1.1.19.7 Example .....	85
2.1.1.20.8.1 Structured Text .....	85
2.1.1.21.9.2 Ladder Diagram .....	85
2.1.1.22.10.3 Function Block Diagram .....	85
2.1.1.23 MLPipeCreate .....	86
2.1.1.24.1 Description .....	86
2.1.1.25.2 Arguments .....	86
2.1.1.26.3.1 Input .....	86
2.1.1.27.4.2 Output .....	86
2.1.1.28.5 Related Functions .....	86
2.1.1.29.6 Example .....	86
2.1.1.30.7.1 Structured Text .....	86
2.1.1.31.8.2 Ladder Diagram .....	86
2.1.1.32.9.3 Function Block Diagram .....	87
2.1.1.33 MLPipeDeact .....	88
2.1.1.34.1 Description .....	88

---

2.1.1.35.2 Arguments .....	88
2.1.1.36.3.1 Input .....	88
2.1.1.37.4.2 Output .....	88
2.1.1.38.5.3 Return Type .....	88
2.1.1.39.6 Related Functions .....	89
2.1.1.40.7 Example .....	89
2.1.1.41.8.1 Structured Text .....	89
2.1.1.42.9.2 Ladder Diagram .....	89
2.1.1.43.10.3 Function Block Diagram .....	89
2.1.2 Motion Library - Block .....	90
2.1.2.1 MLBlkCreate .....	90
2.1.2.2.1 Description .....	90
2.1.2.3.2 Arguments .....	90
2.1.2.4.3.1 Input .....	90
2.1.2.5.4.2 Output .....	90
2.1.2.6.5 Related Functions .....	91
2.1.2.7.6 Example .....	91
2.1.2.8.7.1 Structured Text .....	91
2.1.2.9.8.2 Ladder Diagram .....	91
2.1.2.10.9.3 Function Block Diagram .....	91
2.1.2.11 MLBlkReadOutVal .....	92
2.1.2.12.1 Description .....	92
2.1.2.13.2 Arguments .....	92
2.1.2.14.3.1 Input .....	92
2.1.2.15.4.2 Output .....	92
2.1.2.16.5 Related Functions .....	92
2.1.2.17.6 Example .....	92
2.1.2.18.7.1 Structured Text .....	92
2.1.2.19.8.2 Ladder Diagram .....	92
2.1.2.20.9.3 Function Block Diagram .....	93
2.1.2.21 MLBlkReadModPos .....	94
2.1.2.22.1 Description .....	94
2.1.2.23.2 Arguments .....	94
2.1.2.24.3.1 Input .....	94
2.1.2.25.4.2 Output .....	94
2.1.2.26.5 Related Functions .....	94
2.1.2.27.6 Example .....	94
2.1.2.28.7.1 Structured Text .....	94
2.1.2.29.8.2 Ladder Diagram .....	95

2.1.2.30.9.3 Function Block Diagram .....	95
2.1.2.31 MLBlkIsReady .....	96
2.1.2.32.1 Description .....	96
2.1.2.33.2 Arguments .....	96
2.1.2.34.3.1 Input .....	96
2.1.2.35.4.2 Output .....	96
2.1.2.36.5.3 Return Type .....	96
2.1.2.37.6 Related Functions .....	96
2.1.2.38.7 Example .....	96
2.1.2.39.8.1 Structured Text .....	96
2.1.2.40.9.2 Ladder Diagram .....	96
2.1.2.41.10.3 Function Block Diagram .....	97
2.1.2.42 MLBlkWriteModPos .....	98
2.1.2.43.1 Description .....	98
2.1.2.44.2 Arguments .....	98
2.1.2.45.3.1 Input .....	98
2.1.2.46.4.2 Output .....	98
2.1.2.47.5.3 Return Type .....	98
2.1.2.48.6 Related Functions .....	98
2.1.2.49.7 Example .....	99
2.1.2.50.8.1 Structured Text .....	99
2.1.2.51.9.2 Ladder Diagram .....	99
2.1.2.52.10.3 Function Block Diagram .....	99
2.1.3 Motion Library - Adder .....	100
2.1.3.1 MLAddInit .....	100
2.1.3.2.1 Description .....	100
2.1.3.3.2 Arguments .....	100
2.1.3.4.3.1 Input .....	100
2.1.3.5.4.2 Output .....	101
2.1.3.6.5.3 Return Type .....	101
2.1.3.7.6 Related Functions .....	101
2.1.3.8.7 Example .....	102
2.1.3.9.8.1 Structured Text .....	102
2.1.3.10.9.2 Ladder Diagram .....	102
2.1.3.11.10.3 Function Block Diagram .....	102
2.1.3.12 MLAddReadOff1 .....	103
2.1.3.13.1 Description .....	103
2.1.3.14.2 Arguments .....	103
2.1.3.15.3.1 Input .....	103

---

2.1.3.16.4.2 Output .....	103
2.1.3.17.5 Related Functions .....	103
2.1.3.18.6 Example .....	103
2.1.3.19.7.1 Structured Text .....	103
2.1.3.20.8.2 Ladder Diagram .....	104
2.1.3.21.9.3 Function Block Diagram .....	104
2.1.3.22 MAddReadOff2 .....	105
2.1.3.23.1 Description .....	105
2.1.3.24.2 Arguments .....	105
2.1.3.25.3.1 Input .....	105
2.1.3.26.4.2 Output .....	105
2.1.3.27.5 Related Functions .....	105
2.1.3.28.6 Example .....	105
2.1.3.29.7.1 Structured Text .....	105
2.1.3.30.8.2 Ladder Diagram .....	106
2.1.3.31.9.3 Function Block Diagram .....	106
2.1.3.32 MAddReadRatio1 .....	107
2.1.3.33.1 Description .....	107
2.1.3.34.2 Arguments .....	107
2.1.3.35.3.1 Input .....	107
2.1.3.36.4.2 Output .....	107
2.1.3.37.5 Related Functions .....	107
2.1.3.38.6 Example .....	107
2.1.3.39.7.1 Structured Text .....	107
2.1.3.40.8.2 Ladder Diagram .....	108
2.1.3.41.9.3 Function Block Diagram .....	108
2.1.3.42 MAddReadRatio2 .....	109
2.1.3.43.1 Description .....	109
2.1.3.44.2 Arguments .....	109
2.1.3.45.3.1 Input .....	109
2.1.3.46.4.2 Output .....	109
2.1.3.47.5 Related Functions .....	109
2.1.3.48.6 Example .....	109
2.1.3.49.7.1 Structured Text .....	109
2.1.3.50.8.2 Ladder Diagram .....	110
2.1.3.51.9.3 Function Block Diagram .....	110
2.1.3.52 MAddWriteInput .....	111
2.1.3.53.1 Description .....	111
2.1.3.54.2 Arguments .....	111

---

2.1.3.55.3.1 Input .....	111
2.1.3.56.4.2 Output .....	112
2.1.3.57.5.3 Return Type .....	112
2.1.3.58.6 Related Functions .....	112
2.1.3.59.7 Example .....	112
2.1.3.60.8.1 Structured Text .....	112
2.1.3.61.9.2 Ladder Diagram .....	112
2.1.3.62.10.3 Function Block Diagram .....	112
2.1.3.63 MAddWriteOff1 .....	113
2.1.3.64.1 Description .....	113
2.1.3.65.2 Arguments .....	113
2.1.3.66.3.1 Input .....	113
2.1.3.67.4.2 Output .....	113
2.1.3.68.5.3 Return Type .....	113
2.1.3.69.6 Related Functions .....	113
2.1.3.70.7 Example .....	114
2.1.3.71.8.1 Structured Text .....	114
2.1.3.72.9.2 Ladder Diagram .....	114
2.1.3.73.10.3 Function Block Diagram .....	114
2.1.3.74 MAddWriteOff2 .....	115
2.1.3.75.1 Description .....	115
2.1.3.76.2 Arguments .....	115
2.1.3.77.3.1 Input .....	115
2.1.3.78.4.2 Output .....	115
2.1.3.79.5.3 Return Type .....	115
2.1.3.80.6 Related Functions .....	115
2.1.3.81.7 Example .....	116
2.1.3.82.8.1 Structured Text .....	116
2.1.3.83.9.2 Ladder Diagram .....	116
2.1.3.84.10.3 Function Block Diagram .....	116
2.1.3.85 MAddWriteRat1 .....	117
2.1.3.86.1 Description .....	117
2.1.3.87.2 Arguments .....	117
2.1.3.88.3.1 Input .....	117
2.1.3.89.4.2 Output .....	117
2.1.3.90.5.3 Return Type .....	117
2.1.3.91.6 Related Functions .....	117
2.1.3.92.7 Example .....	118
2.1.3.93.8.1 Structured Text .....	118

---

2.1.3.94.9.2 Ladder Diagram .....	118
2.1.3.95.10.3 Function Block Diagram .....	118
2.1.3.96 MLAddWriteRat2 .....	119
2.1.3.97.1 Description .....	119
2.1.3.98.2 Arguments .....	119
2.1.3.99.3.1 Input .....	119
2.1.3.100.4.2 Output .....	119
2.1.3.101.5.3 Return Type .....	119
2.1.3.102.6 Related Functions .....	119
2.1.3.103.7 Example .....	120
2.1.3.104.8.1 Structured Text .....	120
2.1.3.105.9.2 Ladder Diagram .....	120
2.1.3.106.10.3 Function Block Diagram .....	120
2.1.4 Motion Library - Axis .....	121
2.1.4.1 MLAxisAbs .....	122
2.1.4.2.1 Description .....	122
2.1.4.3.2 Arguments .....	122
2.1.4.4.3.1 Input .....	122
2.1.4.5 Position with Modulo On .....	123
2.1.4.6 Forcing the direction of rotation .....	123
2.1.4.7 Travel Speed Update with MLAxisAbs .....	124
2.1.4.8.1.1 Output .....	125
2.1.4.9.2 Related Functions .....	125
2.1.4.10.3 Example .....	125
2.1.4.11.4.1 Structured Text .....	125
2.1.4.12.5.2 Ladder Diagram .....	126
2.1.4.13.6.3 Function Block Diagram .....	126
2.1.4.14 MLAxisAdd .....	127
2.1.4.15.1 Description .....	127
2.1.4.16.2 Arguments .....	127
2.1.4.17.3.1 Input .....	127
2.1.4.18.4.2 Output .....	127
2.1.4.19.5 Related Functions .....	127
2.1.4.20.6 Example .....	127
2.1.4.21.7.1 Structured Text .....	127
2.1.4.22.8.2 Ladder Diagram .....	127
2.1.4.23.9.3 Function Block Diagram .....	128
2.1.4.24 MLAxisAddress .....	129
2.1.4.25.1 Description .....	129
2.1.4.26.2 Arguments .....	129



2.1.4.27.3.1 Input .....	129
2.1.4.28.4.2 Output .....	129
2.1.4.29.5 Example .....	129
2.1.4.30.6.1 Structured Text .....	129
2.1.4.31.7.2 Ladder Diagram .....	129
2.1.4.32.8.3 Function Block Diagram .....	129
2.1.4.33 MLAxisAddTq .....	130
2.1.4.34.1 Description .....	130
2.1.4.35.2 Arguments .....	130
2.1.4.36.3.1 Input .....	130
2.1.4.37.4.2 Output .....	130
2.1.4.38.5 Related Functions .....	130
2.1.4.39.6 Example .....	130
2.1.4.40.7.1 Structured Text .....	130
2.1.4.41.8.2 Ladder Diagram .....	130
2.1.4.42.9.3 Function Block Diagram .....	131
2.1.4.43 MLAxisCfgFastIn .....	132
2.1.4.44.1 Description .....	132
2.1.4.45.2 Arguments .....	132
2.1.4.46.3.1 Input .....	132
2.1.4.47.4.2 Output .....	132
2.1.4.48.5 Related Functions .....	133
2.1.4.49.6 See Also .....	133
2.1.4.50.7 Example .....	133
2.1.4.51.8.1 Structured Text .....	133
2.1.4.52.9.2 Ladder Diagram .....	133
2.1.4.53.10.3 Function Block Diagram .....	133
2.1.4.54 MLAxisCmdPos .....	134
2.1.4.55.1 Description .....	134
2.1.4.56.2 Arguments .....	134
2.1.4.57.3.1 Input .....	134
2.1.4.58.4.2 Output .....	134
2.1.4.59.5 Related Functions .....	134
2.1.4.60.6 Previous Function Name .....	134
2.1.4.61.7 Example .....	134
2.1.4.62.8.1 Structured Text .....	134
2.1.4.63.9.2 Ladder Diagram .....	134
2.1.4.64.10.3 Function Block Diagram .....	135
2.1.4.65 MLAxisCreate .....	136

---

2.1.4.66.1 Description .....	136
2.1.4.67.2 Arguments .....	136
2.1.4.68.3.1 Input .....	136
2.1.4.69.4.2 Output .....	136
2.1.4.70.5 Example .....	137
2.1.4.71.6.1 Structured Text .....	137
2.1.4.72.7.2 Ladder Diagram .....	137
2.1.4.73.8.3 Function Block Diagram .....	137
2.1.4.74 MAxisFBackPos .....	138
2.1.4.75.1 Description .....	138
2.1.4.76.2 Arguments .....	138
2.1.4.77.3.1 Input .....	138
2.1.4.78.4.2 Output .....	138
2.1.4.79.5 Related Functions .....	138
2.1.4.80.6 Example .....	138
2.1.4.81.7.1 Structured Text .....	138
2.1.4.82.8.2 Ladder Diagram .....	138
2.1.4.83.9.3 Function Block Diagram .....	139
2.1.4.84 MAxisGenEN .....	140
2.1.4.85.1 Description .....	140
2.1.4.86.2 Arguments .....	140
2.1.4.87.3.1 Input .....	140
2.1.4.88.4.2 Output .....	140
2.1.4.89.5 Related Functions .....	140
2.1.4.90.6 Example .....	140
2.1.4.91.7.1 Structured Text .....	140
2.1.4.92.8.2 Ladder Diagram .....	140
2.1.4.93.9.3 Function Block Diagram .....	141
2.1.4.94 MAxisGenIsEN .....	142
2.1.4.95.1 Description .....	142
2.1.4.96.2 Arguments .....	142
2.1.4.97.3.1 Input .....	142
2.1.4.98.4.2 Output .....	142
2.1.4.99.5 Related Functions .....	142
2.1.4.100.6 Example .....	142
2.1.4.101.7.1 Structured Text .....	142
2.1.4.102.8.2 Ladder Diagram .....	142
2.1.4.103.9.3 Function Block Diagram .....	142
2.1.4.104 MAxisGenIsRdy .....	143

---

2.1.4.105.1 Description .....	143
2.1.4.106.2 Arguments .....	143
2.1.4.107.3.1 Input .....	143
2.1.4.108.4.2 Output .....	143
2.1.4.109.5 Related Functions .....	143
2.1.4.110.6 Example .....	143
2.1.4.111.7.1 Structured Text .....	143
2.1.4.112.8.2 Ladder Diagram .....	143
2.1.4.113.9.3 Function Block Diagram .....	143
2.1.4.114 MLAxisGenPos .....	144
2.1.4.115.1 Description .....	144
2.1.4.116.2 Arguments .....	144
2.1.4.117.3.1 Input .....	144
2.1.4.118.4.2 Output .....	144
2.1.4.119.5 Related Functions .....	144
2.1.4.120.6 Example .....	144
2.1.4.121.7.1 Structured Text .....	144
2.1.4.122.8.2 Ladder Diagram .....	144
2.1.4.123.9.3 Function Block Diagram .....	145
2.1.4.124 MLAxisGenReadAcc .....	146
2.1.4.125.1 Description .....	146
2.1.4.126.2 Arguments .....	146
2.1.4.127.3.1 Input .....	146
2.1.4.128.4.2 Output .....	146
2.1.4.129.5 Related Functions .....	146
2.1.4.130.6 Example .....	146
2.1.4.131.7.1 Structured Text .....	146
2.1.4.132.8.2 Ladder Diagram .....	146
2.1.4.133.9.3 Function Block Diagram .....	147
2.1.4.134 MLAxisGenReadDec .....	148
2.1.4.135.1 Description .....	148
2.1.4.136.2 Arguments .....	148
2.1.4.137.3.1 Input .....	148
2.1.4.138.4.2 Output .....	148
2.1.4.139.5 Related Functions .....	148
2.1.4.140.6 Example .....	148
2.1.4.141.7.1 Structured Text .....	148
2.1.4.142.8.2 Ladder Diagram .....	148
2.1.4.143.9.3 Function Block Diagram .....	148

---

2.1.4.144	MLAxisGenReadSpd	150
2.1.4.145.1	Description	150
2.1.4.146.2	Arguments	150
2.1.4.147.3.1	Input	150
2.1.4.148.4.2	Output	150
2.1.4.149.5	Related Functions	150
2.1.4.150.6	Example	150
2.1.4.151.7.1	Structured Text	150
2.1.4.152.8.2	Ladder Diagram	150
2.1.4.153.9.3	Function Block Diagram	150
2.1.4.154	MLAxisGenWriteAcc	152
2.1.4.155.1	Description	152
2.1.4.156.2	Arguments	152
2.1.4.157.3.1	Input	152
2.1.4.158.4.2	Output	152
2.1.4.159.5	Related Functions	152
2.1.4.160.6	Example	152
2.1.4.161.7.1	Structured Text	152
2.1.4.162.8.2	Ladder Diagram	152
2.1.4.163.9.3	Function Block Diagram	153
2.1.4.164	MLAxisGenWriteDec	154
2.1.4.165.1	Description	154
2.1.4.166.2	Arguments	154
2.1.4.167.3.1	Input	154
2.1.4.168.4.2	Output	154
2.1.4.169.5	Related Functions	154
2.1.4.170.6	Example	154
2.1.4.171.7.1	Structured Text	154
2.1.4.172.8.2	Ladder Diagram	154
2.1.4.173.9.3	Function Block Diagram	155
2.1.4.174	MLAxisGenWriteSpd	156
2.1.4.175.1	Description	156
2.1.4.176.2	Arguments	156
2.1.4.177.3.1	Input	156
2.1.4.178.4.2	Output	156
2.1.4.179.5	Related Functions	156
2.1.4.180.6	Example	156
2.1.4.181.7.1	Structured Text	156
2.1.4.182.8.2	Ladder Diagram	156

2.1.4.183.9.3 Function Block Diagram .....	157
2.1.4.184 MLAxisInit .....	158
2.1.4.185.1 Description .....	158
2.1.4.186.2 Arguments .....	158
2.1.4.187.3.1 Input .....	158
2.1.4.188.4.2 Output .....	159
2.1.4.189.5 Example .....	159
2.1.4.190.6.1 Structured Text .....	159
2.1.4.191.7.2 Ladder Diagram .....	160
2.1.4.192.8.3 Function Block Diagram .....	160
2.1.4.193 MLAxisIsCncd .....	161
2.1.4.194.1 Description .....	161
2.1.4.195.2 Arguments .....	161
2.1.4.196.3.1 Input .....	161
2.1.4.197.4.2 Output .....	161
2.1.4.198.5 Example .....	161
2.1.4.199.6.1 Structured Text .....	161
2.1.4.200.7.2 Ladder Diagram .....	161
2.1.4.201.8.3 Function Block Diagram .....	161
2.1.4.202 MLAxisIsTriggered .....	162
2.1.4.203.1 Description .....	162
2.1.4.204.2 Arguments .....	162
2.1.4.205.3.1 Input .....	162
2.1.4.206.4.2 Output .....	162
2.1.4.207.5 Related Functions .....	162
2.1.4.208.6 Example .....	163
2.1.4.209.7.1 Structured Text .....	163
2.1.4.210.8.2 Ladder Diagram .....	163
2.1.4.211.9.3 Function Block Diagram .....	163
2.1.4.212 MLAxisMoveVel .....	164
2.1.4.213.1 Description .....	164
2.1.4.214.2 Arguments .....	164
2.1.4.215.3.1 Input .....	164
2.1.4.216.4.2 Output .....	164
2.1.4.217.5 Related Functions .....	164
2.1.4.218.6 Previous Function Name .....	164
2.1.4.219.7 Example .....	164
2.1.4.220.8.1 Structured Text .....	164
2.1.4.221.9.2 Ladder Diagram .....	165

---

2.1.4.222.10.3 Function Block Diagram .....	165
2.1.4.223 MLAxisPipePos .....	166
2.1.4.224.1 Description .....	166
2.1.4.225.2 Arguments .....	166
2.1.4.226.3.1 Input .....	166
2.1.4.227.4.2 Output .....	166
2.1.4.228.5 Related Functions .....	166
2.1.4.229.6 Example .....	166
2.1.4.230.7.1 Structured Text .....	166
2.1.4.231.8.2 Ladder Diagram .....	166
2.1.4.232.9.3 Function Block Diagram .....	167
2.1.4.233 MLAxisPower .....	168
2.1.4.234.1 Description .....	168
2.1.4.235.2 Arguments .....	168
2.1.4.236.3.1 Input .....	168
2.1.4.237.4.2 Output .....	168
2.1.4.238.5 Related Functions .....	168
2.1.4.239.6 Previous Function Name .....	168
2.1.4.240.7 Example .....	168
2.1.4.241.8.1 Structured Text .....	168
2.1.4.242.9.2 Ladder Diagram .....	168
2.1.4.243.10.3 Function Block Diagram .....	169
2.1.4.244 MLAxisPowerDOff .....	170
2.1.4.245.1 Description .....	170
2.1.4.246.2 Arguments .....	170
2.1.4.247.3.1 Input .....	170
2.1.4.248.4.2 Output .....	170
2.1.4.249.5 Related Functions .....	170
2.1.4.250.6 Example .....	170
2.1.4.251.7.1 Structured Text .....	170
2.1.4.252.8.2 Ladder Diagram .....	170
2.1.4.253.9.3 Function Block Diagram .....	170
2.1.4.254 MLAxisRatedTq .....	172
2.1.4.255.1 Description .....	172
2.1.4.256.2 Arguments .....	172
2.1.4.257.3.1 Input .....	172
2.1.4.258.4.2 Output .....	173
2.1.4.259.5 Related Functions .....	173
2.1.4.260.6 Example .....	173

---

2.1.4.261.7.1	Structured Text	173
2.1.4.262.8.2	Ladder Diagram	173
2.1.4.263.9.3	Function Block Diagram	173
2.1.4.264	MLAxisRead2ndFB	174
2.1.4.265.1	Description	174
2.1.4.266.2	Arguments	174
2.1.4.267.3.1	Input	174
2.1.4.268.4.2	Output	174
2.1.4.269.5	Related Functions	174
2.1.4.270.6	Example	174
2.1.4.271.7.1	Structured Text	174
2.1.4.272.8.2	Ladder Diagram	174
2.1.4.273.9.3	Function Block Diagram	174
2.1.4.274	MLAxisReadActPos	175
2.1.4.275.1	Description	175
2.1.4.276.2	Arguments	175
2.1.4.277.3.1	Input	175
2.1.4.278.4.2	Output	175
2.1.4.279.5	Related Functions	175
2.1.4.280.6	Previous Function Name	175
2.1.4.281.7	Example	175
2.1.4.282.8.1	Structured Text	175
2.1.4.283.9.2	Ladder Diagram	175
2.1.4.284.10.3	Function Block Diagram	176
2.1.4.285	MLAxisReadFBUnit	177
2.1.4.286.1	Description	177
2.1.4.287.2	Arguments	177
2.1.4.288.3.1	Input	177
2.1.4.289.4.2	Output	177
2.1.4.290.5	Example	177
2.1.4.291.6.1	Structured Text	177
2.1.4.292.7.2	Ladder Diagram	177
2.1.4.293.8.3	Function Block Diagram	177
2.1.4.294	MLAxisReadFEUU	178
2.1.4.295.1	Description	178
2.1.4.296.2	Arguments	178
2.1.4.297.3.1	Input	178
2.1.4.298.4.2	Output	178
2.1.4.299.5	Related Functions	178

---

2.1.4.300.6 Example .....	178
2.1.4.301.7.1 Structured Text .....	178
2.1.4.302.8.2 Ladder Diagram .....	178
2.1.4.303.9.3 Function Block Diagram .....	178
2.1.4.304 MAxisReadGenStatus .....	179
2.1.4.305.1 Description .....	179
2.1.4.306.2 Arguments .....	179
2.1.4.307.3.1 Input .....	179
2.1.4.308.4.2 Output .....	179
2.1.4.309.5 Related Functions .....	179
2.1.4.310.6 Previous Function Name .....	179
2.1.4.311.7 Example .....	179
2.1.4.312.8.1 Structured Text .....	179
2.1.4.313.9.2 Ladder Diagram .....	180
2.1.4.314.10.3 Function Block Diagram .....	180
2.1.4.315 MAxisReadModPos .....	181
2.1.4.316.1 Description .....	181
2.1.4.317.2 Arguments .....	181
2.1.4.318.3.1 Input .....	181
2.1.4.319.4.2 Output .....	181
2.1.4.320.5 Example .....	181
2.1.4.321.6.1 Structured Text .....	181
2.1.4.322.7.2 Ladder Diagram .....	181
2.1.4.323.8.3 Function Block Diagram .....	181
2.1.4.324 MAxisReadTq .....	182
2.1.4.325.1 Description .....	182
2.1.4.326.2 Arguments .....	182
2.1.4.327.3.1 Input .....	182
2.1.4.328.4.2 Output .....	182
2.1.4.329.5 Related Functions .....	182
2.1.4.330.6 Example .....	182
2.1.4.331.7.1 Structured Text .....	182
2.1.4.332.8.2 Ladder Diagram .....	182
2.1.4.333.9.3 Function Block Diagram .....	183
2.1.4.334 MAxisReadUUnits .....	184
2.1.4.335.1 Description .....	184
2.1.4.336.2 Arguments .....	184
2.1.4.337.3.1 Input .....	184
2.1.4.338.4.2 Output .....	184



---

2.1.4.339.5 Example .....	184
2.1.4.340.6.1 Structured Text .....	184
2.1.4.341.7.2 Ladder Diagram .....	184
2.1.4.342.8.3 Function Block Diagram .....	184
2.1.4.343 MAxisReadVel .....	185
2.1.4.344.1 Description .....	185
2.1.4.345.2 Arguments .....	185
2.1.4.346.3.1 Input .....	185
2.1.4.347.4.2 Output .....	185
2.1.4.348.5 Related Functions .....	185
2.1.4.349.6 Example .....	185
2.1.4.350.7.1 Structured Text .....	185
2.1.4.351.8.2 Ladder Diagram .....	185
2.1.4.352.9.3 Function Block Diagram .....	185
2.1.4.353 MAxisReAlgnRdy .....	186
2.1.4.354.1 Description .....	186
2.1.4.355.2 Arguments .....	186
2.1.4.356.3.1 Input .....	186
2.1.4.357.4.2 Output .....	186
2.1.4.358.5 Related Functions .....	186
2.1.4.359.6 Example .....	186
2.1.4.360.7.1 Structured Text .....	186
2.1.4.361.8.2 Ladder Diagram .....	186
2.1.4.362.9.3 Function Block Diagram .....	186
2.1.4.363 MAxisReAlign .....	187
2.1.4.364.1 Description .....	187
2.1.4.365.2 Arguments .....	187
2.1.4.366.3.1 Input .....	187
2.1.4.367.4.2 Output .....	188
2.1.4.368.5 Related Functions .....	188
2.1.4.369.6 Example .....	188
2.1.4.370.7.1 Structured Text .....	188
2.1.4.371.8.2 Ladder Diagram .....	188
2.1.4.372.9.3 Function Block Diagram .....	188
2.1.4.373 MAxisRel .....	189
2.1.4.374.1 Description .....	189
2.1.4.375.2 Arguments .....	189
2.1.4.376.3.1 Input .....	189
2.1.4.377.4.2 Output .....	189

---

2.1.4.378.5	Related Functions	189
2.1.4.379.6	Example	189
2.1.4.380.7.1	Structured Text	190
2.1.4.381.8.2	Ladder Diagram	190
2.1.4.382.9.3	Function Block Diagram	190
2.1.4.383	MLAxisResetErrors	191
2.1.4.384.1	Description	191
2.1.4.385.2	Arguments	191
2.1.4.386.3.1	Input	191
2.1.4.387.4.2	Output	191
2.1.4.388.5	Previous Function Name	191
2.1.4.389.6	Example	191
2.1.4.390.7.1	Structured Text	191
2.1.4.391.8.2	Ladder Diagram	191
2.1.4.392.9.3	Function Block Diagram	191
2.1.4.393	MLAxisRstFastIn	192
2.1.4.394.1	Description	192
2.1.4.395.2	Arguments	192
2.1.4.396.3.1	Input	192
2.1.4.397.4.2	Output	192
2.1.4.398.5	Related Functions	192
2.1.4.399.6	Example	192
2.1.4.400.7.1	Structured Text	192
2.1.4.401.8.2	Ladder Diagram	192
2.1.4.402.9.3	Function Block Diagram	193
2.1.4.403	MLAxisStatus	194
2.1.4.404.1	Description	194
2.1.4.405.2	Arguments	194
2.1.4.406.3.1	Input	194
2.1.4.407.4.2	Output	194
2.1.4.408.5	Example	195
2.1.4.409.6.1	Structured Text	195
2.1.4.410.7.2	Ladder Diagram	195
2.1.4.411.8.3	Function Block Diagram	195
2.1.4.412	MLAxisStop	196
2.1.4.413.1	Description	196
2.1.4.414.2	Arguments	196
2.1.4.415.3.1	Input	196
2.1.4.416.4.2	Output	196

2.1.4.417.5	Related Functions .....	197
2.1.4.418.6	Example .....	197
2.1.4.419.7.1	Structured Text .....	197
2.1.4.420.8.2	Ladder Diagram .....	198
2.1.4.421.9.3	Function Block Diagram .....	198
2.1.4.422	MLAxisTimeStamp .....	199
2.1.4.423.1	Description .....	199
2.1.4.424.2	Arguments .....	199
2.1.4.425.3.1	Input .....	199
2.1.4.426.4.2	Output .....	199
2.1.4.427.5	Related Functions .....	200
2.1.4.428.6	Example .....	200
2.1.4.429.7.1	Structured Text .....	200
2.1.4.430.8.2	Ladder Diagram .....	200
2.1.4.431.9.3	Function Block Diagram .....	200
2.1.4.432	MLAxisWriteModPos .....	201
2.1.4.433.1	Description .....	201
2.1.4.434.2	Arguments .....	201
2.1.4.435.3.1	Input .....	201
2.1.4.436.4.2	Output .....	201
2.1.4.437.5	Example .....	201
2.1.4.438.6.1	Structured Text .....	201
2.1.4.439.7.2	Ladder Diagram .....	201
2.1.4.440.8.3	Function Block Diagram .....	201
2.1.4.441	MLAxisWritePipPos .....	202
2.1.4.442.1	Description .....	202
2.1.4.443.2	Arguments .....	202
2.1.4.444.3.1	Input .....	202
2.1.4.445.4.2	Output .....	202
2.1.4.446.5	Related Functions .....	202
2.1.4.447.6	Example .....	202
2.1.4.448.7.1	Structured Text .....	202
2.1.4.449.8.2	Ladder Diagram .....	202
2.1.4.450.9.3	Function Block Diagram .....	202
2.1.4.451	MLAxisWritePos .....	204
2.1.4.452.1	Description .....	204
2.1.4.453.2	Arguments .....	205
2.1.4.454.3.1	Input .....	205
2.1.4.455.4.2	Output .....	205

---

2.1.4.456.5	Previous Function Name	205
2.1.4.457.6	Example	206
2.1.4.458.7.1	Structured Text	206
2.1.4.459.8.2	Ladder Diagram	206
2.1.4.460.9.3	Function Block Diagram	206
2.1.4.461	MLAxisWriteUUnits	207
2.1.4.462.1	Description	207
2.1.4.463.2	Arguments	207
2.1.4.464.3.1	Input	207
2.1.4.465.4.2	Output	207
2.1.4.466.5	Example	207
2.1.4.467.6.1	Structured Text	207
2.1.4.468.7.2	Ladder Diagram	207
2.1.4.469.8.3	Function Block Diagram	207
2.1.4.470	Usage Example of Axis Functions	208
2.1.5	Motion Library - Cam Profile	209
2.1.5.1	MLCamInit	210
2.1.5.2.1	Description	210
2.1.5.3.2	Arguments	210
2.1.5.4.3.1	Input	210
2.1.5.5.4.2	Output	211
2.1.5.6.5.3	Return Type	211
2.1.5.7.6	Related Functions	211
2.1.5.8.7	Example	211
2.1.5.9.8.1	Structured Text	211
2.1.5.10.9.2	Ladder Diagram	211
2.1.5.11.10.3	Function Block Diagram	211
2.1.5.12	MLCamSwitch	212
2.1.5.13.1	Description	212
2.1.5.14.2	Arguments	212
2.1.5.15.3.1	Input	212
2.1.5.16.4.2	Output	212
2.1.5.17.5.3	Return Type	212
2.1.5.18.6	Related Functions	212
2.1.5.19.7	Example	212
2.1.5.20.8.1	Structured Text	213
2.1.5.21.9.2	Ladder Diagram	213
2.1.5.22.10.3	Function Block Diagram	213
2.1.5.23	MLPrfReadIOffset	214
2.1.5.24.1	Description	214

---

2.1.5.25.2 Arguments .....	214
2.1.5.26.3.1 Input .....	214
2.1.5.27.4.2 Output .....	214
2.1.5.28.5 Related Functions .....	214
2.1.5.29.6 Example .....	214
2.1.5.30.7.1 Structured Text .....	214
2.1.5.31.8.2 Ladder Diagram .....	214
2.1.5.32.9.3 Function Block Diagram .....	215
2.1.5.33 MLPrfReadIScale .....	216
2.1.5.34.1 Description .....	216
2.1.5.35.2 Arguments .....	216
2.1.5.36.3.1 Input .....	216
2.1.5.37.4.2 Output .....	216
2.1.5.38.5 Related Functions .....	216
2.1.5.39.6 Previous Function Name .....	216
2.1.5.40.7 Example .....	216
2.1.5.41.8.1 Structured Text .....	216
2.1.5.42.9.2 Ladder Diagram .....	217
2.1.5.43.10.3 Function Block Diagram .....	217
2.1.5.44 MLPrfReadOOffset .....	218
2.1.5.45.1 Description .....	218
2.1.5.46.2 Arguments .....	218
2.1.5.47.3.1 Input .....	218
2.1.5.48.4.2 Output .....	218
2.1.5.49.5 Related Functions .....	218
2.1.5.50.6 Example .....	218
2.1.5.51.7.1 Structured Text .....	218
2.1.5.52.8.2 Ladder Diagram .....	218
2.1.5.53.9.3 Function Block Diagram .....	219
2.1.5.54 MLPrfReadOScale .....	220
2.1.5.55.1 Description .....	220
2.1.5.56.2 Arguments .....	220
2.1.5.57.3.1 Input .....	220
2.1.5.58.4.2 Output .....	220
2.1.5.59.5 Related Functions .....	220
2.1.5.60.6 Previous Function Name .....	220
2.1.5.61.7 Example .....	220
2.1.5.62.8.1 Structured Text .....	220
2.1.5.63.9.2 Ladder Diagram .....	221

---

2.1.5.64.10.3 Function Block Diagram .....	221
2.1.5.65 MLPfWriteIOffset .....	222
2.1.5.66.1 Description .....	222
2.1.5.67.2 Arguments .....	222
2.1.5.68.3.1 Input .....	222
2.1.5.69.4.2 Output .....	222
2.1.5.70.5.3 Return Type .....	222
2.1.5.71.6 Related Functions .....	222
2.1.5.72.7 Example .....	222
2.1.5.73.8.1 Structured Text .....	222
2.1.5.74.9.2 Ladder Diagram .....	223
2.1.5.75.10.3 Function Block Diagram .....	223
2.1.5.76 MLPfWriteIScale .....	224
2.1.5.77.1 Description .....	224
2.1.5.78.2 Arguments .....	224
2.1.5.79.3.1 Input .....	224
2.1.5.80.4.2 Output .....	224
2.1.5.81.5.3 Return Type .....	224
2.1.5.82.6 Related Functions .....	224
2.1.5.83.7 Previous Function Name .....	224
2.1.5.84.8 Example .....	224
2.1.5.85.9.1 Structured Text .....	225
2.1.5.86.10.2 Ladder Diagram .....	225
2.1.5.87.11.3 Function Block Diagram .....	225
2.1.5.88 MLPfWriteOOffset .....	226
2.1.5.89.1 Description .....	226
2.1.5.90.2 Arguments .....	226
2.1.5.91.3.1 Input .....	226
2.1.5.92.4.2 Output .....	226
2.1.5.93.5.3 Return Type .....	226
2.1.5.94.6 Related Functions .....	226
2.1.5.95.7 Example .....	226
2.1.5.96.8.1 Structured Text .....	226
2.1.5.97.9.2 Ladder Diagram .....	227
2.1.5.98.10.3 Function Block Diagram .....	227
2.1.5.99 MLPfWriteOScale .....	228
2.1.5.100.1 Description .....	228
2.1.5.101.2 Arguments .....	228
2.1.5.102.3.1 Input .....	228

---

2.1.5.103.4.2 Output .....	228
2.1.5.104.5.3 Return Type .....	228
2.1.5.105.6 Related Functions .....	228
2.1.5.106.7 Previous Function Name .....	228
2.1.5.107.8 Example .....	229
2.1.5.108.9.1 Structured Text .....	229
2.1.5.109.10.2 Ladder Diagram .....	229
2.1.5.110.11.3 Function Block Diagram .....	229
2.1.6 Motion Library - Comparator .....	230
2.1.6.1 MLCompCheck .....	230
2.1.6.2.1 Description .....	230
2.1.6.3.2 Arguments .....	230
2.1.6.4.3.1 Input .....	230
2.1.6.5.4.2 Output .....	230
2.1.6.6.5.3 Return Type .....	231
2.1.6.7.6 Related Functions .....	231
2.1.6.8.7 Example .....	231
2.1.6.9.8.1 Structured Text .....	231
2.1.6.10.9.2 Ladder Diagram .....	231
2.1.6.11.10.3 Function Block Diagram .....	231
2.1.6.12 MLComplnit .....	232
2.1.6.13.1 Description .....	232
2.1.6.14.2 Arguments .....	232
2.1.6.15.3.1 Input .....	232
2.1.6.16.4.2 Output .....	233
2.1.6.17.5.3 Return Type .....	233
2.1.6.18.6 Related Functions .....	233
2.1.6.19.7 Example .....	233
2.1.6.20.8.1 Structured Text .....	233
2.1.6.21.9.2 Ladder Diagram .....	233
2.1.6.22.10.3 Function Block Diagram .....	233
2.1.6.23 MLCompReadRef .....	235
2.1.6.24.1 Description .....	235
2.1.6.25.2 Arguments .....	235
2.1.6.26.3.1 Input .....	235
2.1.6.27.4.2 Output .....	235
2.1.6.28.5 Related Functions .....	235
2.1.6.29.6 Example .....	235
2.1.6.30.7.1 Structured Text .....	235

---

2.1.6.31.8.2 Ladder Diagram .....	235
2.1.6.32.9.3 Function Block Diagram .....	235
2.1.6.33 MLCompReset .....	236
2.1.6.34.1 Description .....	236
2.1.6.35.2 Arguments .....	236
2.1.6.36.3.1 Input .....	236
2.1.6.37.4.2 Output .....	236
2.1.6.38.5.3 Return Type .....	236
2.1.6.39.6 Related Functions .....	236
2.1.6.40.7 Example .....	236
2.1.6.41.8.1 Structured Text .....	236
2.1.6.42.9.2 Ladder Diagram .....	236
2.1.6.43.10.3 Function Block Diagram .....	236
2.1.6.44 MLCompWriteRef .....	238
2.1.6.45.1 Description .....	238
2.1.6.46.2 Arguments .....	238
2.1.6.47.3.1 Input .....	238
2.1.6.48.4.2 Output .....	239
2.1.6.49.5.3 Return Type .....	239
2.1.6.50.6 Related Functions .....	239
2.1.6.51.7 Example .....	239
2.1.6.52.8.1 Structured Text .....	239
2.1.6.53.9.2 Ladder Diagram .....	239
2.1.6.54.10.3 Function Block Diagram .....	239
2.1.6.55 Usage example of Comparator Functions .....	240
2.1.7 Motion Library - Convertor .....	242
2.1.7.1 MLCNVConnect .....	242
2.1.7.2.1 Description .....	242
2.1.7.3.2 Arguments .....	242
2.1.7.4.3.1 Input .....	242
2.1.7.5.4.2 Output .....	243
2.1.7.6.5.3 Return Type .....	243
2.1.7.7.6 Related Functions .....	243
2.1.7.8.7 Example .....	243
2.1.7.9.8.1 Structured Text .....	243
2.1.7.10.9.2 Ladder Diagram .....	243
2.1.7.11.10.3 Function Block Diagram .....	243
2.1.7.12 MLCNVConnectEx .....	244
2.1.7.13.1 Description .....	244
2.1.7.14.2 Arguments .....	244



---

2.1.7.15.3.1	Input	244
2.1.7.16.4.2	Output	245
2.1.7.17.5.3	Return Type	245
2.1.7.18.6	Related Functions	245
2.1.7.19.7	Example	245
2.1.7.20.8.1	Structured Text	245
2.1.7.21.9.2	Ladder Diagram	245
2.1.7.22.10.3	Function Block Diagram	246
2.1.7.23	MLCNVConECAT	247
2.1.7.24.1	Description	247
2.1.7.25.2	Arguments	247
2.1.7.26.3.1	Input	247
2.1.7.27.4.2	Output	248
2.1.7.28.5	Related Functions	248
2.1.7.29.6	Example	248
2.1.7.30.7.1	Structured Text	248
2.1.7.31.8.2	Ladder Diagram	248
2.1.7.32.9.3	Function Block Diagram	248
2.1.7.33	MLCNVDisconnect	249
2.1.7.34.1	Description	249
2.1.7.35.2	Arguments	249
2.1.7.36.3.1	Input	249
2.1.7.37.4.2	Output	249
2.1.7.38.5.3	Return Type	249
2.1.7.39.6	Related Functions	249
2.1.7.40.7	Example	249
2.1.7.41.8.1	Structured Text	249
2.1.7.42.9.2	Ladder Diagram	249
2.1.7.43.10.3	Function Block Diagram	250
2.1.7.44	MLCNVInit	251
2.1.7.45.1	Description	251
2.1.7.46.2	Arguments	251
2.1.7.47.3.1	Input	251
2.1.7.48.4.2	Output	251
2.1.7.49.5.3	Return Type	251
2.1.7.50.6	Related Functions	251
2.1.7.51.7	Example	251
2.1.7.52.8.1	Structured Text	251
2.1.7.53.9.2	Ladder Diagram	252

---

2.1.7.54.10.3 Function Block Diagram .....	252
2.1.8 Motion Library - Delay .....	253
2.1.8.1 MLDelayInit .....	253
2.1.8.2.1 Description .....	253
2.1.8.3.2 Arguments .....	253
2.1.8.4.3.1 Input .....	253
2.1.8.5.4.2 Example .....	253
2.1.8.6.5.3 Structured Text .....	253
2.1.8.7.6.4 Ladder Diagram .....	253
2.1.8.8.7.5 Function Block Diagram .....	254
2.1.9 Motion Library - Derivator .....	255
2.1.9.1 MLDerInit .....	255
2.1.9.2.1 Description .....	255
2.1.9.3.2 Arguments .....	255
2.1.9.4.3.1 Input .....	255
2.1.9.5.4.2 Output .....	256
2.1.9.6.5.3 Return Type .....	256
2.1.9.7.6 Related Functions .....	256
2.1.9.8.7 Example .....	256
2.1.9.9.8.1 Structured Text .....	256
2.1.9.10.9.2 Ladder Diagram .....	256
2.1.9.11.10.3 Function Block Diagram .....	256
2.1.9.12 MLDerReadInModPos .....	257
2.1.9.13.1 Description .....	257
2.1.9.14.2 Arguments .....	257
2.1.9.15.3.1 Input .....	257
2.1.9.16.4.2 Output .....	257
2.1.9.17.5 Related Functions .....	258
2.1.9.18.6 Example .....	258
2.1.9.19.7.1 Structured Text .....	258
2.1.9.20.8.2 Ladder Diagram .....	258
2.1.9.21.9.3 Function Block Diagram .....	258
2.1.9.22 MLDerWriteInModPos .....	259
2.1.9.23.1 Description .....	259
2.1.9.24.2 Arguments .....	259
2.1.9.25.3.1 Input .....	259
2.1.9.26.4.2 Output .....	260
2.1.9.27.5.3 Return Type .....	260
2.1.9.28.6 Related Functions .....	260
2.1.9.29.7 Example .....	260

2.1.9.30.8.1	Structured Text .....	260
2.1.9.31.9.2	Ladder Diagram .....	260
2.1.9.32.10.3	Function Block Diagram .....	260
2.1.10	Motion Library - Gear .....	261
2.1.10.1	Usage example of Gear Functions .....	261
2.1.10.2	MLGearInit .....	264
2.1.10.3.1	Description .....	264
2.1.10.4.2	Arguments .....	264
2.1.10.5.3.1	Input .....	264
2.1.10.6.4.2	Output .....	266
2.1.10.7.5.3	Return Type .....	266
2.1.10.8.6	Related Functions .....	266
2.1.10.9.7	Example .....	266
2.1.10.10.8.1	Structured Text .....	266
2.1.10.11.9.2	Ladder Diagram .....	266
2.1.10.12.10.3	Function Block Diagram .....	266
2.1.10.13	MLGearReadOffset .....	268
2.1.10.14.1	Description .....	268
2.1.10.15.2	Arguments .....	268
2.1.10.16.3.1	Input .....	268
2.1.10.17.4.2	Output .....	268
2.1.10.18.5	Related Functions .....	268
2.1.10.19.6	Example .....	268
2.1.10.20.7.1	Structured Text .....	268
2.1.10.21.8.2	Ladder Diagram .....	268
2.1.10.22.9.3	Function Block Diagram .....	268
2.1.10.23	MLGearReadOffSlp .....	269
2.1.10.24.1	Description .....	269
2.1.10.25.2	Arguments .....	269
2.1.10.26.3.1	Input .....	269
2.1.10.27.4.2	Output .....	269
2.1.10.28.5	Related Functions .....	269
2.1.10.29.6	Example .....	269
2.1.10.30.7.1	Structured Text .....	269
2.1.10.31.8.2	Ladder Diagram .....	269
2.1.10.32.9.3	Function Block Diagram .....	269
2.1.10.33	MLGearReadRatio .....	271
2.1.10.34.1	Description .....	271
2.1.10.35.2	Arguments .....	271
2.1.10.36.3.1	Input .....	271

---

2.1.10.37.4.2 Output .....	271
2.1.10.38.5 Related Functions .....	271
2.1.10.39.6 Example .....	271
2.1.10.40.7.1 Structured Text .....	271
2.1.10.41.8.2 Ladder Diagram .....	271
2.1.10.42.9.3 Function Block Diagram .....	271
2.1.10.43 MLGearReadRatSlp .....	272
2.1.10.44.1 Description .....	272
2.1.10.45.2 Arguments .....	272
2.1.10.46.3.1 Input .....	272
2.1.10.47.4.2 Output .....	272
2.1.10.48.5 Related Functions .....	272
2.1.10.49.6 Example .....	272
2.1.10.50.7.1 Structured Text .....	272
2.1.10.51.8.2 Ladder Diagram .....	272
2.1.10.52.9.3 Function Block Diagram .....	272
2.1.10.53 MLGearWriteOff .....	273
2.1.10.54.1 Description .....	273
2.1.10.55.2 Arguments .....	273
2.1.10.56.3.1 Input .....	273
2.1.10.57.4.2 Output .....	273
2.1.10.58.5.3 Return Type .....	273
2.1.10.59.6 Related Functions .....	273
2.1.10.60.7 Example .....	273
2.1.10.61.8.1 Structured Text .....	273
2.1.10.62.9.2 Ladder Diagram .....	274
2.1.10.63.10.3 Function Block Diagram .....	274
2.1.10.64 MLGearWriteOSlp .....	275
2.1.10.65.1 Description .....	275
2.1.10.66.2 Arguments .....	275
2.1.10.67.3.1 Input .....	275
2.1.10.68.4.2 Output .....	275
2.1.10.69.5.3 Return Type .....	275
2.1.10.70.6 Related Functions .....	275
2.1.10.71.7 Example .....	275
2.1.10.72.8.1 Structured Text .....	275
2.1.10.73.9.2 Ladder Diagram .....	276
2.1.10.74.10.3 Function Block Diagram .....	276
2.1.10.75 MLGearWriteRatio .....	277

---

2.1.10.76.1	Description .....	277
2.1.10.77.2	Arguments .....	277
2.1.10.78.3.1	Input .....	277
2.1.10.79.4.2	Output .....	277
2.1.10.80.5.3	Return Type .....	277
2.1.10.81.6	Related Functions .....	277
2.1.10.82.7	Example .....	277
2.1.10.83.8.1	Structured Text .....	277
2.1.10.84.9.2	Ladder Diagram .....	278
2.1.10.85.10.3	Function Block Diagram .....	278
2.1.10.86	MLGearWriteRatSlp .....	279
2.1.10.87.1	Description .....	279
2.1.10.88.2	Arguments .....	279
2.1.10.89.3.1	Input .....	279
2.1.10.90.4.2	Output .....	279
2.1.10.91.5.3	Return Type .....	279
2.1.10.92.6	Related Functions .....	280
2.1.10.93.7	Example .....	280
2.1.10.94.8.1	Structured Text .....	280
2.1.10.95.9.2	Ladder Diagram .....	280
2.1.10.96.10.3	Function Block Diagram .....	280
2.1.10.97.11.4	RatioSlope Offset .....	280
2.1.11	Motion Library - Integrator .....	281
2.1.11.1	MLIntInit .....	281
2.1.11.2.1	Description .....	281
2.1.11.3.2	Arguments .....	281
2.1.11.4.3.1	Input .....	281
2.1.11.5.4.2	Output .....	282
2.1.11.6.5.3	Return Type .....	282
2.1.11.7.6	Related Functions .....	282
2.1.11.8.7	Example .....	282
2.1.11.9.8.1	Structured Text .....	282
2.1.11.10.9.2	Ladder Diagram .....	282
2.1.11.11.10.3	Function Block Diagram .....	282
2.1.11.12	MLIntWriteOutVal .....	284
2.1.11.13.1	Description .....	284
2.1.11.14.2	Arguments .....	284
2.1.11.15.3.1	Input .....	284
2.1.11.16.4.2	Output .....	284

---

2.1.11.17.5.3 Return Type .....	284
2.1.11.18.6 Related Functions .....	284
2.1.11.19.7 Example .....	284
2.1.11.20.8.1 Structured Text .....	284
2.1.11.21.9.2 Ladder Diagram .....	285
2.1.11.22.10.3 Function Block Diagram .....	285
2.1.12 Motion Library - Master .....	286
2.1.12.1 Usage example of Master Functions .....	286
2.1.12.2 MLMstAbs .....	287
2.1.12.3.1 Description .....	287
2.1.12.4.2 Arguments .....	287
2.1.12.5.3.1 Input .....	287
2.1.12.6.4.2 Output .....	288
2.1.12.7.5 Related Functions .....	289
2.1.12.8.6 Examples .....	289
2.1.12.9.7.1 Structured Text .....	289
2.1.12.10.8.2 Ladder Diagram .....	289
2.1.12.11.9.3 Function Block Diagram .....	289
2.1.12.12 MLMstAdd .....	290
2.1.12.13.1 Description .....	290
2.1.12.14.2 Arguments .....	290
2.1.12.15.3.1 Input .....	290
2.1.12.16.4.2 Output .....	290
2.1.12.17.5 Related Functions .....	290
2.1.12.18.6 Example .....	290
2.1.12.19.7.1 Structured Text .....	290
2.1.12.20.8.2 Ladder Diagram .....	291
2.1.12.21.9.3 Function Block Diagram .....	291
2.1.12.22 MLMstForcePos .....	292
2.1.12.23.1 Description .....	292
2.1.12.24.2 Arguments .....	292
2.1.12.25.3.1 Input .....	292
2.1.12.26.4.2 Output .....	292
2.1.12.27.5 Related Functions .....	292
2.1.12.28.6 Example .....	292
2.1.12.29.7.1 Structured Text .....	292
2.1.12.30.8.2 Ladder Diagram .....	293
2.1.12.31.9.3 Function Block Diagram .....	293
2.1.12.32 MLMstInit .....	294
2.1.12.33.1 Description .....	294

2.1.12.34.2 Arguments .....	294
2.1.12.35.3.1 Input .....	294
2.1.12.36.4.2 Output .....	296
2.1.12.37.5 Example .....	296
2.1.12.38.6.1 Structured Text .....	296
2.1.12.39.7.2 Ladder Diagram .....	296
2.1.12.40.8.3 Function Block Diagram .....	296
2.1.12.41 MLMstReadAccel .....	297
2.1.12.42.1 Description .....	297
2.1.12.43.2 Arguments .....	297
2.1.12.44.3.1 Input .....	297
2.1.12.45.4.2 Output .....	297
2.1.12.46.5 Related Functions .....	297
2.1.12.47.6 Example .....	297
2.1.12.48.7.1 Structured Text .....	297
2.1.12.49.8.2 Ladder Diagram .....	297
2.1.12.50 Function Block Diagram .....	298
2.1.12.51 MLMstReadDecel .....	299
2.1.12.52.1 Description .....	299
2.1.12.53.2 Arguments .....	299
2.1.12.54.3.1 Input .....	299
2.1.12.55.4.2 Output .....	299
2.1.12.56.5 Example .....	299
2.1.12.57.6.1 Structured Text .....	299
2.1.12.58.7.2 Ladder Diagram .....	299
2.1.12.59.8.3 Function Block Diagram .....	300
2.1.12.60 MLMstReadInitPos .....	301
2.1.12.61.1 Description .....	301
2.1.12.62.2 Arguments .....	301
2.1.12.63.3.1 Input .....	301
2.1.12.64.4.2 Output .....	301
2.1.12.65.5 Example .....	301
2.1.12.66.6.1 Structured Text .....	301
2.1.12.67.7.2 Ladder Diagram .....	301
2.1.12.68.8.3 Function Block Diagram .....	302
2.1.12.69 MLMstReadSpeed .....	303
2.1.12.70.1 Description .....	303
2.1.12.71.2 Arguments .....	303
2.1.12.72.3.1 Input .....	303

---

2.1.12.73.4.2 Output .....	303
2.1.12.74.5 Related Functions .....	303
2.1.12.75.6 Example .....	303
2.1.12.76.7.1 Structured Text .....	303
2.1.12.77.8.2 Ladder Diagram .....	303
2.1.12.78.9.3 Function Block Diagram .....	304
2.1.12.79 MLMstRel .....	305
2.1.12.80.1 Description .....	305
2.1.12.81.2 Arguments .....	305
2.1.12.82.3.1 Input .....	305
2.1.12.83.4.2 Output .....	305
2.1.12.84.5 Related Functions .....	305
2.1.12.85.6 Example .....	305
2.1.12.86.7.1 Structured Text .....	305
2.1.12.87.8.2 Ladder Diagram .....	306
2.1.12.88.9.3 Function Block Diagram .....	306
2.1.12.89 MLMstRun .....	307
2.1.12.90.1 Description .....	307
2.1.12.91.2 Arguments .....	307
2.1.12.92.3.1 Input .....	307
2.1.12.93.4.2 Output .....	307
2.1.12.94.5 Related Functions .....	307
2.1.12.95.6 Example .....	307
2.1.12.96.7.1 Structured Text .....	307
2.1.12.97.8.2 Ladder Diagram .....	308
2.1.12.98.9.3 Function Block Diagram .....	308
2.1.12.99 MLMstStatus .....	309
2.1.12.100.1 Description .....	309
2.1.12.101.2 Arguments .....	309
2.1.12.102.3.1 Input .....	309
2.1.12.103.4.2 Output .....	309
2.1.12.104.5 Example .....	310
2.1.12.105.6.1 Structured Text .....	310
2.1.12.106.7.2 Ladder Diagram .....	310
2.1.12.107.8.3 Function Block Diagram .....	310
2.1.12.108 MLMstWriteAccel .....	311
2.1.12.109.1 Description .....	311
2.1.12.110.2 Arguments .....	311
2.1.12.111.3.1 Input .....	311



2.1.12.112.4.2 Output .....	311
2.1.12.113.5 Related Functions .....	311
2.1.12.114.6 Example .....	311
2.1.12.115.7.1 Structured Text .....	312
2.1.12.116.8.2 Ladder Diagram .....	312
2.1.12.117.9.3 Function Block Diagram .....	312
2.1.12.118 MLMstWriteDecel .....	313
2.1.12.119.1 Description .....	313
2.1.12.120.2 Arguments .....	313
2.1.12.121.3.1 Input .....	313
2.1.12.122.4.2 Output .....	313
2.1.12.123.5 Related Functions .....	313
2.1.12.124.6 Example .....	313
2.1.12.125.7.1 Structured Text .....	313
2.1.12.126.8.2 Ladder Diagram .....	314
2.1.12.127.9.3 Function Block Diagram .....	314
2.1.12.128 MLMstWriteInitPos .....	315
2.1.12.129.1 Description .....	315
2.1.12.130.2 Arguments .....	315
2.1.12.131.3.1 Input .....	315
2.1.12.132.4.2 Output .....	315
2.1.12.133.5 Example .....	315
2.1.12.134.6.1 Structured Text .....	315
2.1.12.135.7.2 Ladder Diagram .....	315
2.1.12.136.8.3 Function Block Diagram .....	316
2.1.12.137 MLMstWriteSpeed .....	317
2.1.12.138.1 Description .....	317
2.1.12.139.2 Arguments .....	317
2.1.12.140.3.1 Input .....	317
2.1.12.141.4.2 Output .....	317
2.1.12.142.5 Related Functions .....	317
2.1.12.143.6 Example .....	317
2.1.12.144.7.1 Structured Text .....	317
2.1.12.145.8.2 Ladder Diagram .....	318
2.1.12.146.9.3 Function Block Diagram .....	318
2.1.13 Motion Library - Phaser .....	319
2.1.13.1 Usage example of Phaser Functions .....	319
2.1.13.2 MLPhalnit .....	320
2.1.13.3.1 Description .....	320
2.1.13.4.2 Arguments .....	320

---

2.1.13.5.3.1 Input .....	320
2.1.13.6.4.2 Output .....	321
2.1.13.7.5 Related Functions .....	321
2.1.13.8.6 Example .....	321
2.1.13.9.7.1 Structured Text .....	321
2.1.13.10.8.2 Ladder Diagram .....	322
2.1.13.11.9.3 Function Block Diagram .....	322
2.1.13.12 MLPhaReadActPhase .....	323
2.1.13.13.1 Description .....	323
2.1.13.14.2 Arguments .....	323
2.1.13.15.3.1 Input .....	323
2.1.13.16.4.2 Output .....	323
2.1.13.17.5 Related Functions .....	323
2.1.13.18 MLPhaReadPhase .....	324
2.1.13.19.1 Description .....	324
2.1.13.20.2 Arguments .....	324
2.1.13.21.3.1 Input .....	324
2.1.13.22.4.2 Output .....	324
2.1.13.23.5 Example .....	324
2.1.13.24.6.1 Structured Text .....	324
2.1.13.25.7.2 Ladder Diagram .....	324
2.1.13.26.8.3 Function Block Diagram .....	324
2.1.13.27 MLPhaReadSlope .....	325
2.1.13.28.1 Description .....	325
2.1.13.29.2 Arguments .....	325
2.1.13.30.3.1 Input .....	325
2.1.13.31.4.2 Output .....	325
2.1.13.32.5 Related Functions .....	325
2.1.13.33.6 Example .....	325
2.1.13.34.7.1 Structured Text .....	325
2.1.13.35.8.2 Ladder Diagram .....	325
2.1.13.36.9.3 Function Block Diagram .....	325
2.1.13.37 MLPhaWritePhase .....	326
2.1.13.38.1 Description .....	326
2.1.13.39.2 Arguments .....	326
2.1.13.40.3.1 Input .....	326
2.1.13.41.4.2 Output .....	326
2.1.13.42.5 Related Functions .....	326
2.1.13.43.6 Example .....	326

---

2.1.13.44.7.1	Structured Text .....	326
2.1.13.45.8.2	Ladder Diagram .....	326
2.1.13.46.9.3	Function Block Diagram .....	327
2.1.13.47	MLPhaWriteSlope .....	328
2.1.13.48.1	Description .....	328
2.1.13.49.2	Arguments .....	328
2.1.13.50.3.1	Input .....	328
2.1.13.51.4.2	Output .....	328
2.1.13.52.5	Related Functions .....	328
2.1.13.53.6	Example .....	328
2.1.13.54.7.1	Structured Text .....	328
2.1.13.55.8.2	Ladder Diagram .....	328
2.1.13.56.9.3	Function Block Diagram .....	329
2.1.14	Motion Library - PMP .....	330
2.1.14.1	MLPmpAbs .....	330
2.1.14.2.1	Description .....	330
2.1.14.3.2	Arguments .....	330
2.1.14.4.3.1	Input .....	330
2.1.14.5.4.2	Output .....	331
2.1.14.6.5	Related Functions .....	331
2.1.14.7.6	Example .....	331
2.1.14.8.7.1	Structured Text .....	331
2.1.14.9.8.2	Ladder Diagram .....	331
2.1.14.10.9.3	Function Block Diagram .....	331
2.1.14.11	MLPmpForcePos .....	332
2.1.14.12.1	Description .....	332
2.1.14.13.2	Arguments .....	332
2.1.14.14.3.1	Input .....	332
2.1.14.15.4.2	Output .....	332
2.1.14.16.5	Related Functions .....	332
2.1.14.17.6	Example .....	332
2.1.14.18.7.1	Structured Text .....	332
2.1.14.19.8.2	Ladder Diagram .....	333
2.1.14.20.9.3	Function Block Diagram .....	333
2.1.14.21	MLPmplnit .....	334
2.1.14.22.1	Description .....	334
2.1.14.23.2	Arguments .....	334
2.1.14.24.3.1	Input .....	334
2.1.14.25.4.2	Output .....	336

---

2.1.14.26.5	Related Functions	336
2.1.14.27.6	Example	336
2.1.14.28.7.1	Structured Text	336
2.1.14.29.8.2	Ladder Diagram	336
2.1.14.30.9.3	Function Block Diagram	336
2.1.14.31	MLPmpReadAccel	338
2.1.14.32.1	Description	338
2.1.14.33.2	Arguments	338
2.1.14.34.3.1	Input	338
2.1.14.35.4.2	Output	338
2.1.14.36.5	Related Functions	338
2.1.14.37.6	Example	338
2.1.14.38.7.1	Structured Text	338
2.1.14.39.8.2	Ladder Diagram	338
2.1.14.40.9.3	Function Block Diagram	338
2.1.14.41	MLPmpReadFstSpd	339
2.1.14.42.1	Descriptions	339
2.1.14.43.2	Arguments	339
2.1.14.44.3.1	Input	339
2.1.14.45.4.2	Output	339
2.1.14.46.5	Related Functions	339
2.1.14.47.6	Example	339
2.1.14.48.7.1	Structured Text	339
2.1.14.49.8.2	Ladder Diagram	339
2.1.14.50.9.3	Function Block Diagram	340
2.1.14.51	MLPmpReadInitPos	341
2.1.14.52.1	Description	341
2.1.14.53.2	Arguments	341
2.1.14.54.3.1	Input	341
2.1.14.55.4.2	Output	341
2.1.14.56.5	Related Functions	341
2.1.14.57.6	Example	341
2.1.14.58.7.1	Structured Text	341
2.1.14.59.8.2	Ladder Diagram	341
2.1.14.60.9.3	Function Block Diagram	341
2.1.14.61	MLPmpReadJerk	342
2.1.14.62.1	Description	342
2.1.14.63.2	Arguments	342
2.1.14.64.3.1	Input	342

---

2.1.14.65.4.2 Output .....	342
2.1.14.66.5 Example .....	342
2.1.14.67.6.1 Structured Text .....	342
2.1.14.68.7.2 Ladder Diagram .....	342
2.1.14.69.8.3 Function Block Diagram .....	342
2.1.14.70 MLPmpReadLstSpd .....	343
2.1.14.71.1 Description .....	343
2.1.14.72.2 Arguments .....	343
2.1.14.73.3.1 Input .....	343
2.1.14.74.4.2 Output .....	343
2.1.14.75.5 Related Functions .....	343
2.1.14.76.6 Example .....	343
2.1.14.77.7.1 Structured Text .....	343
2.1.14.78.8.2 Ladder Diagram .....	343
2.1.14.79.9.3 Function Block Diagram .....	343
2.1.14.80 MLPmpRel .....	345
2.1.14.81.1 Description .....	345
2.1.14.82.2 Arguments .....	345
2.1.14.83.3.1 Input .....	345
2.1.14.84.4.2 Output .....	346
2.1.14.85.5 Related Functions .....	346
2.1.14.86.6 Example .....	346
2.1.14.87.7.1 Structured Text .....	346
2.1.14.88.8.2 Ladder Diagram .....	346
2.1.14.89.9.3 Function Block Diagram .....	346
2.1.14.90 MLPmpRun .....	347
2.1.14.91.1 Description .....	347
2.1.14.92.2 Arguments .....	347
2.1.14.93.3.1 Input .....	347
2.1.14.94.4.2 Output .....	347
2.1.14.95.5 Related Functions .....	347
2.1.14.96.6 Example .....	347
2.1.14.97.7.1 Structured Text .....	347
2.1.14.98.8.2 Ladder Diagram .....	348
2.1.14.99.9.3 Function Block Diagram .....	348
2.1.14.100 MLPmpStatus .....	349
2.1.14.101.1 Description .....	349
2.1.14.102.2 Arguments .....	349
2.1.14.103.3.1 Input .....	349

---

2.1.14.104.4.2 Output .....	349
2.1.14.105.5 Example .....	349
2.1.14.106.6.1 Structured Text .....	350
2.1.14.107.7.2 Ladder Diagram .....	350
2.1.14.108.8.3 Function Block Diagram .....	350
2.1.14.109 MLPmpWriteAccel .....	351
2.1.14.110.1 Description .....	351
2.1.14.111.2 Arguments .....	351
2.1.14.112.3.1 Input .....	351
2.1.14.113.4.2 Output .....	351
2.1.14.114.5 Related Functions .....	351
2.1.14.115.6 Example .....	351
2.1.14.116.7.1 Structured Text .....	351
2.1.14.117.8.2 Ladder Diagram .....	352
2.1.14.118.9.3 Function Block Diagram .....	352
2.1.14.119 MLPmpWriteFstSpd .....	353
2.1.14.120.1 Description .....	353
2.1.14.121.2 Arguments .....	353
2.1.14.122.3.1 Input .....	353
2.1.14.123.4.2 Output .....	353
2.1.14.124.5 Related Functions .....	353
2.1.14.125.6 Example .....	353
2.1.14.126.7.1 Structured Text .....	353
2.1.14.127.8.2 Ladder Diagram .....	354
2.1.14.128.9.3 Function Block Diagram .....	354
2.1.14.129 MLPmpWriteJerk .....	355
2.1.14.130.1 Description .....	355
2.1.14.131.2 Arguments .....	355
2.1.14.132.3.1 Input .....	355
2.1.14.133.4.2 Output .....	355
2.1.14.134.5 Related Functions .....	355
2.1.14.135.6 Example .....	355
2.1.14.136.7.1 Structured Text .....	355
2.1.14.137.8.2 Ladder Diagram .....	356
2.1.14.138.9.3 Function Block Diagram .....	356
2.1.14.139 MLPmpWriteLstSpd .....	357
2.1.14.140.1 Description .....	357
2.1.14.141.2 Arguments .....	357
2.1.14.142.3.1 Input .....	357

---

2.1.14.143.4.2 Output .....	357
2.1.14.144.5 Related Functions .....	357
2.1.14.145.6 Example .....	357
2.1.14.146.7.1 Structured Text .....	357
2.1.14.147.8.2 Ladder Diagram .....	358
2.1.14.148.9.3 Function Block Diagram .....	358
2.1.15 Motion Library - Sampler .....	359
2.1.15.1 MLSmpConnect .....	359
2.1.15.2.1 Description .....	359
2.1.15.3.2.1 Related Function Blocks .....	359
2.1.15.4.3 Arguments .....	359
2.1.15.5.4.1 Input .....	359
2.1.15.6.5.2 Output .....	359
2.1.15.7.6.3 Return Type .....	360
2.1.15.8.7 Example .....	360
2.1.15.9.8.1 Structured Text .....	360
2.1.15.10.9.2 Ladder Diagram .....	360
2.1.15.11.10.3 Function Block Diagram .....	360
2.1.15.12 MLSmpConECAT .....	361
2.1.15.13.1 Description .....	361
2.1.15.14.2.1 Related Function Blocks .....	361
2.1.15.15.3 Arguments .....	361
2.1.15.16.4.1 Input .....	361
2.1.15.17.5.2 Output .....	362
2.1.15.18.6.3 Return Type .....	362
2.1.15.19.7 Example .....	362
2.1.15.20.8.1 Structured Text .....	362
2.1.15.21.9.2 Ladder Diagram .....	362
2.1.15.22.10.3 Function Block Diagram .....	362
2.1.15.23 MLSmpConnectEx .....	363
2.1.15.24.1 Description .....	363
2.1.15.25 MLSmplnit .....	363
2.1.15.26.1 Description .....	363
2.1.15.27.2.1 Related Function Blocks .....	363
2.1.15.28.3 Arguments .....	363
2.1.15.29.4.1 Input .....	364
2.1.15.30.5.2 Output .....	364
2.1.15.31.6 Example .....	365
2.1.15.32.7.1 Structured Text .....	365

---

2.1.15.33.8.2 Ladder Diagram .....	365
2.1.15.34.9.3 Function Block Diagram .....	365
2.1.15.35 MLsmpConPLCAxis .....	366
2.1.15.36.1 Description .....	366
2.1.15.37.2 Arguments .....	366
2.1.15.38.3.1 Input .....	366
2.1.15.39.4.2 Output .....	366
2.1.15.40.5.3 Return Type .....	367
2.1.15.41.6 Example .....	367
2.1.15.42.7.1 Structured Text .....	367
2.1.15.43.8.2 Ladder Diagram .....	367
2.1.15.44.9.3 Function Block Diagram .....	367
2.1.15.45 MLsmpConPNAxis .....	368
2.1.15.46.1 Description .....	368
2.1.15.47.2 Arguments .....	368
2.1.15.48.3.1 Input .....	368
2.1.15.49.4.2 Output .....	369
2.1.15.50.5.3 Return Type .....	369
2.1.15.51.6 Example .....	369
2.1.15.52.7.1 Structured Text .....	369
2.1.15.53.8.2 Ladder Diagram .....	369
2.1.15.54.9.3 Function Block Diagram .....	369
2.1.16 Motion Library - Synchronizer .....	370
2.1.16.1 MLSynclnit .....	370
2.1.16.2.1 Description .....	370
2.1.16.3.2 Arguments .....	370
2.1.16.4.3.1 Input .....	370
2.1.16.5.4.2 Output .....	371
2.1.16.6.5 Related Functions .....	371
2.1.16.7.6 Example .....	371
2.1.16.8.7.1 Structured Text .....	371
2.1.16.9.8.2 Ladder Diagram .....	371
2.1.16.10.9.3 Function Block Diagram .....	371
2.1.16.11 MLSyncReadDeltaS .....	372
2.1.16.12.1 Description .....	372
2.1.16.13.2 Arguments .....	372
2.1.16.14.3.1 Input .....	372
2.1.16.15.4.2 Output .....	373
2.1.16.16.5 Related Functions .....	373



---

2.1.16.17.6 Example .....	373
2.1.16.18.7.1 Structured Text .....	373
2.1.16.19.8.2 Ladder Diagram .....	373
2.1.16.20.9.3 Function Block Diagram .....	373
2.1.16.21 MLSyncStart .....	374
2.1.16.22.1 Description .....	374
2.1.16.23.2 Arguments .....	374
2.1.16.24.3.1 Input .....	374
2.1.16.25.4.2 Output .....	374
2.1.16.26.5 Example .....	374
2.1.16.27.6.1 Structured Text .....	374
2.1.16.28.7.2 Ladder Diagram .....	374
2.1.16.29.8.3 Function Block Diagram .....	375
2.1.16.30 MLSyncStop .....	376
2.1.16.31.1 Description .....	376
2.1.16.32.2 Arguments .....	376
2.1.16.33.3.1 Input .....	376
2.1.16.34.4.2 Output .....	376
2.1.16.35.5 Example .....	376
2.1.16.36.6.1 Structured Text .....	376
2.1.16.37.7.2 Ladder Diagram .....	376
2.1.16.38.8.3 Function Block Diagram .....	377
2.1.16.39 MLSyncWriteDeltaS .....	378
2.1.16.40.1 Description .....	378
2.1.16.41.2 Arguments .....	378
2.1.16.42.3.1 Input .....	378
2.1.16.43.4.2 Output .....	379
2.1.16.44.5 Example .....	379
2.1.16.45.6.1 Structured Text .....	379
2.1.16.46.7.2 Ladder Diagram .....	379
2.1.16.47.8.3 Function Block Diagram .....	379
2.1.16.48 Usage example of Synchronizer Functions .....	380
2.1.17 Motion Library - Trigger .....	381
2.1.17.1 MLTrigClearFlag .....	381
2.1.17.2.1 Description .....	381
2.1.17.3.2 Arguments .....	381
2.1.17.4.3.1 Input .....	381
2.1.17.5.4.2 Output .....	381
2.1.17.6.5.3 Return Type .....	381
2.1.17.7.6 Related Functions .....	382

---

2.1.17.8 See Also .....	382
2.1.17.9.1 Example .....	382
2.1.17.10.2.1 Structured Text .....	382
2.1.17.11.3.2 Ladder Diagram .....	382
2.1.17.12.4.3 Function Block Diagram .....	382
2.1.17.13 MLTrigInit .....	383
2.1.17.14.1 Description .....	383
2.1.17.15.2 Arguments .....	383
2.1.17.16.3.1 Input .....	383
2.1.17.17.4.2 Output .....	384
2.1.17.18.5.3 Return Type .....	384
2.1.17.19.6 Related Functions .....	384
2.1.17.20 See Also .....	384
2.1.17.21.1 Example .....	384
2.1.17.22.2.1 Structured Text .....	384
2.1.17.23.3.2 Ladder Diagram .....	384
2.1.17.24.4.3 Function Block Diagram .....	385
2.1.17.25 MLTrigsTriggered .....	386
2.1.17.26.1 Description .....	386
2.1.17.27.2 Arguments .....	386
2.1.17.28.3.1 Input .....	386
2.1.17.29.4.2 Output .....	386
2.1.17.30.5.3 Return Type .....	387
2.1.17.31.6 Related Functions .....	387
2.1.17.32 See Also .....	387
2.1.17.33.1 Example .....	387
2.1.17.34.2.1 Ladder Diagram .....	387
2.1.17.35.3.2 Function Block Diagram .....	387
2.1.17.36 MLTrigReadDelay .....	388
2.1.17.37.1 Description .....	388
2.1.17.38.2.1 Input .....	388
2.1.17.39.3.2 Output .....	388
2.1.17.40.4 Related Functions .....	388
2.1.17.41 See Also .....	388
2.1.17.42 MLTrigReadPos .....	389
2.1.17.43.1 Description .....	389
2.1.17.44.2 Arguments .....	389
2.1.17.45.3.1 Input .....	389
2.1.17.46.4.2 Output .....	389

2.1.17.47.5 Related Functions .....	390
2.1.17.48 See Also .....	390
2.1.17.49.1 Previous Function Name .....	390
2.1.17.50.2 Example .....	390
2.1.17.51.3.1 Structured Text .....	390
2.1.17.52.4.2 Ladder Diagram .....	390
2.1.17.53.5.3 Function Block Diagram .....	390
2.1.17.54 MLTrigReadTime .....	391
2.1.17.55.1 Description .....	391
2.1.17.56.2 Arguments .....	391
2.1.17.57.3.1 Input .....	391
2.1.17.58.4.2 Output .....	391
2.1.17.59.5 Related Functions .....	392
2.1.17.60 See Also .....	392
2.1.17.61.1 Previous Function Name .....	392
2.1.17.62.2 Example .....	392
2.1.17.63.3.1 Ladder Diagram .....	392
2.1.17.64.4.2 Function Block Diagram .....	392
2.1.17.65 MLTrigSetEdge .....	393
2.1.17.66.1 Description .....	393
2.1.17.67.2 Arguments .....	393
2.1.17.68.3.1 Input .....	393
2.1.17.69.4.2 Output .....	393
2.1.17.70.5.3 Return Type .....	393
2.1.17.71 See Also .....	393
2.1.17.72.1 Examples .....	393
2.1.17.73.2.1 Function Block Diagram .....	393
2.1.17.74.3.2 Ladder Diagram .....	394
2.1.17.75.4.3 Structured Text .....	394
2.1.17.76 MLTrigWriteDelay .....	394
2.1.17.77.1 Description .....	394
2.1.17.78.2.1 Input .....	394
2.1.17.79.3.2 Output .....	394
2.1.17.80.4.3 Return Type .....	395
2.1.17.81.5 Related Functions .....	395
2.1.17.82 See Also .....	395
2.1.17.83 Usage example of Trigger Functions .....	396
<b>2.2 Motion Library / PLCopen .....</b>	<b>398</b>
2.2.1 Control Functions .....	399
2.2.1.1 MC_ClearFaults .....	399

---

2.2.1.2.1 Description .....	399
2.2.1.3.2 Arguments .....	399
2.2.1.4.3.1 Input .....	399
2.2.1.5.4.2 Output .....	400
2.2.1.6.5 Usage .....	400
2.2.1.7.6 Related Functions .....	400
2.2.1.8.7 Example .....	400
2.2.1.9.8.1 Structured Text .....	400
2.2.1.10.9.2 Function Block Diagram .....	400
2.2.1.11.10.3 Ladder Diagram .....	400
2.2.1.12 MC_CreateAxis .....	401
2.2.1.13.1 Description .....	401
2.2.1.14.2 Arguments .....	401
2.2.1.15.3.1 Input .....	401
2.2.1.16.4.2 Output .....	403
2.2.1.17.5 Example .....	403
2.2.1.18.6.1 Structured Text .....	403
2.2.1.19.7.2 Function Block Diagram .....	403
2.2.1.20.8.3 Ladder Diagram .....	404
2.2.1.21 MC_EStop .....	405
2.2.1.22.1 Description .....	405
2.2.1.23.2 Arguments .....	405
2.2.1.24.3.1 Input .....	405
2.2.1.25.4.2 Output .....	405
2.2.1.26.5 Usage .....	405
2.2.1.27.6 Related Functions .....	405
2.2.1.28.7 Example .....	405
2.2.1.29.8.1 Structured Text .....	405
2.2.1.30.9.2 Function Block Diagram .....	406
2.2.1.31.10.3 Ladder Diagram .....	406
2.2.1.32 MC_InitAxis .....	407
2.2.1.33.1 Description .....	407
2.2.1.34.2 Arguments .....	407
2.2.1.35.3.1 Input .....	407
2.2.1.36.4.2 Output .....	408
2.2.1.37.5 Example .....	408
2.2.1.38.6.1 Structured Text .....	408
2.2.1.39.7.2 Function Block Diagram .....	409
2.2.1.40.8.3 Ladder Diagram .....	409

2.2.1.41 MC_Power .....	410
2.2.1.42.1 Description .....	410
2.2.1.43.2 Arguments .....	410
2.2.1.44.3.1 Input .....	410
2.2.1.45.4.2 Output .....	411
2.2.1.46.5 Example .....	411
2.2.1.47.6.1 Structured Text .....	411
2.2.1.48.7.2 Ladder Diagram .....	412
2.2.1.49 MC_ErrorDescription .....	413
2.2.1.50.1 Arguments .....	413
2.2.1.51.2.1 Inputs .....	413
2.2.1.52.3.2 Outputs .....	413
2.2.1.53.4 Examples .....	414
2.2.1.54.5.1 Structured Text .....	414
2.2.1.55.6.2 IL .....	414
2.2.1.56.7.3 Function Block .....	414
2.2.1.57.8.4 Ladder Diagram .....	414
2.2.1.58 MC_ResetError .....	415
2.2.1.59.1 Description .....	415
2.2.1.60.2 Arguments .....	415
2.2.1.61.3.1 Input .....	415
2.2.1.62.4.2 Output .....	415
2.2.1.63.5 Example .....	415
2.2.1.64.6.1 Structured Text .....	415
2.2.1.65.7.2 Ladder Diagram .....	416
2.2.1.66 MC_Stop .....	417
2.2.1.67.1 Description .....	417
2.2.1.68.2 Time Diagram .....	417
2.2.1.69.3 Arguments .....	417
2.2.1.70.4.1 Input .....	417
2.2.1.71.5.2 Output .....	418
2.2.1.72.6 Example .....	419
2.2.1.73.7.1 Structured Text .....	419
2.2.1.74.8.2 Ladder Diagram .....	419
2.2.2 I/O Functions .....	420
2.2.2.1 MC_AbortTrigger .....	420
2.2.2.2.1 Description .....	420
2.2.2.3.2 Arguments .....	420
2.2.2.4.3.1 Input .....	420

---

2.2.2.5.4.2 Output .....	421
2.2.2.6.5 Usage .....	421
2.2.2.7.6 Related Functions .....	421
2.2.2.8.7 Example .....	421
2.2.2.9.8.1 Structured Text .....	421
2.2.2.10.9.2 Ladder Diagram .....	421
2.2.2.11 MC_TouchProbe .....	422
2.2.2.12.1 Description .....	422
2.2.2.13.2 Arguments .....	423
2.2.2.14.3.1 Input .....	423
2.2.2.15.4.2 Output .....	425
2.2.2.16.5 Usage .....	425
2.2.2.17.6 Limitations .....	426
2.2.2.18.7 Related Functions .....	426
2.2.2.19.8 See Also .....	426
2.2.2.20.9 Example .....	426
2.2.2.21.10.1 Structured Text .....	426
2.2.2.22.11.2 Ladder Diagram .....	426
2.2.3 Information Functions .....	427
2.2.3.1 MC_ReadActPos .....	427
2.2.3.2.1 Description .....	427
2.2.3.3.2 Arguments .....	427
2.2.3.4.3.1 Input .....	427
2.2.3.5.4.2 Output .....	427
2.2.3.6.5 Example .....	428
2.2.3.7.6.1 Structured Text .....	428
2.2.3.8.7.2 Ladder Diagram .....	428
2.2.3.9 MC_ReadActVel .....	429
2.2.3.10.1 Description .....	429
2.2.3.11.2 Arguments .....	429
2.2.3.12.3.1 Input .....	429
2.2.3.13.4.2 Output .....	429
2.2.3.14.5 Example .....	430
2.2.3.15.6.1 Structured Text .....	430
2.2.3.16.7.2 Ladder Diagram .....	430
2.2.3.17 MC_ReadAxisErr .....	431
2.2.3.18.1 Description .....	431
2.2.3.19.2 Arguments .....	431
2.2.3.20.3.1 Input .....	431

---

2.2.3.21.4.2 Output .....	431
2.2.3.22.5 Example .....	432
2.2.3.23.6.1 Structured Text .....	432
2.2.3.24.7.2 Ladder Diagram .....	432
2.2.3.25 MC_ReadBoolPar .....	433
2.2.3.26.1 Description .....	433
2.2.3.27.2 Arguments .....	433
2.2.3.28.3.1 Input .....	433
2.2.3.29.4.2 Output .....	433
2.2.3.30.5 Example .....	434
2.2.3.31.6.1 Structured Text .....	434
2.2.3.32.7.2 Ladder Diagram .....	434
2.2.3.33 MC_ReadParam .....	435
2.2.3.34.1 Description .....	435
2.2.3.35.2 Arguments .....	435
2.2.3.36.3.1 Input .....	435
2.2.3.37.4.2 Output .....	435
2.2.3.38.5 Example .....	436
2.2.3.39.6.1 Structured Text .....	436
2.2.3.40.7.2 Ladder Diagram .....	436
2.2.3.41 MC_ReadStatus .....	437
2.2.3.42.1 Description .....	437
2.2.3.43.2 Arguments .....	437
2.2.3.44.3.1 Input .....	437
2.2.3.45.4.2 Output .....	437
2.2.3.46.5 Example .....	438
2.2.3.47.6.1 Structured Text .....	438
2.2.3.48.7.2 Ladder Diagram .....	439
2.2.3.49 MC_WriteBoolPar .....	440
2.2.3.50.1 Description .....	440
2.2.3.51.2 Arguments .....	440
2.2.3.52.3.1 Input .....	440
2.2.3.53.4.2 Output .....	441
2.2.3.54.5 Example .....	441
2.2.3.55.6.1 Structured Text .....	441
2.2.3.56.7.2 Ladder Diagram .....	441
2.2.3.57 MC_WriteParam .....	442
2.2.3.58.1 Description .....	442
2.2.3.59.2 Arguments .....	442

---

2.2.3.60.3.1 Input .....	442
2.2.3.61.4.2 Output .....	443
2.2.3.62.5 Example .....	443
2.2.3.63.6.1 Structured Text .....	443
2.2.3.64.7.2 Ladder Diagram .....	443
2.2.4 PLCOpenMotion Functions .....	444
2.2.4.1 MC_Halt .....	444
2.2.4.2.1 Description .....	444
2.2.4.3.2 Time Diagram .....	444
2.2.4.4.3 Arguments .....	444
2.2.4.5.4.1 Input .....	445
2.2.4.6.5.2 Output .....	446
2.2.4.7.6 Example .....	446
2.2.4.8.7.1 Structured Text .....	446
2.2.4.9.8.2 Ladder Diagram .....	446
2.2.4.10 MC_MoveAbsolute .....	448
2.2.4.11.1 Description .....	448
2.2.4.12.2 Time Diagram .....	448
2.2.4.13.3 Arguments .....	449
2.2.4.14.4.1 Input .....	449
2.2.4.15.5.2 Output .....	451
2.2.4.16.6 Example .....	452
2.2.4.17.7.1 Structured Text .....	452
2.2.4.18.8.2 Ladder Diagram .....	452
2.2.4.19 MC_MoveAdditive .....	453
2.2.4.20.1 Description .....	453
2.2.4.21.2 Time Diagram .....	453
2.2.4.22.3 Arguments .....	454
2.2.4.23.4.1 Input .....	454
2.2.4.24.5.2 Output .....	456
2.2.4.25.6 Example .....	456
2.2.4.26.7.1 Structured Text .....	456
2.2.4.27.8.2 Ladder Diagram .....	456
2.2.4.28 MC_MoveRelative .....	458
2.2.4.29.1 Description .....	458
2.2.4.30.2 Time Diagram .....	458
2.2.4.31.3 Arguments .....	459
2.2.4.32.4.1 Input .....	459
2.2.4.33.5.2 Output .....	461



---

2.2.4.34.6 Example .....	461
2.2.4.35.7.1 Structured Text .....	461
2.2.4.36.8.2 Ladder Diagram .....	462
2.2.4.37 MC_MoveSuperimp .....	463
2.2.4.38.1 Description .....	463
2.2.4.39.2 Time Diagram .....	463
2.2.4.40.3 Arguments .....	464
2.2.4.41.4.1 Input .....	464
2.2.4.42.5.2 Output .....	466
2.2.4.43.6 Example .....	466
2.2.4.44.7.1 Structured Text .....	466
2.2.4.45.8.2 Ladder Diagram .....	466
2.2.4.46 MC_MoveVelocity .....	468
2.2.4.47.1 Description .....	468
2.2.4.48.2 Time Diagram .....	468
2.2.4.49.3 Arguments .....	468
2.2.4.50.4.1 Input .....	468
2.2.4.51.5.2 Output .....	470
2.2.4.52.6 Example .....	471
2.2.4.53.7.1 Structured Text .....	471
2.2.4.54.8.2 Ladder Diagram .....	471
2.2.4.55 MC_MoveContVel .....	472
2.2.4.56.1 Description .....	472
2.2.4.57.2 Time Diagram .....	472
2.2.4.58.3 Arguments .....	473
2.2.4.59.4.1 Input .....	473
2.2.4.60.5.2 Output .....	475
2.2.4.61.6 Example .....	475
2.2.4.62.7.1 Structured Text .....	475
2.2.4.63.8.2 Freeform Ladder Diagram .....	475
2.2.4.64 MC_SetOverride .....	477
2.2.4.65.1 Description .....	477
2.2.4.66.2 Arguments .....	477
2.2.4.67.3.1 Input .....	477
2.2.4.68.4.2 Output .....	477
2.2.4.69.5 Example .....	478
2.2.4.70.6.1 Structured Text .....	478
2.2.4.71.7.2 Ladder Diagram .....	478
2.2.5 Profile Functions .....	479

---

2.2.5.1 MC_CamIn .....	479
2.2.5.2.1 Description .....	479
2.2.5.3.2 Arguments .....	479
2.2.5.4.3.1 Input .....	479
2.2.5.5.4.2 Output .....	481
2.2.5.6.5 Usage .....	482
2.2.5.7.6 Related Functions .....	483
2.2.5.8.7 Examples .....	483
2.2.5.9.8.1 Structured Text .....	483
2.2.5.10.9.2 Ladder Diagram .....	483
2.2.5.11.10.3 Example 1 .....	484
2.2.5.12.11.4 Example 2 .....	484
2.2.5.13.12.5 Example 3 .....	485
2.2.5.14 MC_CamOut .....	486
2.2.5.15.1 Description .....	486
2.2.5.16.2 Arguments .....	486
2.2.5.17.3.1 Input .....	486
2.2.5.18.4.2 Output .....	487
2.2.5.19.5 Usage .....	488
2.2.5.20.6 Related Functions .....	488
2.2.5.21.7 Example .....	488
2.2.5.22.8.1 Structured Text .....	488
2.2.5.23.9.2 Ladder Diagram .....	488
2.2.5.24 MC_CamResumePos .....	490
2.2.5.25.1 Description .....	490
2.2.5.26.2 Related Functions .....	490
2.2.5.27.3 Arguments .....	490
2.2.5.28.4.1 Inputs .....	490
2.2.5.29.5.2 Outputs .....	491
2.2.5.30.6 Examples .....	491
2.2.5.31.7.1 ST .....	491
2.2.5.32.8.2 IL .....	492
2.2.5.33.9.3 FBD .....	492
2.2.5.34.10.4 FFLD .....	492
2.2.5.35 MC_CamStartPos .....	493
2.2.5.36.1 Description .....	493
2.2.5.37.2 Arguments .....	493
2.2.5.38.3.1 Inputs .....	493
2.2.5.39.4.2 Outputs .....	494

2.2.5.40.5	Examples	495
2.2.5.41.6.1	ST	495
2.2.5.42.7.2	IL	495
2.2.5.43.8.3	FBD	495
2.2.5.44.9.4	FFLD	495
2.2.5.45	MC_CamTblSelect	497
2.2.5.46.1	Description	497
2.2.5.47.2	Arguments	497
2.2.5.48.3.1	Input	497
2.2.5.49.4.2	Output	498
2.2.5.50.5	Usage	498
2.2.5.51.6	Related Functions	499
2.2.5.52.7	Example	499
2.2.5.53.8.1	Structured Text	499
2.2.5.54.9.2	Ladder Diagram	499
2.2.5.55	MC_GearIn	500
2.2.5.56.1	Description	500
2.2.5.57.2	Time Diagram	500
2.2.5.58.3	Arguments	501
2.2.5.59.4.1	Input	501
2.2.5.60.5.2	Output	503
2.2.5.61.6	Example	503
2.2.5.62.7.1	Structured Text	503
2.2.5.63.8.2	Ladder Diagram	503
2.2.5.64	MC_GearInPos	505
2.2.5.65.1	Description	505
2.2.5.66.2	Time Diagram	505
2.2.5.67.3	Arguments	506
2.2.5.68.4.1	Input	506
2.2.5.69.5.2	Output	509
2.2.5.70.6	Example	510
2.2.5.71.7.1	Example Description	510
2.2.5.72.8.2	Structured Text	510
2.2.5.73.9.3	Ladder Diagram	511
2.2.5.74	MC_GearOut	512
2.2.5.75.1	Description	512
2.2.5.76.2	Arguments	512
2.2.5.77.3.1	Input	512
2.2.5.78.4.2	Output	513

---

2.2.5.79.5 Example .....	514
2.2.5.80.6.1 Structured Text .....	514
2.2.5.81.7.2 Ladder Diagram .....	514
2.2.5.82 MC_Phasing .....	515
2.2.5.83.1 Description .....	515
2.2.5.84.2 Arguments .....	515
2.2.5.85.3.1 Input .....	515
2.2.5.86.4.2 Output .....	517
2.2.5.87.5 Example .....	517
2.2.5.88.6.1 Structured Text .....	517
2.2.5.89.7.2 Ladder Diagram .....	517
2.2.5.90 MC_SyncSlaves .....	519
2.2.5.91.1 Description .....	519
2.2.5.92.2 Arguments .....	519
2.2.5.93.3.1 Input .....	519
2.2.5.94.4.2 Output .....	520
2.2.5.95.5 Usage .....	520
2.2.5.96.6 Related Functions .....	520
2.2.5.97.7 Example .....	520
2.2.5.98.8.1 Structured Text .....	520
2.2.5.99.9.2 Ladder Diagram .....	521
2.2.6 Reference Functions .....	522
2.2.6.1 MC_Reference .....	522
2.2.6.2.1 Description .....	522
2.2.6.3.2 Arguments .....	522
2.2.6.4.3.1 Input .....	522
2.2.6.5.4.2 Output .....	524
2.2.6.6.5 Usage .....	525
2.2.6.7.6 Related Functions .....	526
2.2.6.8.7 Example .....	526
2.2.6.9.8.1 Structured Text .....	526
2.2.6.10.9.2 Ladder Diagram .....	526
2.2.6.11 MC_SetPos .....	527
2.2.6.12.1 Description .....	527
2.2.6.13.2 Arguments .....	527
2.2.6.14.3.1 Inputs .....	527
2.2.6.15.4.2 Outputs .....	528
2.2.6.16.5 Example .....	528
2.2.6.17.6.1 Structured Text .....	528

2.2.6.18.7.2 Ladder Diagram .....	529
2.2.6.19 MC_SetPosition .....	530
2.2.6.20.1 Description .....	530
2.2.7 Registration Function Blocks .....	530
2.2.7.1 MC_MachRegist .....	530
2.2.7.2 Description .....	530
2.2.7.3.1 Arguments .....	531
2.2.7.4.2.1 Input .....	531
2.2.7.5.3.2 Outputs .....	535
2.2.7.6.4 Related Functions .....	536
2.2.7.7.5 Examples .....	536
2.2.7.8.6.1 Function Block .....	536
2.2.7.9.7.2 Instruction List .....	536
2.2.7.10.8.3 Ladder Diagram .....	536
2.2.7.11.9.4 Structured Text .....	536
2.2.7.12 MC_MarkRegist .....	537
2.2.7.13.1 Description .....	537
2.2.7.14.2 Arguments .....	538
2.2.7.15.3.1 Input .....	538
2.2.7.16.4.2 Outputs .....	541
2.2.7.17.5 Related Functions .....	541
2.2.7.18.6 Examples .....	541
2.2.7.19.7.1 Function Block .....	541
2.2.7.20.8.2 Instruction List .....	542
2.2.7.21.9.3 Ladder Diagram .....	542
2.2.7.22.10.4 Structured Text .....	542
2.2.7.23 MC_StopRegist .....	543
2.2.7.24.1 Description .....	543
2.2.7.25.2 Arguments .....	543
2.2.7.26.3.1 Input .....	543
2.2.7.27.4.2 Outputs .....	544
2.2.7.28.5 Related Functions .....	544
2.2.7.29.6 Examples .....	544
2.2.7.30.7.1 Function Block .....	544
2.2.7.31.8.2 Ladder Diagram .....	544
2.2.7.32.9.3 Structured Text .....	544
2.2.8 Superimposed Axes .....	545
2.2.8.1 MC_AddSuperAxis .....	545
2.2.8.2.1 Inputs .....	545
2.2.8.3.2 Outputs .....	545

2.2.8.4.3	Examples	546
2.2.8.5.4.1	Structured Text	546
2.2.8.6.5.2	Function Block Diagram	546
2.2.8.7.6.3	Ladder Diagram	546
2.2.8.8.7	Related Functions	546
2.2.8.9	MC_RemSuperAxis	547
2.2.8.10.1	Inputs	547
2.2.8.11.2	Outputs	547
2.2.8.12.3	Examples	547
2.2.8.13.4.1	Structured Text	547
2.2.8.14.5.2	Function Block Diagram	548
2.2.8.15.6.3	Ladder Diagram	548
2.2.8.16.7	Related Functions	548
<b>2.3</b>	<b>MotionLibrary- Common</b>	<b>549</b>
2.3.1	Motion Library - Common - Info	549
2.3.1.1	MC_ErrorDescription	549
2.3.1.2.1	Arguments	550
2.3.1.3.2.1	Inputs	550
2.3.1.4.3.2	Outputs	550
2.3.1.5.4	Examples	550
2.3.1.6.5.1	Structured Text	550
2.3.1.7.6.2	IL	550
2.3.1.8.7.3	Function Block	550
2.3.1.9.8.4	Ladder Diagram	551
2.3.2	Motion Library - Common - Profiles	552
2.3.2.1	MLProfileBuild	552
2.3.2.2.1	Description	552
2.3.2.3.2.1	CamProps_Ref Structure	552
2.3.2.4.3.2	CamData_Ref Structure	552
2.3.2.5.4	Arguments	553
2.3.2.6.5.1	Input	553
2.3.2.7.6.2	Output	555
2.3.2.8.7	Error Codes	556
2.3.2.9.8	Related Functions	557
2.3.2.10.9.1	See Also	557
2.3.2.11.10	Example of How to Use MLProfileBuild	557
2.3.2.12.11	Example of Building a Parabolic Cam Profile	559
2.3.2.13.12	Code Examples	560
2.3.2.14	MLProfileCreate	561
2.3.2.15.1	Description	561

2.3.2.16.2 Arguments .....	561
2.3.2.17.3.1 Input .....	561
2.3.2.18.4.2 Output .....	561
2.3.2.19.5 Related Functions .....	561
2.3.2.20.6 Example .....	561
2.3.2.21.7.1 Structured Text .....	561
2.3.2.22.8.2 Ladder Diagram .....	562
2.3.2.23.9.3 Function Block Diagram .....	562
2.3.2.24 MLProfileInit .....	563
2.3.2.25.1 Description .....	563
2.3.2.26.2 Arguments .....	563
2.3.2.27.3.1 Input .....	563
2.3.2.28.4.2 Output .....	564
2.3.2.29.5.3 Return Type .....	564
2.3.2.30.6 Related Functions .....	564
2.3.2.31.7 Example .....	565
2.3.2.32.8.1 Structured Text .....	565
2.3.2.33.9.2 Ladder Diagram .....	565
2.3.2.34.10.3 Function Block Diagram .....	565
2.3.2.35 MLProfileRelease .....	566
2.3.2.36.1 Arguments .....	566
2.3.2.37.2.1 Input .....	566
2.3.2.38.3.2 Output .....	567
2.3.2.39.4 Error Codes .....	567
2.3.2.40.5 Related Functions .....	567
2.3.2.41.6 Example .....	568
2.3.2.42.7.1 Structured Text .....	568
2.3.2.43.8.2 Ladder Diagram .....	568
2.3.2.44.9.3 Function Block Diagram .....	568
2.3.3 Motion Library .....	569
2.3.3.1 State Machine .....	569
2.3.3.2 MLMotionCycleTime .....	569
2.3.3.3.1 Arguments .....	569
2.3.3.4.2.1 Input .....	569
2.3.3.5.3.2 Output .....	570
2.3.3.6.4 Example .....	570
2.3.3.7.5.1 Structured Text .....	570
2.3.3.8.6.2 Ladder Diagram .....	570
2.3.3.9.7.3 Function Block Diagram .....	570
2.3.3.10 MLMotionInit .....	571

---

2.3.3.11.1 Description .....	571
2.3.3.12.2 Parameter .....	571
2.3.3.13.3 Return Type .....	571
2.3.3.14.4 Example .....	571
2.3.3.15.5.1 ST .....	571
2.3.3.16.6.2 FBD .....	571
2.3.3.17.7.3 FFLD .....	572
2.3.3.18 MLMotionRstErr .....	573
2.3.3.19.1 Description .....	573
2.3.3.20.2 Return Type .....	573
2.3.3.21.3 Example .....	573
2.3.3.22.4.1 ST .....	573
2.3.3.23.5.2 FBD .....	573
2.3.3.24.6.3 FFLD .....	573
2.3.3.25 MLMotionStart .....	574
2.3.3.26.1 Description .....	574
2.3.3.27.2 Return Type .....	574
2.3.3.28.3 Example .....	574
2.3.3.29.4.1 ST .....	574
2.3.3.30.5.2 FBD .....	574
2.3.3.31.6.3 FFLD .....	574
2.3.3.32 MLMotionStatus .....	576
2.3.3.33.1 Description .....	576
2.3.3.34.2 Parameter .....	576
2.3.3.35.3 Return Type .....	576
2.3.3.36.4 Example .....	576
2.3.3.37.5.1 ST .....	576
2.3.3.38.6.2 FBD .....	576
2.3.3.39.7.3 FFLD .....	577
2.3.3.40 MLMotionStop .....	578
2.3.3.41.1 Description .....	578
2.3.3.42.2 Return Type .....	578
2.3.3.43.3 Example .....	578
2.3.3.44.4.1 ST .....	578
2.3.3.45.5.2 FBD .....	578
2.3.3.46.6.3 FFLD .....	578
2.3.3.47 MLMotionSysTime .....	579
2.3.3.48.1 Description .....	579
2.3.3.49.2 Return Type .....	579



2.3.3.50.3 Units .....	579
2.3.3.51.4 Example .....	579
2.3.3.52.5.1 ST .....	579
2.3.3.53.6.2 FBD .....	579
2.3.3.54.7.3 FFLD .....	579
2.3.4 Coordinated Motion Function Blocks .....	580
2.3.4.1 Coordinated Motion Group Control Library .....	581
2.3.4.2.1 MC_AddAxisToGrp .....	582
2.3.4.3.2.1 Description .....	582
2.3.5 Related Functions .....	582
2.3.5.1.1.1 Arguments .....	582
2.3.6 Input .....	583
2.3.7 Output .....	583
2.3.7.1.1.1 Example .....	584
2.3.8 Structured Text .....	584
2.3.9 IL .....	584
2.3.10 FBD .....	584
2.3.11 Ladder Diagram .....	584
2.3.11.1.1 MC_CreateAxesGrp .....	585
2.3.11.2.2.1 Description .....	585
2.3.12 Related Function Blocks .....	585
2.3.12.1.1.1 Arguments .....	585
2.3.12.2.2.2 Input .....	585
2.3.12.3.3.3 Output .....	586
2.3.12.4.4.4 Example .....	587
2.3.13 Structured Text .....	587
2.3.14 Instruction List .....	587
2.3.15 Function Block Diagram .....	587
2.3.16 Ladder Diagram .....	587
2.3.16.1.1 MC_GrpDisable .....	588
2.3.16.2.2.1 Description .....	588
2.3.17 Related Functions .....	588
2.3.17.1.1.1 Arguments .....	588
2.3.18 Input .....	588
2.3.19 Output .....	588
2.3.19.1.1.1 Example .....	589
2.3.20 ST .....	589
2.3.21 IL .....	589
2.3.22 FBD .....	589
2.3.23 FFLD .....	589
2.3.23.1.1 MC_GrpEnable .....	590

---

2.3.23.2.2.1 Description .....	590
2.3.24 Related Functions .....	590
2.3.24.1.1.1 Arguments .....	590
2.3.25 Input .....	590
2.3.26 Output .....	590
2.3.26.1.1.1 Example .....	591
2.3.27 Structured Text .....	591
2.3.28 IL .....	591
2.3.29 FBD .....	591
2.3.30 FFLD .....	591
2.3.30.1.1 MC_GrpReadBoolPar .....	592
2.3.30.2.2.1 Description .....	592
2.3.31 Related Function Blocks .....	592
2.3.31.1.1.1 Arguments .....	592
2.3.32 Input .....	592
2.3.33 Output .....	593
2.3.33.1.1.1 Example .....	593
2.3.34 ST .....	593
2.3.35 IL .....	593
2.3.36 FBD .....	594
2.3.37 FFLD .....	594
2.3.37.1.1 MC_GrpReadParam .....	595
2.3.37.2.2.1 Arguments .....	595
2.3.38 Input .....	595
2.3.39 Output .....	596
2.3.39.1.1.1 Examples .....	596
2.3.40 ST .....	596
2.3.41 FBD .....	596
2.3.42 FFLD .....	596
2.3.42.1.1 MC_GrpReset .....	597
2.3.42.2.2.1 Description .....	597
2.3.43 Related Functions .....	597
2.3.43.1.1.1 Arguments .....	597
2.3.44 Input .....	597
2.3.45 Output .....	597
2.3.45.1.1.1 Example .....	598
2.3.46 ST .....	598
2.3.47 FBD .....	598
2.3.48 IL .....	598
2.3.49 FFLD .....	598
2.3.49.1.1 MC_GrpStop .....	599
2.3.49.2.2.1 Description .....	599

2.3.50 Related Functions .....	599
2.3.50.1.1.1 Arguments .....	599
2.3.51 Input .....	599
2.3.52 Output .....	600
2.3.52.1.1.1 Example .....	600
2.3.53 Structured Text .....	600
2.3.54 IL .....	600
2.3.55 FBD .....	601
2.3.56 FFLD .....	601
2.3.56.1.1 MC_GrpWriteBoolPar .....	602
2.3.56.2.2.1 Description .....	602
2.3.57 Related Function Blocks .....	602
2.3.57.1.1.1 Arguments .....	602
2.3.58 Input .....	602
2.3.59 Output .....	603
2.3.59.1.1.1 Example .....	603
2.3.60 ST .....	603
2.3.61 IL .....	603
2.3.62 FBD .....	604
2.3.63 FFLD .....	604
2.3.63.1.1 MC_GrpWriteParam .....	605
2.3.63.2.2.1 Arguments .....	605
2.3.64 Input .....	605
2.3.65 Output .....	606
2.3.65.1.1.1 Examples .....	606
2.3.66 ST .....	606
2.3.67 FBD .....	606
2.3.68 FFLD .....	606
2.3.68.1.1 MC_InitAxesGrp .....	608
2.3.68.2.2.1 Description .....	608
2.3.69 Related Function Blocks .....	608
2.3.69.1.1.1 Arguments .....	608
2.3.70 Inputs .....	608
2.3.71 Outputs .....	609
2.3.71.1.1.1 Example .....	610
2.3.72 Structured Text .....	610
2.3.73 IL .....	610
2.3.74 FBD .....	610
2.3.75 FFLD .....	610
2.3.75.1.1 MC_RemAxisFromGrp .....	611
2.3.75.2.2.1 Description .....	611
2.3.76 Related Functions .....	611

---

2.3.76.1.1.1 Arguments .....	611
2.3.77 Input .....	611
2.3.78 Output .....	612
2.3.78.1.1.1 Example .....	612
2.3.79 ST .....	612
2.3.80 IL .....	612
2.3.81 FBD .....	612
2.3.82 FFLD .....	613
2.3.82.1.1 MC_UngroupAllAxes .....	614
2.3.82.2.2.1 Description .....	614
2.3.83 Related Functions .....	614
2.3.83.1.1.1 Arguments .....	614
2.3.84 Input .....	614
2.3.85 Output .....	614
2.3.85.1.1.1 Examples .....	615
2.3.86 ST .....	615
2.3.87 IL .....	615
2.3.88 FBD .....	615
2.3.89 FFLD .....	615
2.3.89.1 Coordinated Motion Info Library .....	616
2.3.89.2.1 MC_GrpReadActAcc .....	616
2.3.89.3.2.1 Description .....	616
2.3.90 Related Functions .....	616
2.3.90.1.1.1 Arguments .....	617
2.3.91 Input .....	617
2.3.92 Output .....	618
2.3.92.1.1.1 Example .....	618
2.3.93 Structured Text .....	618
2.3.94 IL .....	618
2.3.95 FBD .....	618
2.3.96 Ladder Diagram .....	618
2.3.96.1.1 MC_GrpReadActPos .....	620
2.3.96.2.2.1 Description .....	620
2.3.97 Related Functions .....	620
2.3.97.1.1.1 Arguments .....	620
2.3.98 Input .....	620
2.3.99 Output .....	621
2.3.99.1.1.1 Example .....	621
2.3.100 Structured Text .....	621
2.3.101 IL .....	622
2.3.102 FBD .....	622
2.3.103 Ladder Diagram .....	622

2.3.103.1.1 MC_GrpReadActVel .....	623
2.3.103.2.2.1 Description .....	623
2.3.104 Related Functions .....	623
2.3.104.1.1.1 Arguments .....	623
2.3.105 Input .....	623
2.3.106 Output .....	624
2.3.106.1.1.1 Example .....	625
2.3.107 Structured Text .....	625
2.3.108 IL .....	625
2.3.109 FBD .....	625
2.3.110 FFLD .....	625
2.3.110.1.1 MC_GrpReadCmdPos .....	626
2.3.110.2.2.1 Description .....	626
2.3.111 Related Function Blocks .....	626
2.3.111.1.1.1 Arguments .....	626
2.3.112 Input .....	626
2.3.113 Output .....	627
2.3.113.1.1.1 Example .....	627
2.3.114 Structured Text .....	628
2.3.115 IL .....	628
2.3.116 FBD .....	628
2.3.117 FFLD .....	628
2.3.117.1.1 MC_GrpReadCmdVel .....	629
2.3.117.2.2.1 Description .....	629
2.3.118 Related Function Blocks .....	629
2.3.118.1.1.1 Arguments .....	629
2.3.119 Input .....	629
2.3.120 Output .....	630
2.3.120.1.1.1 Example .....	631
2.3.121 Structured Text .....	631
2.3.122 IL .....	631
2.3.123 FBD .....	631
2.3.124 FFLD .....	631
2.3.124.1.1 MC_GrpReadError .....	632
2.3.124.2.2.1 Description .....	632
2.3.125 Related Functions .....	632
2.3.125.1.1.1 Arguments .....	632
2.3.126 Input .....	632
2.3.127 Output .....	632
2.3.127.1.1.1 Examples .....	632
2.3.128 Structured Text .....	632
2.3.129 FBD .....	633

---

2.3.130 FFLD .....	633
2.3.130.1.1 MC_GrpReadStatus .....	634
2.3.130.2.2.1 Description .....	634
2.3.131 Related Functions .....	634
2.3.131.1.1.1 Arguments .....	634
2.3.131.2.2.2 Input .....	634
2.3.131.3.3.3 Output .....	635
2.3.131.4.4.4 Example .....	636
2.3.132 Structured Text .....	636
2.3.133 IL .....	636
2.3.134 FBD .....	637
2.3.135 FFLD .....	637
2.3.135.1 Coordinated Motion Motion Library .....	638
2.3.135.2.1 MC_AxisSetDefaults .....	638
2.3.135.3.2.1 Description .....	638
2.3.136 Related Functions .....	638
2.3.136.1.1.1 Arguments .....	639
2.3.137 Input .....	639
2.3.138 Output .....	640
2.3.138.1.1.1 Example .....	640
2.3.139 Structured Text .....	640
2.3.140 Instruction List .....	641
2.3.141 Function Block Diagram .....	641
2.3.142 Ladder Diagram .....	641
2.3.142.1.1 MC_GrpHalt .....	642
2.3.142.2.2.1 Description .....	642
2.3.143 Related Functions .....	642
2.3.143.1.1.1 Arguments .....	642
2.3.144 Input .....	642
2.3.145 Output .....	643
2.3.145.1.1.1 Example .....	643
2.3.146 Structured Text .....	643
2.3.147 IL .....	644
2.3.148 FBD .....	644
2.3.149 FFLD .....	644
2.3.149.1.1 MC_GrpSetOverride .....	645
2.3.149.2.2.1 Description .....	645
2.3.150 Related Functions .....	645
2.3.150.1.1.1 Arguments .....	645
2.3.151 Input .....	645
2.3.152 Output .....	646
2.3.152.1.1.1 Example .....	646

2.3.153 ST .....	646
2.3.154 IL .....	646
2.3.155 FBD .....	646
2.3.156 FFLD .....	646
2.3.156.1.1 MC_MoveCircAbs .....	647
2.3.156.2.2.1 Description .....	647
2.3.157 Related Functions .....	647
2.3.157.1.1.1 Arguments .....	647
2.3.158 Input .....	647
2.3.159 Output .....	652
2.3.159.1.1.1 Example .....	653
2.3.160 ST .....	653
2.3.161 IL .....	653
2.3.162 FBD .....	653
2.3.163 FFLD .....	653
2.3.163.1.1 MC_MoveCircRel .....	655
2.3.163.2.2.1 Description .....	655
2.3.164 Related Functions .....	655
2.3.164.1.1.1 Arguments .....	655
2.3.165 Input .....	655
2.3.166 Output .....	660
2.3.166.1.1.1 Example .....	660
2.3.167 ST .....	660
2.3.168 IL .....	660
2.3.169 FBD .....	661
2.3.170 FFLD .....	661
2.3.170.1.1 MC_MoveDirAbs .....	662
2.3.170.2.2.1 Description .....	662
2.3.171 Related Functions .....	662
2.3.171.1.1.1 Arguments .....	662
2.3.172 Input .....	662
2.3.173 Output .....	663
2.3.173.1.1.1 Example .....	664
2.3.174 Structure Text .....	664
2.3.175 IL .....	664
2.3.176 Function Block Diagram .....	664
2.3.177 Ladder Diagram .....	664
2.3.177.1.1 MC_MoveDirRel .....	665
2.3.177.2.2.1 Description .....	665
2.3.178 Related Functions .....	665
2.3.178.1.1.1 Arguments .....	665
2.3.179 Input .....	665

2.3.180 Output .....	666
2.3.180.1.1.1 Example .....	667
2.3.181 Structure Text .....	667
2.3.182 IL .....	667
2.3.183 Function Block Diagram .....	667
2.3.184 Ladder Diagram .....	667
2.3.184.1.1 MC_MoveLinAbs .....	668
2.3.184.2.2.1 Description .....	668
2.3.185 Related Functions .....	668
2.3.185.1.1.1 Arguments .....	668
2.3.186 Input .....	668
2.3.187 Output .....	672
2.3.187.1.1.1 Example .....	672
2.3.188 Structured Text .....	672
2.3.189 IL .....	672
2.3.190 FBD .....	673
2.3.191 FFLD .....	673
2.3.191.1.1 MC_MoveLinRel .....	674
2.3.191.2.2.1 Description .....	674
2.3.192 Related Functions .....	674
2.3.192.1.1.1 Arguments .....	674
2.3.193 Input .....	674
2.3.194 Output .....	678
2.3.194.1.1.1 Example .....	678
2.3.195 Structured Text .....	678
2.3.196 IL .....	678
2.3.197 FBD .....	679
2.3.198 FFLD .....	679
2.3.198.1 Coordinated Motion Reference Library .....	680
2.3.198.2.1 MC_GrpSetPos .....	680
2.3.198.3.2.1 Description .....	680
2.3.199 Related Functions .....	680
2.3.199.1.1.1 Arguments .....	680
2.3.200 Input .....	680
2.3.201 Output .....	681
2.3.201.1.1.1 Example .....	682
2.3.202 ST .....	682
2.3.203 FBD .....	682
2.3.204 IL .....	682
2.3.205 FFLD .....	682
<b>3 Fieldbus Library .....</b>	<b>683</b>
<b>3.1 EtherCAT Library .....</b>	<b>684</b>
3.1.1 EtherCAT Library - Drive .....	685



3.1.1.1.1 Execution Time .....	685
3.1.1.2.2.1 Reason .....	685
3.1.1.3.3.2 Result .....	685
3.1.1.4.4.3 Solution .....	685
3.1.1.5 DriveParamRead .....	686
3.1.1.6.1 Description .....	686
3.1.1.7.2 Arguments .....	686
3.1.1.8.3.1 Input .....	686
3.1.1.9.4.2 Output .....	687
3.1.1.10.5 Usage .....	689
3.1.1.11.6 Related Functions .....	689
3.1.1.12.7 Example .....	689
3.1.1.13.8.1 Structured Text .....	689
3.1.1.14.9.2 FBD .....	689
3.1.1.15.10.3 FFLD .....	689
3.1.1.16 DriveParamWrite .....	691
3.1.1.17.1 Description .....	691
3.1.1.18.2 Arguments .....	691
3.1.1.19.3.1 Input .....	691
3.1.1.20.4.2 Output .....	692
3.1.1.21.5 Usage .....	693
3.1.1.22.6 Related Functions .....	693
3.1.1.23.7 Example .....	693
3.1.1.24.8.1 Structured Text .....	693
3.1.1.25.9.2 FBD .....	693
3.1.1.26.10.3 FFLD .....	693
3.1.2 EtherCAT Library - SDO .....	694
3.1.2.1 ECATReadSDO .....	694
3.1.2.2.1 Description .....	694
3.1.2.3.2.1 State Diagram .....	694
3.1.2.4.3 Arguments .....	695
3.1.2.5.4.1 Input .....	695
3.1.2.6.5.2 Output .....	697
3.1.2.7.6 Related Functions .....	697
3.1.2.8.7 Example .....	698
3.1.2.9.8.1 Structured Text .....	698
3.1.2.10.9.2 FBD .....	698
3.1.2.11.10.3 FFLD .....	698
3.1.2.12 ECATWriteSDO .....	699

---

3.1.2.13.1 Description .....	699
3.1.2.14.2.1 State Diagram .....	699
3.1.2.15.3 Arguments .....	700
3.1.2.16.4.1 Input .....	700
3.1.2.17.5.2 Output .....	701
3.1.2.18.6 Related Functions .....	702
3.1.2.19.7 Example .....	702
3.1.2.20.8.1 Structured Text .....	702
3.1.2.21.9.2 FBD .....	702
3.1.2.22.10.3 FFLD .....	702
3.1.3 EtherCAT Library - Debug .....	704
3.1.3.1 ECATGetObjVal .....	704
3.1.3.2 ECATReadData .....	704
3.1.3.3.1 Description .....	704
3.1.3.4.2 Arguments .....	704
3.1.3.5.3.1 Input .....	704
3.1.3.6.4.2 Output .....	705
3.1.3.7.5 Related Functions .....	705
3.1.3.8.6 Example .....	705
3.1.3.9.7.1 Structured Text .....	705
3.1.3.10.8.2 FBD .....	705
3.1.3.11.9.3 FFLD .....	705
3.1.3.12 ECATWriteData .....	706
3.1.3.13.1 Description .....	706
3.1.3.14.2 Arguments .....	706
3.1.3.15.3.1 Input .....	706
3.1.3.16.4.2 Output .....	706
3.1.3.17.5 Related Functions .....	707
3.1.3.18.6 Example .....	707
3.1.3.19.7.1 Structured Text .....	707
3.1.3.20.8.2 FBD .....	707
3.1.3.21.9.3 FFLD .....	707
3.1.4 EtherCAT Library - Status .....	708
3.1.4.1 ECATCommErrors .....	708
3.1.4.2.1 Description .....	708
3.1.4.3.2 Arguments .....	708
3.1.4.4.3.1 Input .....	708
3.1.4.5.4.2 Output .....	709
3.1.4.6.5 Related Functions .....	709
3.1.4.7.6 Example .....	710

3.1.4.8.7.1 Structured Text .....	710
3.1.4.9.8.2 FBD .....	710
3.1.4.10.9.3 FFLD .....	710
3.1.4.11 ECATDeviceStatus .....	711
3.1.4.12.1 Description .....	711
3.1.4.13.2 Arguments .....	711
3.1.4.14.3.1 Input .....	711
3.1.4.15.4.2 Output .....	711
3.1.4.16.5 Related Functions .....	713
3.1.4.17.6 Example .....	713
3.1.4.18.7.1 Structured Text .....	713
3.1.4.19.8.2 FBD .....	713
3.1.4.20.9.3 FFLD .....	713
3.1.4.21 ECATGetStatus .....	715
3.1.4.22.1 Description .....	715
3.1.4.23.2 Arguments .....	715
3.1.4.24.3.1 Input .....	715
3.1.4.25.4.2 Output .....	715
3.1.4.26.5 Related Functions .....	715
3.1.4.27.6 Example .....	715
3.1.4.28.7.1 Structured Text .....	715
3.1.4.29.8.2 FBD .....	715
3.1.4.30.9.3 FFLD .....	716
3.1.4.31 ECATMasterStatus .....	717
3.1.4.32.1 Description .....	717
3.1.4.33.2 Arguments .....	717
3.1.4.34.3.1 Input .....	717
3.1.4.35.4.2 Output .....	717
3.1.4.36.5 Related Functions .....	718
3.1.4.37.6 Examples .....	718
3.1.4.38.7.1 Structured Text .....	718
3.1.4.39.8.2 FBD .....	718
3.1.4.40.9.3 FFLD .....	718
3.1.4.41 ECATSetControl .....	720
3.1.4.42.1 Description .....	720
3.1.4.43.2 Arguments .....	720
3.1.4.44.3.1 Input .....	720
3.1.4.45.4.2 Output .....	720
3.1.4.46.5 Related Functions .....	720

---

3.1.4.47 ECATWCStatus .....	721
3.1.4.48.1 Description .....	721
3.1.4.49.2 Arguments .....	721
3.1.4.50.3.1 Input .....	721
3.1.4.51.4.2 Output .....	721
3.1.4.52.5 Related Functions .....	721
3.1.4.53.6 Example .....	721
3.1.4.54.7.1 Structured Text .....	721
3.1.4.55.8.2 FBD .....	722
3.1.4.56.9.3 FFLD .....	722
<b>3.2 Profibus Library .....</b>	<b>723</b>
<b>4 System Library .....</b>	<b>724</b>
<b>4.1 PrintMessage .....</b>	<b>725</b>
4.1.1 Description .....	725
4.1.1.1 About the Source .....	725
4.1.1.2 About the Level .....	725
4.1.2 Arguments .....	725
4.1.2.1 Input .....	725
4.1.2.2 Output .....	726
4.1.3 Usage .....	726
4.1.4 Example .....	726
4.1.4.1 Structured Text .....	726
4.1.4.2 Function Block Diagram .....	726
4.1.4.3 FFLD .....	727
<b>4.2 GetCtrlErrors .....</b>	<b>728</b>
4.2.1 Arguments .....	728
4.2.1.1 Input .....	728
4.2.1.2 Output .....	728
4.2.2 Examples .....	728
4.2.2.1 FBD .....	728
4.2.2.2 FFLD .....	728
4.2.2.3 ST .....	729
<b>4.3 ClearCtrlErrors .....</b>	<b>729</b>
4.3.1 Arguments .....	729
4.3.2 Input .....	729
4.3.3 Output .....	729
4.3.4 Examples .....	729
4.3.4.1 FBD .....	729
4.3.4.2 FFLD .....	730
4.3.4.3 ST .....	730
<b>4.4 GetCtrlInfo .....</b>	<b>730</b>
4.4.1 Arguments .....	730
4.4.1.1 Input .....	730
4.4.1.2 Output .....	731
4.4.2 Examples .....	731

4.4.2.1	Structured Text .....	731
4.4.2.2	Ladder Diagram .....	731
<b>4.5</b>	<b>GetCtrlPerf .....</b>	<b>733</b>
4.5.1	Description .....	733
4.5.2	Arguments .....	733
4.5.2.1	Input .....	733
4.5.2.2	Output .....	734
4.5.3	Example .....	735
4.5.3.1	FBD .....	735
4.5.3.2	FFLD .....	735
4.5.3.3	Structured Text .....	736
<b>5</b>	<b>Kollmorgen UDFBs .....</b>	<b>737</b>
<b>5.1</b>	<b>How to create an instance .....</b>	<b>740</b>
<b>5.2</b>	<b>Working with Kollmorgen UDFBs .....</b>	<b>741</b>
5.2.1	FB_FirstOrderDigitalFilter .....	742
5.2.1.1	Description .....	742
5.2.1.2	Arguments .....	743
5.2.1.3.1	Inputs .....	743
5.2.1.4.2	Outputs .....	743
5.2.1.5	Usage .....	744
5.2.1.6	Example .....	747
5.2.1.7.1	Structured Text .....	747
5.2.1.8.2	Ladder Diagram .....	747
5.2.1.9.3	Function Block Diagram .....	748
5.2.2	FB_PWDutyOutput .....	749
5.2.2.1	Description .....	749
5.2.2.2	Arguments .....	749
5.2.2.3.1	Input .....	749
5.2.2.4.2	Output .....	750
5.2.2.5	Usage .....	750
5.2.2.6	Related Functions .....	750
5.2.2.7	Example .....	750
5.2.2.8.1	Structured Text .....	750
5.2.2.9.2	Ladder Diagram .....	750
5.2.2.10.3	Function Block Diagram .....	751
5.2.2.11	FB_ScaleInput .....	752
5.2.2.12.1	Description .....	752
5.2.2.13.2	Arguments .....	752
5.2.2.14.3.1	Input .....	752
5.2.2.15.4.2	Output .....	753
5.2.2.16.5	Usage .....	753
5.2.2.17.6	Related Functions .....	753
5.2.2.18.7	Example .....	753

---

5.2.2.19.8.1 Structured Text .....	753
5.2.2.20.9.2 Ladder Diagram .....	753
5.2.2.21.10.3 Function Block Diagram .....	754
5.2.2.22 FB_ScaleOutput .....	755
5.2.2.23.1 Description .....	755
5.2.2.24.2 Arguments .....	755
5.2.2.25.3.1 Input .....	755
5.2.2.26.4.2 Output .....	756
5.2.2.27.5 Usage .....	756
5.2.2.28.6 Related Functions .....	756
5.2.2.29.7 Example .....	756
5.2.2.30.8.1 Structured Text .....	756
5.2.2.31.9.2 Ladder Diagram .....	756
5.2.2.32.10.3 Function Block Diagram .....	756
5.2.3 FB_ElapseTime .....	758
5.2.3.1 Description .....	758
5.2.3.2 Arguments .....	758
5.2.3.3.1 Input .....	758
5.2.3.4.2 Output .....	759
5.2.3.5 Usage .....	759
5.2.3.6 Example .....	759
5.2.3.7.1 Structured text .....	759
5.2.3.8.2 Function Block Diagram .....	760
5.2.3.9.3 Free Form Ladder Diagram .....	760
5.2.4 PipeNetwork_FFLED .....	761
5.2.4.1 Description .....	761
5.2.4.2 Arguments .....	761
5.2.4.3.1 Inputs .....	761
5.2.4.4.2 Outputs .....	762
5.2.4.5 Usage .....	762
5.2.4.6 Example .....	762
5.2.4.7.1 FFLED .....	762
5.2.5 ProfilesCode_FFLED .....	763
5.2.5.1 Description .....	763
5.2.5.2 Arguments .....	763
5.2.5.3.1 Inputs .....	763
5.2.5.4.2 Outputs .....	763
5.2.5.5 Usage .....	764
5.2.5.6 Example .....	764
5.2.5.7.1 FFLED .....	764
5.2.6 FB_TemperaturePID .....	765

---

5.2.6.1 Description .....	765
5.2.6.2 Arguments .....	765
5.2.6.3.1 Inputs .....	765
5.2.6.4.2 Outputs .....	765
5.2.6.5 Usage .....	766
5.2.6.6.1 Tuning Process .....	766
5.2.6.7.2 Start PID Controller .....	767
5.2.6.8 Example .....	767
5.2.6.9.1 ST .....	767
5.2.6.10.2 FBD .....	768
5.2.6.11.3 FFLD .....	768
5.2.6.12 gl MLFB_HomeFindHomeInput .....	769
5.2.6.13.1 Description .....	769
5.2.6.14.2 Arguments .....	769
5.2.6.15.3.1 Input .....	769
5.2.6.16.4.2 Output .....	770
5.2.6.17.5 Example .....	770
5.2.6.18.6.1 ST .....	770
5.2.6.19.7.2 Function Block Diagram .....	771
5.2.6.20.8.3 FFLD .....	771
5.2.6.21 MLFB_HomeFindHomeInputThenZeroAngle .....	772
5.2.6.22.1 Description .....	772
5.2.6.23.2 Arguments .....	772
5.2.6.24.3.1 Input .....	772
5.2.6.25.4.2 Output .....	773
5.2.6.26.5 Example .....	773
5.2.6.27.6.1 ST .....	773
5.2.6.28.7.2 Function Block Diagram .....	773
5.2.6.29.8.3 FFLD .....	774
5.2.6.30 MLFB_HomeFindLimitInput .....	775
5.2.6.31.1 Description .....	775
5.2.6.32.2 Arguments .....	775
5.2.6.33.3.1 Input .....	775
5.2.6.34.4.2 Output .....	775
5.2.6.35.5 Example .....	776
5.2.6.36.6.1 ST .....	776
5.2.6.37.7.2 Function Block Diagram .....	776
5.2.6.38.8.3 FFLD .....	776
5.2.6.39 MLFB_HomeFindLimitInputThenZeroAngle .....	777

---

5.2.6.40.1 Description .....	777
5.2.6.41.2 Arguments .....	777
5.2.6.42.3.1 Input .....	777
5.2.6.43.4.2 Output .....	777
5.2.6.44.5 Example .....	778
5.2.6.45.6.1 ST .....	778
5.2.6.46.7.2 Function Block Diagram .....	778
5.2.6.47.8.3 FFLD .....	778
5.2.6.48 MLFB_HomeFindZeroAngle .....	779
5.2.6.49.1 Description .....	779
5.2.6.50.2 Arguments .....	779
5.2.6.51.3.1 Input .....	779
5.2.6.52.4.2 Output .....	779
5.2.6.53.5 Example .....	780
5.2.6.54.6.1 ST .....	780
5.2.6.55.7.2 Function Block Diagram .....	780
5.2.6.56.8.3 FFLD .....	780
5.2.6.57 MLFB_HomeMoveUntilPosErrExceeded .....	782
5.2.6.58.1 Description .....	782
5.2.6.59.2 Arguments .....	782
5.2.6.60.3.1 Input .....	782
5.2.6.61.4.2 Output .....	782
5.2.6.62.5 Example .....	783
5.2.6.63.6.1 ST .....	783
5.2.6.64.7.2 Function Block Diagram .....	783
5.2.6.65.8.3 FFLD .....	783
5.2.6.66 MLFB_HomeMoveUntilPosErrExceededThenZeroAngle .....	785
5.2.6.67.1 Description .....	785
5.2.6.68.2 Arguments .....	785
5.2.6.69.3.1 Input .....	785
5.2.6.70.4.2 Output .....	785
5.2.6.71.5 Example .....	786
5.2.6.72.6.1 ST .....	786
5.2.6.73.7.2 Function Block Diagram .....	786
5.2.6.74.8.3 FFLD .....	786
5.2.6.75 .....	788
5.2.6.76.1 Description .....	788
5.2.6.77.2 Arguments .....	788
5.2.6.78.3.1 Input .....	788



5.2.6.79.4.2 Output .....	788
5.2.6.80.5 Example .....	788
5.2.6.81.6.1 ST .....	788
5.2.6.82.7.2 Function Block Diagram .....	789
5.2.6.83.8.3 FFLD .....	789
5.2.6.84 MLFB_HomeFindHomeFastInput .....	790
5.2.6.85.1 Description .....	790
5.2.6.86.2 Arguments .....	790
5.2.6.87.3.1 Input .....	790
5.2.6.88.4.2 Output .....	793
5.2.6.89.5 Usage .....	793
5.2.6.90.6 Related Functions .....	794
5.2.6.91.7 Example .....	794
5.2.6.92.8.1 Structured Text .....	794
5.2.6.93.9.2 Ladder Diagram .....	795
5.2.6.94.10.3 Function Block Diagram .....	795
5.2.6.95 MLFB_HomeFindHomeFastInputModulo .....	797
5.2.6.96.1 Description .....	797
5.2.6.97.2 Arguments .....	797
5.2.6.98.3.1 Input .....	797
5.2.6.99.4.2 Output .....	800
5.2.6.100.5 Usage .....	800
5.2.6.101.6 Related Functions .....	801
5.2.6.102.7 Example .....	801
5.2.6.103.8.1 Structured Text .....	801
5.2.6.104.9.2 Ladder Diagram .....	802
5.2.6.105.10.3 Function Block Diagram .....	803
5.2.6.106 .....	804
5.2.6.107.1 Description .....	804
5.2.6.108.2 Arguments .....	804
5.2.6.109.3.1 Input .....	804
5.2.6.110.4.2 Output .....	806
5.2.6.111.5 Usage .....	807
5.2.6.112.6 Related Functions .....	807
5.2.6.113.7 Example .....	807
5.2.6.114.8.1 Structured Text .....	807
5.2.6.115.9.2 Ladder Diagram .....	808
5.2.6.116.10.3 Function Block Diagram .....	808
5.2.6.117 MLFB_HomeFindLimitFastInputModulo .....	810

---

5.2.6.118.1 Description .....	810
5.2.6.119.2 Arguments .....	810
5.2.6.120.3.1 Input .....	810
5.2.6.121.4.2 Output .....	812
5.2.6.122.5 Usage .....	812
5.2.6.123.6 Related Functions .....	812
5.2.6.124.7 Example .....	813
5.2.6.125.8.1 Structured Text .....	813
5.2.6.126.9.2 Ladder Diagram .....	813
5.2.6.127.10.3 Function Block Diagram .....	814
5.2.7 Jog for Pipe Network .....	815
5.2.7.1 Description .....	815
5.2.7.2 Arguments .....	815
5.2.7.3.1 Input .....	815
5.2.7.4.2 Output .....	816
5.2.7.5 Usage .....	816
5.2.7.6 Related Functions .....	816
5.2.7.7 Example .....	816
5.2.7.8.1 Structured Text .....	816
5.2.7.9.2 Ladder Diagram .....	816
5.2.7.10.3 Function Block Diagram .....	817
5.2.7.11 MLFB_PlsPosFw .....	818
5.2.7.12.1 Description .....	818
5.2.7.13.2 Arguments .....	818
5.2.7.14.3.1 Input .....	818
5.2.7.15.4.2 Output .....	818
5.2.7.16.5 Example .....	818
5.2.7.17.6 Timing .....	818
5.2.7.18.7.1 ST .....	818
5.2.7.19.8.2 FBD .....	819
5.2.7.20.9.3 FFLD .....	819
5.2.7.21 MLFB_PlsPosFwBw .....	820
5.2.7.22.1 Description .....	820
5.2.7.23.2 Arguments .....	820
5.2.7.24.3.1 Input .....	820
5.2.7.25.4.2 Output .....	820
5.2.7.26.5 Example .....	820
5.2.7.27.6 Timing .....	820
5.2.7.28.7 Hysteresis .....	820
5.2.7.29.8.1 ST .....	821

5.2.7.30.9.2 FBD .....	821
5.2.7.31.10.3 FFLD .....	821
5.2.7.32 MLFB_PIsTimeFw .....	822
5.2.7.33.1 Description .....	822
5.2.7.34.2 Arguments .....	822
5.2.7.35.3.1 Input .....	822
5.2.7.36.4.2 Output .....	822
5.2.7.37.5 Example .....	822
5.2.7.38.6 Timing .....	822
5.2.7.39.7.1 ST .....	822
5.2.7.40.8.2 FBD .....	823
5.2.7.41.9.3 FFLD .....	823
5.2.7.42 MCFB_StepAbsolutes .....	824
5.2.7.43.1 Description .....	824
5.2.7.44.2 Arguments .....	824
5.2.7.45.3.1 Input .....	824
5.2.7.46.4.2 Output .....	825
5.2.7.47.5 Related Functions .....	826
5.2.7.48.6 Example .....	826
5.2.7.49.7.1 Structured Text .....	826
5.2.7.50.8.2 Ladder Diagram .....	826
5.2.7.51.9.3 Function Block Diagram .....	826
5.2.7.52 MCFB_StepAbsSwitch .....	827
5.2.7.53.1 Description .....	827
5.2.7.54.2 Arguments .....	827
5.2.7.55.3.1 Input .....	827
5.2.7.56.4.2 Output .....	830
5.2.7.57.5 Usage .....	831
5.2.7.58.6 Related Functions .....	832
5.2.7.59.7 Example .....	833
5.2.7.60.8.1 Structured Text .....	833
5.2.7.61.9.2 Ladder Diagram .....	833
5.2.7.62.10.3 Function Block Diagram .....	834
5.2.7.63 MCFB_StepBlock .....	835
5.2.7.64.1 Description .....	835
5.2.7.65.2 Arguments .....	835
5.2.7.66.3.1 Input .....	835
5.2.7.67.4.2 Output .....	837
5.2.7.68.5 Usage .....	838

---

5.2.7.69.6	Related Functions	839
5.2.7.70.7	Example	839
5.2.7.71.8.1	Structured Text	839
5.2.7.72.9.2	Ladder Diagram	839
5.2.7.73.10.3	Function Block Diagram	840
5.2.7.74	MCFB_StepLimitSwitch	841
5.2.7.75.1	Description	841
5.2.7.76.2	Arguments	841
5.2.7.77.3.1	Input	841
5.2.7.78.4.2	Output	844
5.2.7.79.5	Usage	845
5.2.7.80.6	Related Functions	845
5.2.7.81.7	Example	845
5.2.7.82.8.1	Structured Text	845
5.2.7.83.9.2	Ladder Diagram	846
5.2.7.84.10.3	Function Block Diagram	846
5.2.7.85	MCFB_StepRefPulse	848
5.2.7.86.1	Description	848
5.2.7.87.2	Arguments	848
5.2.7.88.3.1	Input	848
5.2.7.89.4.2	Output	851
5.2.7.90.5	Usage	852
5.2.7.91.6	Related Functions	852
5.2.7.92.7	Example	853
5.2.7.93.8.1	Structured Text	853
5.2.7.94.9.2	Ladder Diagram	853
5.2.7.95.10.3	Function Block Diagram	854
5.2.7.96	MCFB_StepAbsSwitchFastInput	855
5.2.7.97.1	Description	855
5.2.7.98.2.1	Input	855
5.2.7.99.3.2	Output	858
5.2.7.100.4	Usage	859
5.2.7.101.5	Related Functions	860
5.2.7.102.6	Example	860
5.2.7.103.7.1	Structured Text	860
5.2.7.104.8.2	FBD	861
5.2.7.105.9.3	Ladder Diagram	861
5.2.7.106	MCFB_StepLimitSwitchFastInput	863
5.2.7.107.1	Description	863

5.2.7.108.2.1 Input .....	863
5.2.7.109.3.2 Output .....	866
5.2.7.110.4 Usage .....	867
5.2.7.111.5 Related Functions .....	867
5.2.7.112.6 Example .....	867
5.2.7.113.7.1 Structured Text .....	867
5.2.7.114.8.2 FBD .....	868
5.2.7.115.9.3 Ladder Diagram .....	868
5.2.8 Jog for PLCopen .....	869
5.2.8.1 Description .....	869
5.2.8.2 Arguments .....	869
5.2.8.3.1 Input .....	869
5.2.8.4.2 Output .....	870
5.2.8.5 Usage .....	870
5.2.8.6 Related Functions .....	870
5.2.8.7 Example .....	870
5.2.8.8.1 Structured Text .....	870
5.2.8.9.2 Ladder Diagram .....	871
5.2.8.10.3 Function Block Diagram .....	871
5.2.9 MCFB_GearedWebTension .....	872
5.2.9.1 Arguments .....	872
5.2.9.2.1 Inputs .....	872
5.2.9.3.2 Output .....	875
5.2.9.4 Usage .....	876
5.2.9.5.1 Example 1 .....	876
5.2.9.6.2 Example 2 .....	876
5.2.9.7.3 PID Function in KAS: .....	877
5.2.9.8.4 Programming tips: .....	877
5.2.9.9.5.1 Example 1 .....	878
5.2.9.10.6.2 Example 2 .....	878
5.2.9.11 Related Functions .....	878
5.2.9.12 Example .....	878
5.2.9.13.1 Ladder Example .....	878
5.2.9.14.2 Function Block Diagram Example .....	879
5.2.9.15.3 Structured Text Example .....	879
5.2.10 FB_FirstOrderDigitalFilter .....	880
5.2.10.1 Description .....	880
5.2.10.2 Arguments .....	880
5.2.10.3.1 Inputs .....	880
5.2.10.4.2 Outputs .....	880
5.2.10.5 Usage .....	881

---

5.2.10.6 Example .....	884
5.2.10.7.1 Structured Text .....	884
5.2.10.8.2 Ladder Diagram .....	884
5.2.10.9.3 Function Block Diagram .....	885
5.2.11 MCFB_AKDFault .....	886
5.2.11.1 Description .....	886
5.2.11.2 Arguments .....	886
5.2.11.3.1 Input .....	886
5.2.11.4.2 Output .....	886
5.2.11.5 Usage .....	887
5.2.11.6 Example .....	887
5.2.11.7.1 Structured Text .....	887
5.2.11.8.2 Ladder Diagram .....	887
5.2.11.9.3 Function Block Diagram .....	887
5.2.12 MCFB_AKDFaultLookup .....	888
5.2.12.1 Description .....	888
5.2.12.2 Arguments .....	888
5.2.12.3.1 Input .....	888
5.2.12.4.2 Output .....	888
5.2.12.5 Usage .....	889
5.2.12.6 Related Functions .....	889
5.2.12.7 Example .....	889
5.2.12.8.1 Structured Text .....	889
5.2.12.9.2 Ladder Diagram .....	889
5.2.12.10.3 Function Block Diagram .....	889
5.2.13 FB_Cylinder .....	890
5.2.13.1 Description .....	890
5.2.13.2 Arguments .....	890
5.2.13.3.1 Input .....	890
5.2.13.4.2 Output .....	890
5.2.13.5 Usage .....	891
5.2.13.6 Example .....	891
5.2.13.7.1 ST .....	891
5.2.13.8.2 Function Block Diagram .....	891
5.2.13.9.3 FFLD .....	891
5.2.13.10 FB_AKDFItRpt .....	893
5.2.13.11.1 Description .....	893
5.2.13.12.2 Arguments .....	893
5.2.13.13.3.1 Input .....	893
5.2.13.14.4.2 Output .....	894
5.2.13.15.5 Usage .....	895

5.2.13.16.6	Related Functions .....	895
5.2.13.17.7	Example .....	895
5.2.13.18.8.1	Structured Text .....	895
5.2.13.19.9.2	Ladder Diagram .....	896
5.2.13.20.10.3	Function Block Diagram .....	896
5.2.13.21	FB_S700FitRpt .....	897
5.2.13.22.1	Description .....	897
5.2.13.23.2	Arguments .....	897
5.2.13.24.3.1	Input .....	897
5.2.13.25.4.2	Output .....	898
5.2.13.26.5	Usage .....	898
5.2.13.27.6	Related Functions .....	898
5.2.13.28.7	Example .....	899
5.2.13.29.8.1	Structured Text .....	899
5.2.13.30.9.2	Ladder Diagram .....	899
5.2.13.31.10.3	Function Block Diagram .....	899
5.2.13.32	FB_AxisPlsPosModulo .....	901
5.2.13.33.1	Description .....	901
5.2.13.34.2	Arguments .....	901
5.2.13.35.3.1	Input .....	901
5.2.13.36.4.2	Output .....	901
5.2.13.37.5	Example .....	901
5.2.13.38.6	Timing .....	901
5.2.13.39.7	Hysteresis .....	902
5.2.13.40.8.1	ST .....	902
5.2.13.41.9.2	Function Block Diagram .....	902
5.2.13.42.10.3	FFLD .....	902
5.2.13.43	FB_AxisPlsPosNoModulo .....	904
5.2.13.44.1	Description .....	904
5.2.13.45.2	Arguments .....	904
5.2.13.46.3.1	Input .....	904
5.2.13.47.4.2	Output .....	904
5.2.13.48.5	Example .....	904
5.2.13.49.6	Timing .....	904
5.2.13.50.7	Hysteresis .....	905
5.2.13.51.8.1	ST .....	905
5.2.13.52.9.2	FBD .....	905
5.2.13.53.10.3	FFLD .....	905
<b>6</b>	<b>Index .....</b>	<b>907</b>

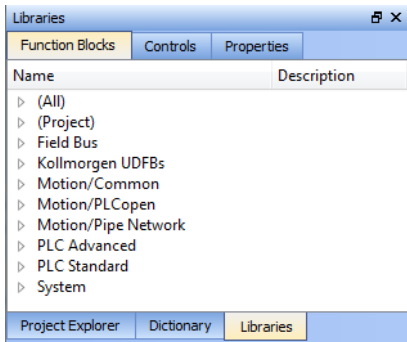
## 2 Motion Library

---

<b>2.1 Motion Library / Pipe Network</b> .....	<b>81</b>
<b>2.2 Motion Library / PLCopen</b> .....	<b>398</b>
<b>2.3 MotionLibrary- Common</b> .....	<b>549</b>



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 SFCSFC programs, ML function blocks (like any other function blocks from the library) are used as part of a stepstep or transitiontransition which are defined with ST, FBD, FFLD or IL languages.

## 2.1 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
<b>Motion</b>	Prepare the physical motion part: init, reset, start, stop
<b>Pipe Network</b>	Manage the Pipe Network: create/activate
<b>Block</b>	Manage the blocks: create/activate
<b>Pipe Block</b>	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.

### 2.1.1 Motion Library - Pipe Network

Name	Description	Return type
<a href="#">MLPipeAct</a>	Activates a pipe	BOOL
<a href="#">MLPipeAddBlock</a>	Adds a Pipe Block to a pipe	BOOL
<a href="#">MLPipeCreate</a>	Creates a new pipe object	None
<a href="#">MLPipeDeact</a>	Deactivates a pipe	BOOL

#### 2.1.1.1 MLPipeAct Pipe Network ✓

##### 2.1.1.2.1 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.

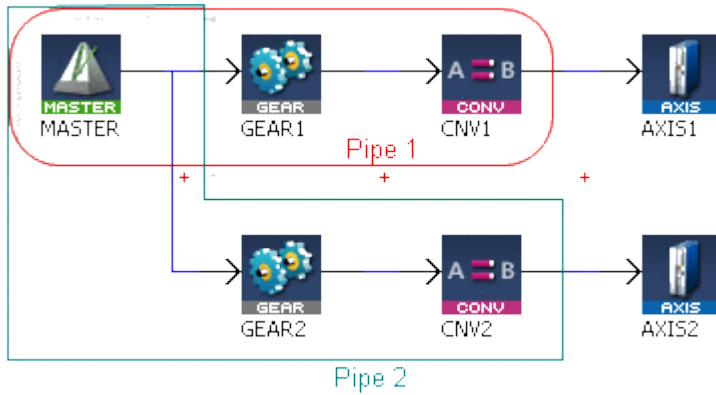


Figure 1-1: MLPipeAct

**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.

2.1.1.3.2 Arguments

2.1.1.4.3.1 Input

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

2.1.1.5.4.2 Output

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

2.1.1.6.5.3 Return Type

BOOL

2.1.1.7.6 Related Functions

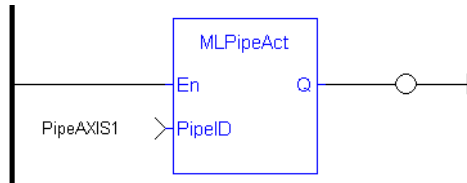
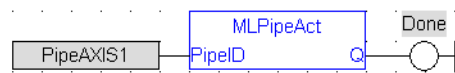
[MLPipeDeact](#)

"MLCNVConnect" (p. 242)

[PipeNetwork\(MLPN\\_ACTIVATE\)](#)

[MLPipeAddBlock](#)**2.1.1.8.7 Example****2.1.1.9.8.1 Structured Text**

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

**2.1.1.10.9.2 Ladder Diagram****2.1.1.11.10.3 Function Block Diagram**

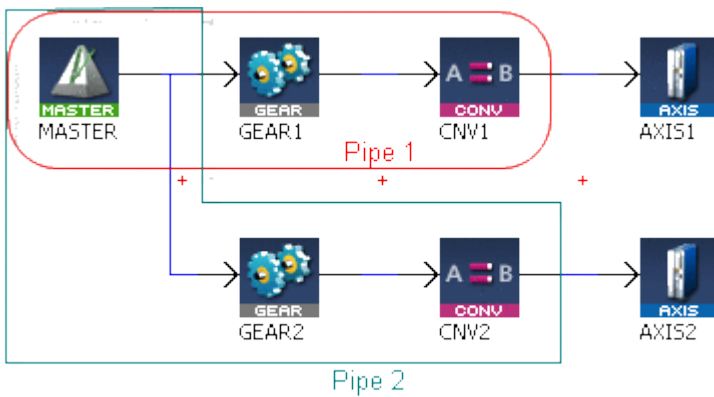
**2.1.1.12 MLPipeAddBlock** Pipe Network ✓

**2.1.1.13.1 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.

**2.1.1.14.2 Arguments**

**2.1.1.15.3.1 Input**

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

**2.1.1.16.4.2 Output**

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

**2.1.1.17.5.3 Return Type**

BOOL

**2.1.1.18.6 Related Functions**

[PipeNetwork\(MLPN\\_CREATE\\_OBJECTS\)](#)

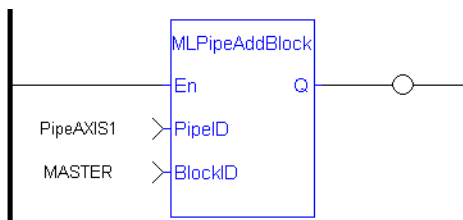
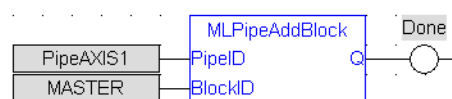
[MLPipeAct](#)

[MLPipeCreate](#)

[MLPipeDeact](#)

**2.1.1.19.7 Example****2.1.1.20.8.1 Structured Text**

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

**2.1.1.21.9.2 Ladder Diagram****2.1.1.22.10.3 Function Block Diagram**

**2.1.1.23 MLPipeCreate** **2.1.1.24.1 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).

**2.1.1.25.2 Arguments****2.1.1.26.3.1 Input**

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

**2.1.1.27.4.2 Output**

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

**2.1.1.28.5 Related Functions**

[PipeNetwork\(MLPN\\_CREATE\\_OBJECTS\)](#)

[MLPipeAddBlock](#)

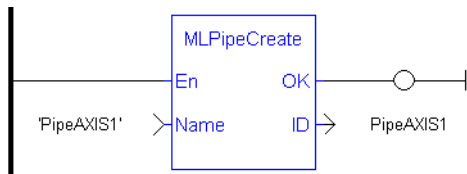
[MLPipeAct](#)

[MLPipeDeact](#)

**2.1.1.29.6 Example****2.1.1.30.7.1 Structured Text**

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

**2.1.1.31.8.2 Ladder Diagram**



### 2.1.1.32.9.3 Function Block Diagram

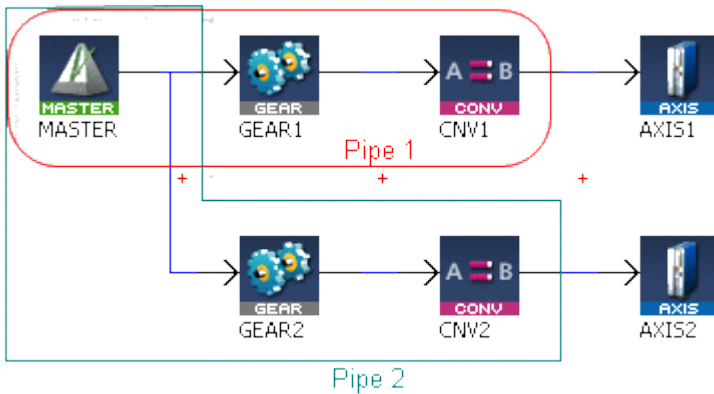


**2.1.1.33 MLPipeDeact** Pipe Network ✓

**2.1.1.34.1 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.

**2.1.1.35.2 Arguments**

**2.1.1.36.3.1 Input**

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

**2.1.1.37.4.2 Output**

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

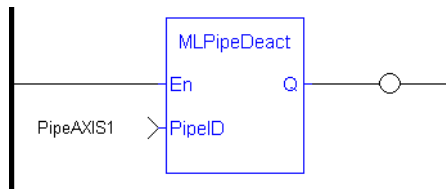
**2.1.1.38.5.3 Return Type**

BOOL



**2.1.1.39.6 Related Functions**[MLPipeAct](#)["MLCNVDisconnect" \(p. 249\)](#)[PipeNetwork\(MLPN\\_DEACTIVATE\)](#)[MLPipeAddBlock](#)**2.1.1.40.7 Example****2.1.1.41.8.1 Structured Text**

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

**2.1.1.42.9.2 Ladder Diagram****2.1.1.43.10.3 Function Block Diagram**

## 2.1.2 Motion Library - Block

Name	Description	Return type
"MLBikCreate" (p. 90)	Creates a new Pipe Block object	None
"MLBikIsReady" (p. 96)	Checks if a Pipe Block currently has a function running	BOOL
"MLBikReadModPos" (p. 94)	Gets the value of the period of a block in user units	None
"MLBikReadOutVal" (p. 92)	Gets the output value of a selected Pipe Block	None
"MLBikWriteModPos" (p. 98)	Sets the value of the period of a block in user units	BOOL

### 2.1.2.1 MLBikCreate

#### 2.1.2.2.1 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" (p. 574) is called.

#### 2.1.2.3.2 Arguments

##### 2.1.2.4.3.1 Input

<b>Name</b>	<b>Description</b>	Desired name for the newly created Pipe Block
	<b>Data type</b>	String
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Type</b>	<b>Description</b>	Type of Pipe Block to create (ex. MASTER, GEAR, PHASER, etc.)
	<b>Data type</b>	String
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

##### 2.1.2.5.4.2 Output

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

**2.1.2.6.5 Related Functions**

[PipeNetwork\(MLPN\\_CREATE\\_OBJECTS\)](#)

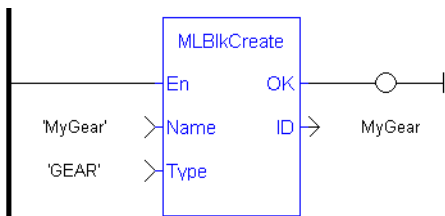
[MLAxisInit](#)

**2.1.2.7.6 Example**

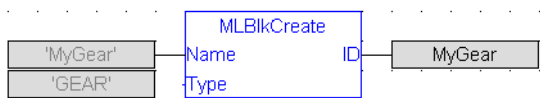
**2.1.2.8.7.1 Structured Text**

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

**2.1.2.9.8.2 Ladder Diagram**



**2.1.2.10.9.3 Function Block Diagram**



**2.1.2.11 MLBlkReadOutVal** Pipe Network ✓

**2.1.2.12.1 Description**

Get the output value a selected Pipe Block.

**2.1.2.13.2 Arguments**

**2.1.2.14.3.1 Input**

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

**2.1.2.15.4.2 Output**

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

**2.1.2.16.5 Related Functions**

[MLBlkReadModPos](#)

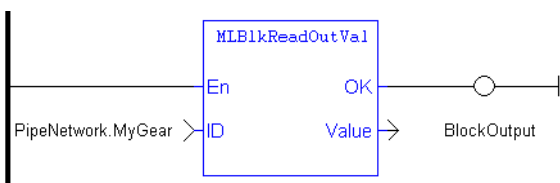
[MLBlkCreate](#)

**2.1.2.17.6 Example**

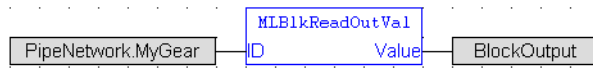
**2.1.2.18.7.1 Structured Text**

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

**2.1.2.19.8.2 Ladder Diagram**



### 2.1.2.20.9.3 Function Block Diagram



### 2.1.2.21 MLBlkReadModPos Pipe Network ✓

#### 2.1.2.22.1 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.

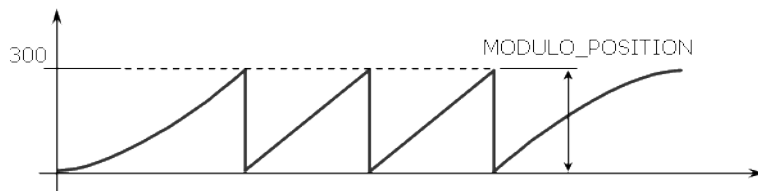


Figure 1-4: MLBlkReadModPos

#### 2.1.2.23.2 Arguments

##### 2.1.2.24.3.1 Input

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

##### 2.1.2.25.4.2 Output

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

#### 2.1.2.26.5 Related Functions

[MLBlkWriteModPos](#)

[MLBlkCreate](#)

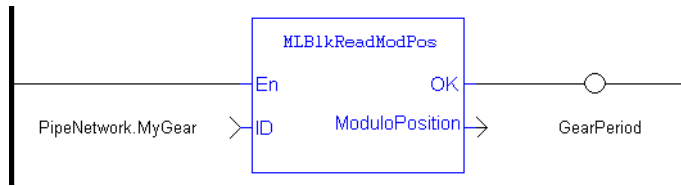
[MLBlkReadOutVal](#)

#### 2.1.2.27.6 Example

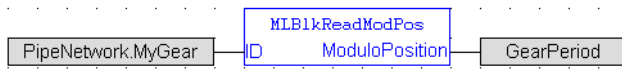
##### 2.1.2.28.7.1 Structured Text

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

### 2.1.2.29.8.2 Ladder Diagram



### 2.1.2.30.9.3 Function Block Diagram



**2.1.2.31 MLBlkIsReady** Pipe Network ✓

**2.1.2.32.1 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

**2.1.2.33.2 Arguments**

**2.1.2.34.3.1 Input**

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

**2.1.2.35.4.2 Output**

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

**2.1.2.36.5.3 Return Type**

BOOL

**2.1.2.37.6 Related Functions**

[MLBlkReadOutVal](#)

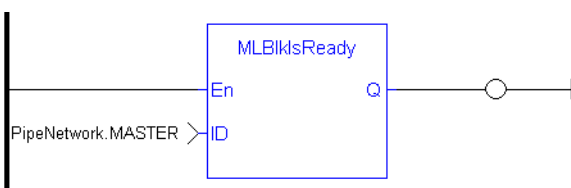
[MLBlkReadModPos](#)

**2.1.2.38.7 Example**

**2.1.2.39.8.1 Structured Text**

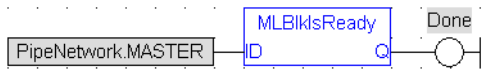
```
//Check if the MST Pipe Block named "MASTER" has a function
running
IsReady := MLBlkIsReady( PipeNetwork.MASTER );
```

**2.1.2.40.9.2 Ladder Diagram**





### 2.1.2.41.10.3 Function Block Diagram



**2.1.2.42 MLBlkWriteModPos** Pipe Network ✓

**2.1.2.43.1 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.

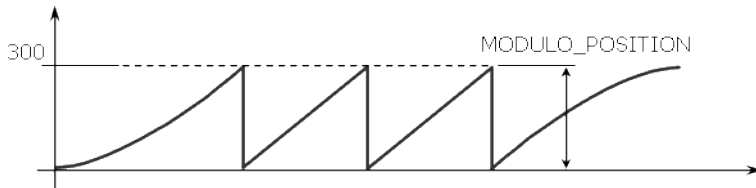


Figure 1-5: MLBlkReadModPos

**2.1.2.44.2 Arguments**

**2.1.2.45.3.1 Input**

<b>ID</b>	<b>Description</b>	ID number of a created Pipe Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ModuloPosition</b>	<b>Description</b>	Desired new Period Value for selected Pipe Block
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—

**2.1.2.46.4.2 Output**

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

**2.1.2.47.5.3 Return Type**

BOOL

**2.1.2.48.6 Related Functions**

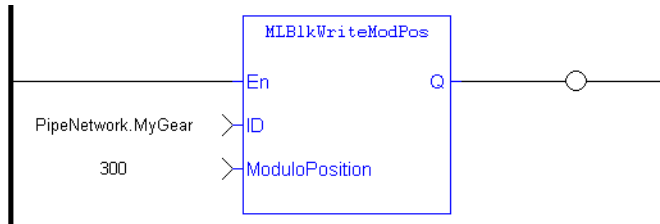
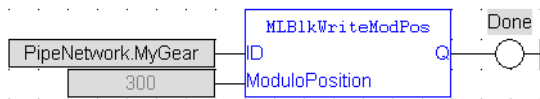
[MLBlkReadModPos](#)

[MLBlkCreate](#)

[MLBlkReadOutVal](#)

**2.1.2.49.7 Example****2.1.2.50.8.1 Structured Text**

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

**2.1.2.51.9.2 Ladder Diagram****2.1.2.52.10.3 Function Block Diagram**

### 2.1.3 Motion Library - Adder

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

#### 2.1.3.1 MLAddInit

##### 2.1.3.2.1 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.

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

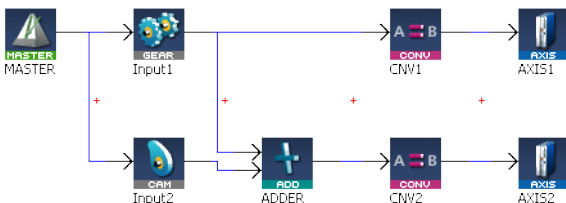


Figure 1-6: MLAddInit

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

##### 2.1.3.3.2 Arguments

##### 2.1.3.4.3.1 Input

<b>BlockID</b>	<b>Description</b>	ID number of a created Pipe Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]

	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Ratio1</b>	<b>Description</b>	Sets the Ratio value of the first entry of an Adder object
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Offset1</b>	<b>Description</b>	Sets the Offset value of the first entry of an Adder object
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Ratio2</b>	<b>Description</b>	Sets the Ratio value of the second entry of an Adder object
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Offset2</b>	<b>Description</b>	Sets the Offset value of the second entry of an Adder object
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 2.1.3.5.4.2 Output

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

#### 2.1.3.6.5.3 Return Type

BOOL

#### 2.1.3.7.6 Related Functions

[MLBlkCreate](#)

[MLAddWriteInput](#)

[MLAddReadOff1](#)

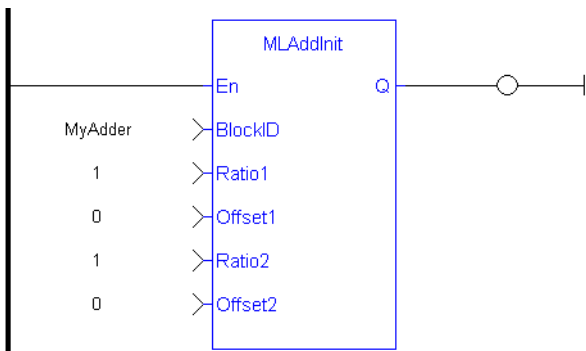
[MLAddReadRatio1](#)

**2.1.3.8.7 Example**

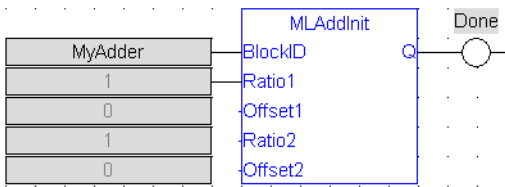
**2.1.3.9.8.1 Structured Text**

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

**2.1.3.10.9.2 Ladder Diagram**



**2.1.3.11.10.3 Function Block Diagram**



### 2.1.3.12 MLAddReadOff1 Pipe Network ✓

#### 2.1.3.13.1 Description

Returns the offset value of the first entry of an Adder block. Can change the offset value with MLAddWriteOff1 function block. Offset1 shifts the value of the first input to the block before its added to the second input.

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

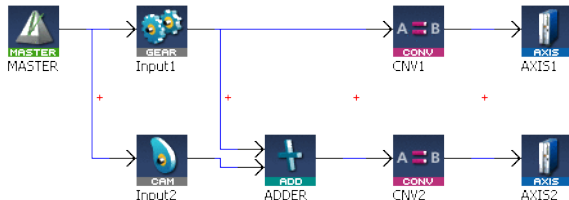


Figure 1-7: MLAddReadOff1

#### 2.1.3.14.2 Arguments

##### 2.1.3.15.3.1 Input

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

##### 2.1.3.16.4.2 Output

<b>Offset</b>	<b>Description</b>	Returns the offset value of the first entry of an Adder object
	<b>Data type</b>	LREAL
	<b>Unit</b>	n/a

#### 2.1.3.17.5 Related Functions

[MLAddWriteOff1](#)

[MLAddReadOff2](#)

[MLAddReadRatio1](#)

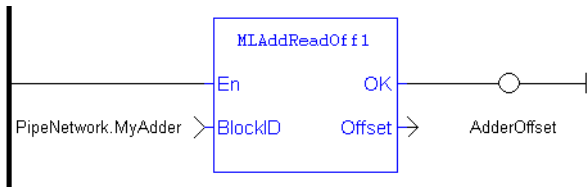
[MLAddWriteRat1](#)

#### 2.1.3.18.6 Example

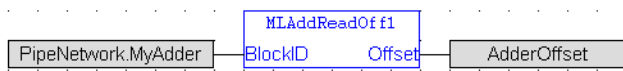
##### 2.1.3.19.7.1 Structured Text

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

### 2.1.3.20.8.2 Ladder Diagram



### 2.1.3.21.9.3 Function Block Diagram





### 2.1.3.22 MLAddReadOff2 Pipe Network ✓

#### 2.1.3.23.1 Description

Returns the offset value of the second entry of an Adder block. Can change the offset value with MLAddWriteOff2 function block. Offset2 shifts the value of the second input to the block before its added to the first input.

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

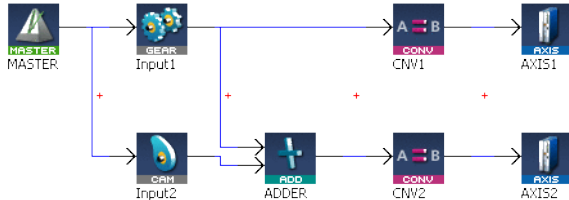


Figure 1-8: MLAddReadOff2

#### 2.1.3.24.2 Arguments

##### 2.1.3.25.3.1 Input

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

##### 2.1.3.26.4.2 Output

<b>Offset</b>	<b>Description</b>	Returns the offset value of the second entry of an Adder object
	<b>Data type</b>	LREAL
	<b>Unit</b>	n/a

#### 2.1.3.27.5 Related Functions

[MLAddWriteOff2](#)

[MLAddReadOff1](#)

[MLAddReadRatio2](#)

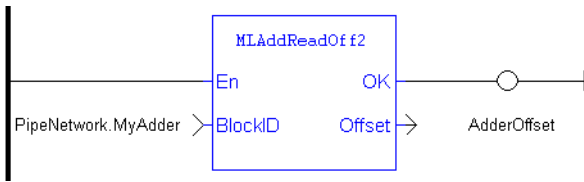
[MLAddWriteRat2](#)

#### 2.1.3.28.6 Example

##### 2.1.3.29.7.1 Structured Text

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

### 2.1.3.30.8.2 Ladder Diagram



### 2.1.3.31.9.3 Function Block Diagram



### 2.1.3.32 MLAddReadRatio1 Pipe Network ✓

#### 2.1.3.33.1 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.

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

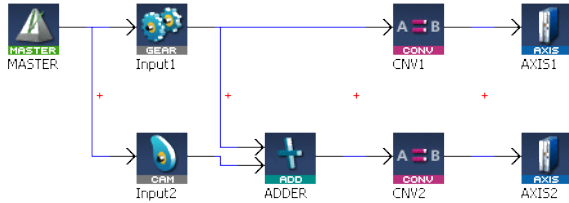


Figure 1-9: MLAddReadRatio1

#### 2.1.3.34.2 Arguments

##### 2.1.3.35.3.1 Input

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

##### 2.1.3.36.4.2 Output

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

#### 2.1.3.37.5 Related Functions

[MLAddWriteRat1](#)

[MLAddReadRatio2](#)

[MLAddReadOff1](#)

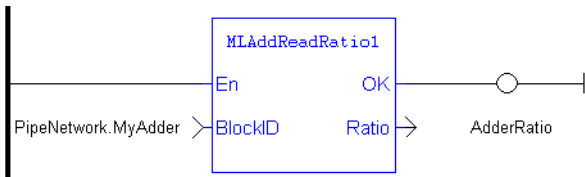
[MLAddReadOff2](#)

#### 2.1.3.38.6 Example

##### 2.1.3.39.7.1 Structured Text

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

### 2.1.3.40.8.2 Ladder Diagram



### 2.1.3.41.9.3 Function Block Diagram



### 2.1.3.42 MLAddReadRatio2 Pipe Network ✓

#### 2.1.3.43.1 Description

Returns the ratio value of the second entry of an Adder block. Can change the ratio value with MLAddWriteRat2 function block. Ratio2 amplifies the value of the second input to the block before its added to the first input.

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

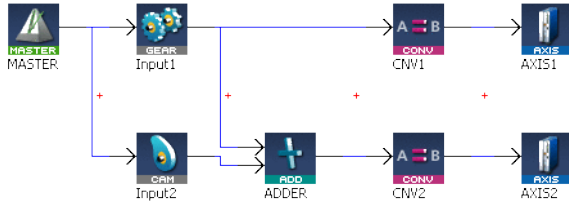


Figure 1-10: MLAddReadRatio2

#### 2.1.3.44.2 Arguments

##### 2.1.3.45.3.1 Input

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

##### 2.1.3.46.4.2 Output

<b>Ratio</b>	<b>Description</b>	Returns the Ratio value of the second entry of an Adder object
	<b>Data type</b>	LREAL
	<b>Unit</b>	n/a

#### 2.1.3.47.5 Related Functions

[MLAddWriteRat2](#)

[MLAddReadRatio1](#)

[MLAddReadOff1](#)

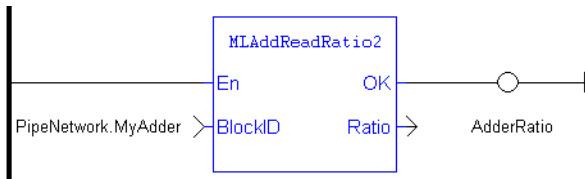
[MLAddReadOff2](#)

#### 2.1.3.48.6 Example

##### 2.1.3.49.7.1 Structured Text

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

### 2.1.3.51.8.2 Ladder Diagram



### 2.1.3.51.9.3 Function Block Diagram



**2.1.3.52 MlAddWriteInput** Pipe Network ✓

**2.1.3.53.1 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.

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

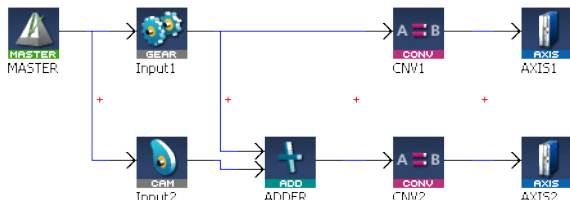


Figure 1-11: MlAddWriteInput

**NOTE**

Adder objects are normally created in the Pipe Network using the graphical engine. Then you do not have to add MlAddWriteInput 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.

**2.1.3.54.2 Arguments**

**2.1.3.55.3.1 Input**

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

### 2.1.3.56.4.2 Output

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

### 2.1.3.57.5.3 Return Type

BOOL

### 2.1.3.58.6 Related Functions

[MLBlkCreate](#)

[MLAddInit](#)

[MLAddReadOff1](#)

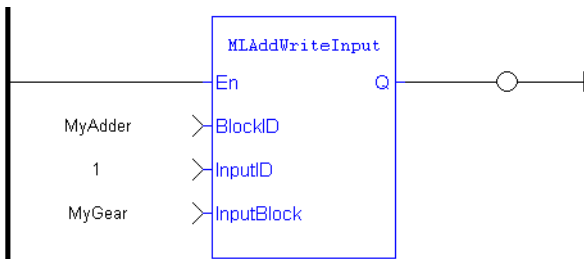
[MLAddReadRatio1](#)

### 2.1.3.59.7 Example

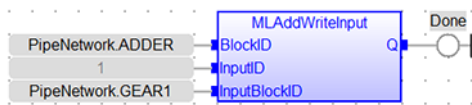
#### 2.1.3.60.8.1 Structured Text

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

### 2.1.3.61.9.2 Ladder Diagram



### 2.1.3.62.10.3 Function Block Diagram





### 2.1.3.63 MAddWriteOff1 Pipe Network ✓

#### 2.1.3.64.1 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.

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

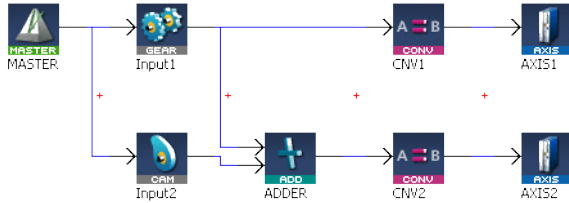


Figure 1-12: MAddWriteOff1

#### ! IMPORTANT

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

#### 2.1.3.65.2 Arguments

##### 2.1.3.66.3.1 Input

<b>BlockID</b>	<b>Description</b>	ID number of an initiated Adder object
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Offset</b>	<b>Description</b>	Desired new value for the Adder Object's Offset1
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

##### 2.1.3.67.4.2 Output

<b>Default (.Q)</b>	<b>Description</b>	Returns TRUE if the Offset value for input one is set
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

##### 2.1.3.68.5.3 Return Type

BOOL

##### 2.1.3.69.6 Related Functions

[MAddReadOff1](#)

[MLAddWriteOff2](#)

[MLAddReadRatio1](#)

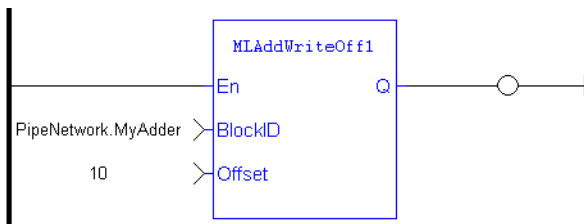
[MLAddWriteRat1](#)

### 2.1.3.70.7 Example

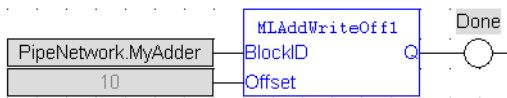
#### 2.1.3.71.8.1 Structured Text

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

#### 2.1.3.72.9.2 Ladder Diagram



#### 2.1.3.73.10.3 Function Block Diagram



### 2.1.3.74 MAddWriteOff2 Pipe Network ✓

#### 2.1.3.75.1 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.

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

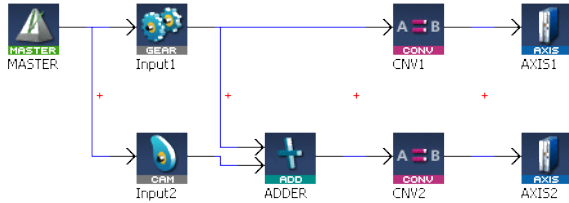


Figure 1-13: MAddWriteOff2

#### ! IMPORTANT

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

#### 2.1.3.76.2 Arguments

##### 2.1.3.77.3.1 Input

<b>BlockID</b>	<b>Description</b>	ID number of an initiated Adder object
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Offset</b>	<b>Description</b>	Desired new value for the Adder Object's Offset2
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

##### 2.1.3.78.4.2 Output

<b>Default (.Q)</b>	<b>Description</b>	Returns TRUE if the Offset value for input two is set
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

#### 2.1.3.79.5.3 Return Type

BOOL

#### 2.1.3.80.6 Related Functions

[MAddReadOff2](#)

[MLAddWriteOff1](#)

[MLAddReadRatio2](#)

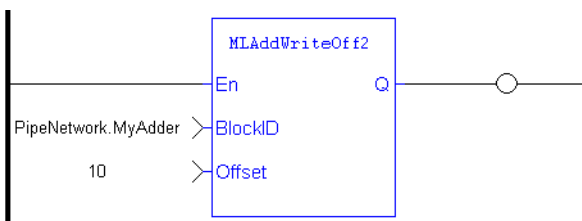
[MLAddWriteRat2](#)

### 2.1.3.81.7 Example

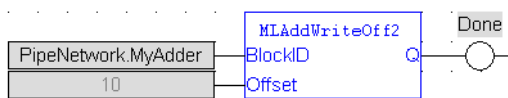
#### 2.1.3.82.8.1 Structured Text

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

#### 2.1.3.83.9.2 Ladder Diagram



#### 2.1.3.84.10.3 Function Block Diagram



**2.1.3.85 MLAddWriteRat1** Pipe Network ✓

**2.1.3.86.1 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.

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

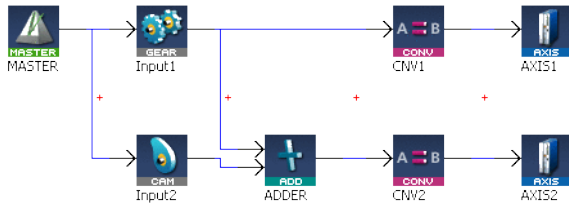


Figure 1-14: MLAddWriteRat1

**ⓘ IMPORTANT**

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

**2.1.3.87.2 Arguments**

**2.1.3.88.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID number of an initiated Adder object
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Ratio</b>	<b>Description</b>	Desired new value for the Adder Object's Ratio1
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.3.89.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns TRUE if the Ratio value for input one is set
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.3.90.5.3 Return Type**

BOOL

**2.1.3.91.6 Related Functions**

[MLAddReadRatio1](#)

[MLAddWriteRat2](#)

[MLAddReadOff1](#)

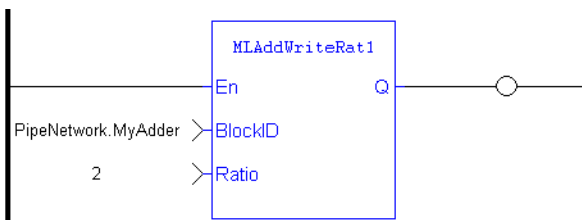
[MLAddWriteOff1](#)

### 2.1.3.92.7 Example

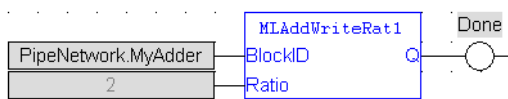
#### 2.1.3.93.8.1 Structured Text

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

#### 2.1.3.94.9.2 Ladder Diagram



#### 2.1.3.95.10.3 Function Block Diagram



**2.1.3.96 MlAddWriteRat2** Pipe Network ✓

**2.1.3.97.1 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.

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

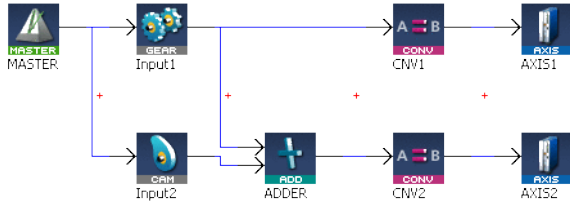


Figure 1-15: MlAddWriteRat2\

**! IMPORTANT**

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

**2.1.3.98.2 Arguments**

**2.1.3.99.3.1 Input**

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

**2.1.3.100.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns TRUE if the Ratio value for input two is set
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.3.101.5.3 Return Type**

BOOL

**2.1.3.102.6 Related Functions**

[MlAddReadRatio2](#)

[MLAddWriteRat1](#)

[MLAddReadOff2](#)

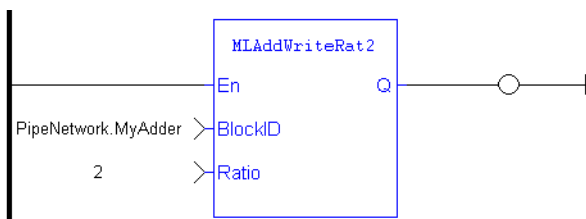
[MLAddWriteOff2](#)

### 2.1.3.103.7 Example

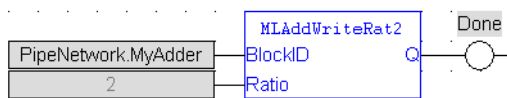
#### 2.1.3.104.8.1 Structured Text

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

#### 2.1.3.105.9.2 Ladder Diagram



#### 2.1.3.106.10.3 Function Block Diagram





## 2.1.4 Motion Library - Axis

### TIP

- For an Axis function example, see "Usage Example of Axis Functions" on page 208

#### Function sorted by types:

Power Stage	Motion Control	Inquiry Functions	Position setting
<a href="#">MLAxisPower</a>	<a href="#">MLAxisAbs</a>	<a href="#">MLAxisGenPos</a>	
<a href="#">MLAxisPowerDOff</a>	<a href="#">MLAxisAdd</a>	<a href="#">MLAxisPipePos</a>	<a href="#">MLAxisReAlign</a>
	<a href="#">MLAxisMoveVel</a>	<a href="#">MLAxisCmdPos</a>	
	<a href="#">MLAxisRel</a>	<a href="#">MLAxisReadActPos</a>	
	<a href="#">MLAxisStop</a>	<a href="#">MLAxisFBackPos</a>	
		<a href="#">MLAxisStatus</a>	
		<a href="#">MLAxisReadGenStatus</a>	
		<a href="#">MLAxisGenIsRdy</a>	
		<a href="#">MLAxisTimeStamp</a>	

#### Functions sorted in alphabetical order:

Name	Description	Return type
<a href="#">MLAxisAbs</a>	Performs a move to an absolute position	BOOL
<a href="#">MLAxisAdd</a>	Performs an additive move relative for a specified distance from the endpoint of the previous move	BOOL
<a href="#">MLAxisAddress</a>	Returns the motion bus address of the axis	DINT
<a href="#">MLAxisAddTq</a>	Sets additive torque	BOOL
<a href="#">MLAxisCfgFastIn</a>	Initializes the Fast Input capability for the axis	BOOL
<a href="#">MLAxisCmdPos</a>	Returns the reference position of the axis	None
<a href="#">MLAxisCreate</a>	Creates a new axis object	None
<a href="#">MLAxisFBackPos</a>	Returns the feedback position of the axis	None
<a href="#">MLAxisGenEN</a>	Enables or disables the internal TMP generator of the axis	BOOL
<a href="#">MLAxisGenIsEN</a>	Checks if the internal TMP generator of the axis is enabled	BOOL
<a href="#">MLAxisGenIsRdy</a>	Checks if an axis is ready	BOOL
<a href="#">MLAxisGenPos</a>	Returns the generator position of the axis	None
<a href="#">MLAxisGenReadAcc</a>	Gets the acceleration of the internal generator of an axis	None
<a href="#">MLAxisGenReadDec</a>	Gets the deceleration of the internal generator of an axis	None
<a href="#">MLAxisGenReadSpd</a>	Gets the speed of the internal generator of an axis	None
<a href="#">MLAxisGenWriteAcc</a>	Sets the acceleration of the internal generator of an axis	BOOL
<a href="#">MLAxisGenWriteDec</a>	Sets the deceleration of the internal generator of an axis	BOOL
<a href="#">MLAxisGenWriteSpd</a>	Sets the speed of the internal generator of an axis	BOOL
<a href="#">MLAxisInit</a>	Initializes an axis object	BOOL
<a href="#">MLAxisIsCnctd</a>	Checks if a pipe is currently connected to the axis	BOOL
<a href="#">MLAxisIsTriggered</a>	Checks if the axis got a trigger event	BOOL
<a href="#">MLAxisMoveVel</a>	Jogs at the specified speed	BOOL
<a href="#">MLAxisPipePos</a>	Returns the pipe position of the axis	None
<a href="#">MLAxisPower</a>	Powers up the axis. Enables Axis Servo Drive.	BOOL

Name	Description	Return type
<a href="#">MLAxisPowerDOff</a>	Returns the adjustment of position done by the last power on to avoid bumps	None
<a href="#">MLAxisRatedTq</a>	Sets rated motor torque	BOOL
<a href="#">MLAxisRead2ndFB</a>	Read secondary feedback	None
<a href="#">MLAxisReadActPos</a>	Returns the actual position of the axis	None
<a href="#">MLAxisReadFBUnit</a>	Gets the feedback units per revolution value of the axis	None
<a href="#">MLAxisReadFEUU</a>	Read following error in user units	None
<a href="#">MLAxisReadGenStatus</a>	Returns the status of the internal generator of the axis	DINT
<a href="#">MLAxisReadModPos</a>	Get the value period of the axis	None
<a href="#">MLAxisReadTq</a>	Read actual torque	None
<a href="#">MLAxisReadJUnits</a>	Get the user units per revolution value of the axis	None
<a href="#">MLAxisReadVel</a>	Read actual velocity	None
<a href="#">MLAxisReAlignRdy</a>	Checks if an axis is ready. Returns TRUE if the internal realignment axis is ready.	BOOL
<a href="#">MLAxisReAlign</a>	Realigns the actual position with the reference position by moving the axis by the specified delta position	BOOL
<a href="#">MLAxisRel</a>	Performs a relative move for a specified distance from the current position	BOOL
<a href="#">MLAxisResetErrors</a>	Clears errors of the specified axis	BOOL
<a href="#">MLAxisRstFastIn</a>	Resets the Fast Input	BOOL
<a href="#">MLAxisStatus</a>	Returns the status of the axis	DINT
<a href="#">MLAxisStop</a>	Stop with the specified deceleration	None
<a href="#">MLAxisTimeStamp</a>	Returns the timestamp of the triggered axis	DINT
<a href="#">MLAxisWriteModPos</a>	Sets the value period of the axis	BOOL
<a href="#">MLAxisWritePipPos</a>	Forces the pipe position internal value. This function is working only when no pipe is connected.	BOOL
<a href="#">MLAxisWritePos</a>	Sets the logical zero position of an axis	BOOL
<a href="#">MLAxisWriteJUnits</a>	Sets the user units per revolution value of the axis	BOOL

#### 2.1.4.1 MLAxisAbs

##### 2.1.4.2.1 Description

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

##### 2.1.4.3.2 Arguments

###### 2.1.4.4.3.1 Input

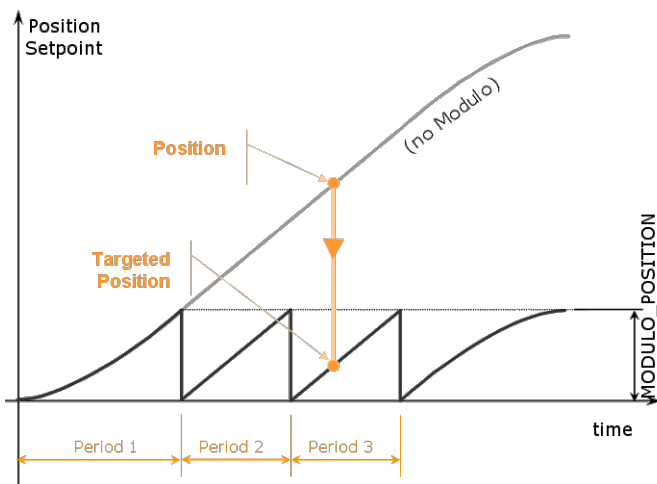
<b>ID</b>	<b>Description</b>	ID name of the Axis Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>Position</b>	<b>Description</b>	Sets the value of the absolute destination position. When the Modulo is turned on, see more explanations below.
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a>
	<b>Default</b>	—

#### 2.1.4.5 Position with Modulo On

When the Modulo is turned on, the Axis Block moves to the targeted position during the corresponding period, calculated as follows:

- If the Position input is between 0 and the Modulo Position, then the Axis Block moves within the **current** period (no position rollover).
- If the Position input is greater than the Modulo Position, then the Axis Block moves during one of the **next** period (positive position rollover).

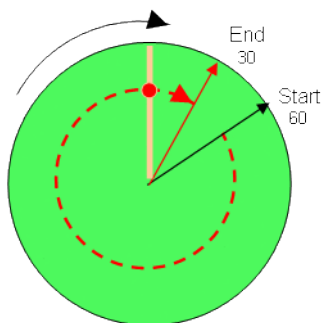


The Axis Block works similarly for negative positions: if the Position input is less than zero, then the Axis Block moves during one of the **previous** period (negative position rollover).

#### 2.1.4.6 Forcing the direction of rotation

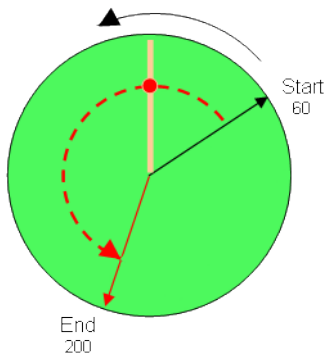
In some applications, the direction of rotation for the axis is forced in one direction only. As a consequence, the motor movement goes to the next or previous modulo in the following situations:

- If the **End Position** is less than the **Start Position** and the direction of rotation for the axis is forced to be clockwise (the **red point** shows when the modulo position is reached)



(see an example in row#2 of the table below)

- If the **End Position** is greater than the **Start Position** and the direction of rotation for the axis is forced to be counter clockwise



(see an example in row#4 of the table below)

### Examples

Start Position	End Position	Direction of rotation	Cross Modulo	Position Input to MLAxisAbs (1)	RelativeDistance Moved (2)	
60	200	clockwise	No	200	140	(i.e. 200 - 60 + 0)
60	30	clockwise	<b>Yes</b>	390	330	(i.e. 30 - 60 + 360)
60	30	counter clockwise	No	30	-30	(i.e. 30 - 60 - 0)
60	200	counter clockwise	<b>Yes</b>	-160	-220	(i.e. 200 - 60 - 360)

With:

(1) **Position Input** = End Position ( + Modulo \* *Direction of rotation*)

(2) **Relative Distance Moved** = End Position - Start Position ( + Modulo \* *Direction of rotation*)

Where:

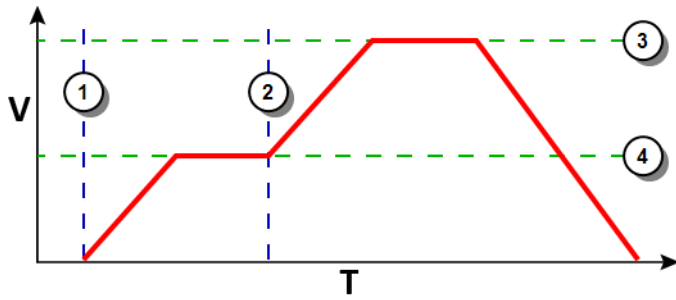
**Direction of rotation** = 1 when clockwise and -1 when anti-clockwise

#### 2.1.4.7 Travel Speed Update with MLAxisAbs

The travel speed of the generator can be updated using the function block "[MLAxisGenWriteSpd](#)" (p. 156). Depending on the state of the generator, this speed is directly reflected on the current move or a future move.

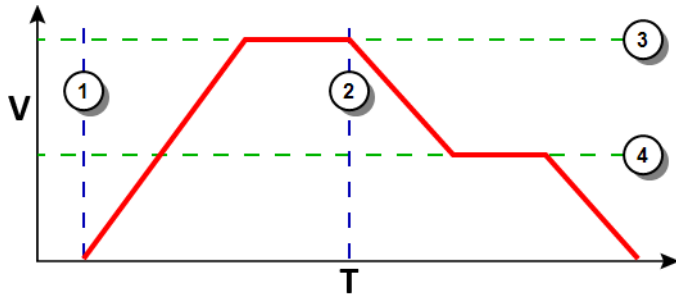
- If MLAxisAbs is not currently being executed, the new travel speed will be applied for the trajectory calculation for a future MLAxisAbs command.
- If MLAxisAbs is currently being executed and a new MLAxisAbs with the same target position is called, the new travel speed will be taken into account only if the current state of the TMP profile is the constant velocity or acceleration. If the axis was decelerating to stop at the goal position the new travel speed will not be taken into account.
- If a MLAxisAbs is currently being executed and a new MLAxisAbs with a different target position is called, the new travel speed is taken into account.

Following are several examples.



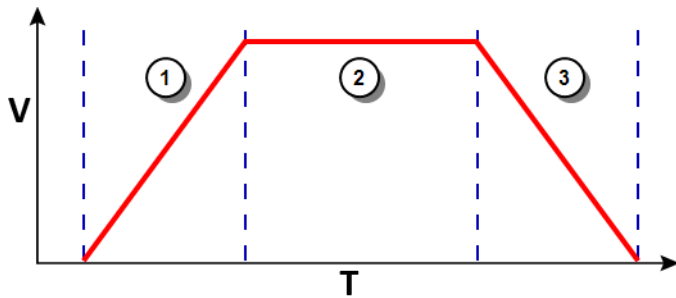
1. First MAxisAbs
2. Second MAxisAbs
3. Initial Speed
4. New Speed

Figure 1-16: Initial speed is smaller than the new speed



1. First MAxisAbs
2. Second MAxisAbs
3. Initial Speed
4. New Speed

Figure 1-17: Initial speed is bigger than the new speed



1. Acceleration
2. Constant Velocity
3. Deceleration

Figure 1-18: The speed update is taken into account only if the second MAxisAbs is triggered during acceleration or constant velocity

#### 2.1.4.8.1.1 Output

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

#### 2.1.4.9.2 Related Functions

[MLAxisGenWriteSpd](#)

[MLAxisGenWriteDec](#)

[MLAxisGenWriteAcc](#)

#### 2.1.4.10.3 Example

See "[Usage Example of Axis Functions](#)" (p. 208) for additional examples.

#### 2.1.4.11.4.1 Structured Text

```
MLAxisAbs ( PipeNetwork.Axis1, 2000 ) ;
```

### 2.1.4.12.5.2 Ladder Diagram



### 2.1.4.13.6.3 Function Block Diagram



**2.1.4.14 MLAxisAdd** **2.1.4.15.1 Description**

A selected Axis performs a move for a specified distance relative to the endpoint of the previous move. The DeltaPosition input is signed so that the move can be in the positive or negative direction, and the Axis moves this distance in user units. The travel speed, acceleration, deceleration, and User Units of the move are values inherited from the selected Axis. The default settings are entered when an Axis is created and initiated, and can be changed with other MLAxis commands such as [MLAxisGenWriteSpd](#), [MLAxisGenWriteAcc](#), and [MLAxisWriteUUnits](#).

**2.1.4.16.2 Arguments****2.1.4.17.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>DeltaPosition</b>	<b>Description</b>	Sets the Axis Delta Position to add to the endpoint of the previous move
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a>
	<b>Default</b>	—

**2.1.4.18.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns true when function successfully executes, after the motion profile is complete
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.4.19.5 Related Functions**

[MLAxisGenWriteAcc](#)

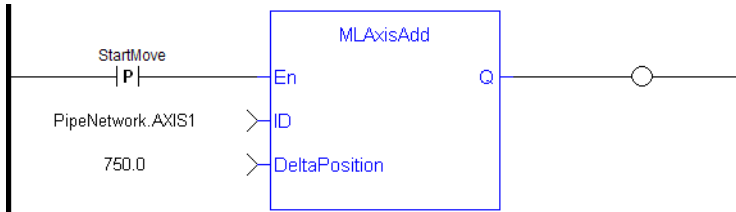
[MLAxisGenWriteDec](#)

[MLAxisGenWriteSpd](#)

**2.1.4.20.6 Example****2.1.4.21.7.1 Structured Text**

```
MLAxisAdd(PipeNetwork.Axis1, LREAL#750.0 ) ;
```

**2.1.4.22.8.2 Ladder Diagram**



**NOTE**

You must use a [pulse contact](#) to start the FB

**2.1.4.23.9.3 Function Block Diagram**





**2.1.4.24 MLAxisAddress** Pipe Network ✓

**2.1.4.25.1 Description**

Returns the motion bus address of the axis

**2.1.4.26.2 Arguments**

**2.1.4.27.3.1 Input**

<b>ID</b>	<b>Description</b>	ID name of the Axis Block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.28.4.2 Output**

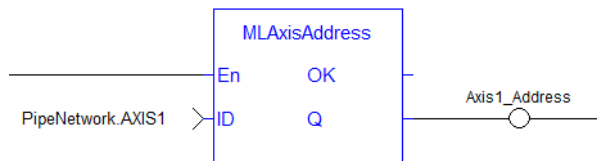
<b>OK</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Default (.Q)</b>	<b>Description</b>	Returns the motion bus address of the axis
	<b>Data type</b>	DINT
	<b>Unit</b>	n/a

**2.1.4.29.5 Example**

**2.1.4.30.6.1 Structured Text**

```
Axis1_Address := MLAxisAddress(PipeNetwork.AXIS1);
```

**2.1.4.31.7.2 Ladder Diagram**



**2.1.4.32.8.3 Function Block Diagram**



**2.1.4.33 MLAxisAddTq** Pipe Network ✓

**2.1.4.34.1 Description**

Allows the application to set the additive torque value to the drive output (Torque feed-forward).

This function is only active after the "MLAxisRatedTq" (p. 172) function has been invoked. Using the PDO, it also requires IL.KBUSFF value to be set to 1 in the drive.

**2.1.4.35.2 Arguments**

**2.1.4.36.3.1 Input**

<b>ID</b>	<b>Description</b>	Pipe network identifier of the axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Torque</b>	<b>Description</b>	Requested additive torque value in N.m (Newton meter).
	<b>Data type</b>	LREAL
	<b>Unit</b>	Rated torque units as used in the drive (i.e. rated motor continuous torque x the Torque factor).

**2.1.4.37.4.2 Output**

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

**2.1.4.38.5 Related Functions**

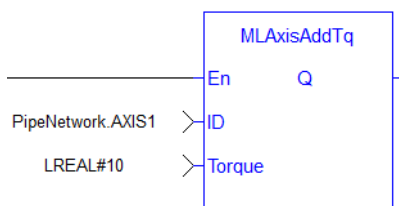
"MLAxisRatedTq" (p. 172)

**2.1.4.39.6 Example**

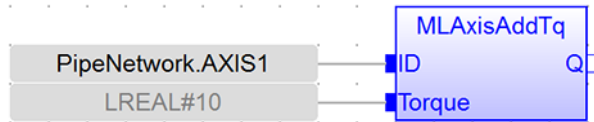
**2.1.4.40.7.1 Structured Text**

```
MLAxisAddTq(PipeNetwork.Axis1, LREAL#10 ) ;
```

**2.1.4.41.8.2 Ladder Diagram**



### 2.1.4.42.9.3 Function Block Diagram



### 2.1.4.43 MLAxisCfgFastIn Pipe Network ✓

#### 2.1.4.44.1 Description

Configures the Fast Input for the axis by writing the expected settings in the Latch Control Word. Fast input can be armed on falling or rising edge.

#### 2.1.4.45.2 Arguments

##### 2.1.4.46.3.1 Input

<b>En</b>	<b>Description</b>	Enables execution
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
	<b>Default</b>	-
<b>AxisID</b>	<b>Description</b>	ID name of the Axis Block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
<b>Default</b>	<b>Description</b>	—
	<b>Data type</b>	DINT
	<b>Range</b>	[0, 1]
	<b>Unit</b>	n/a
<b>InputID</b>	<b>Description</b>	ID of the FastInput of an axis, (ie IN1 and IN2 on S300) 0 = Capture Engine 0 1 = Capture Engine 1 Range is [0, 1] For information on configuring the capture engines, refer to <a href="#">AKD Capture Engine Configuration</a> .
	<b>Data type</b>	DINT
	<b>Range</b>	[0, 1]
	<b>Unit</b>	n/a
<b>Default</b>	<b>Description</b>	—
	<b>Data type</b>	DINT
	<b>Range</b>	[0, 2]
	<b>Unit</b>	n/a
<b>Mode</b>	<b>Description</b>	Configures the Fast Inputs as 0= Disabled, 1=Rising Edge, 2=Falling edge
	<b>Data type</b>	DINT
	<b>Range</b>	[0, 2]
	<b>Unit</b>	n/a
<b>Default</b>	<b>Description</b>	—
	<b>Data type</b>	DINT
	<b>Range</b>	[0, 2]
	<b>Unit</b>	n/a

##### 2.1.4.47.4.2 Output

<b>Q</b>	<b>Description</b>	Returns true when the function successfully executes. Returns false if the fast input could not be configured due to an invalid PDO mapping in the .XML file.
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

#### 2.1.4.48.5 Related Functions

[MLAxisIsTriggered](#)

[MLAxisRstFastIn](#)

#### 2.1.4.49.6 See Also

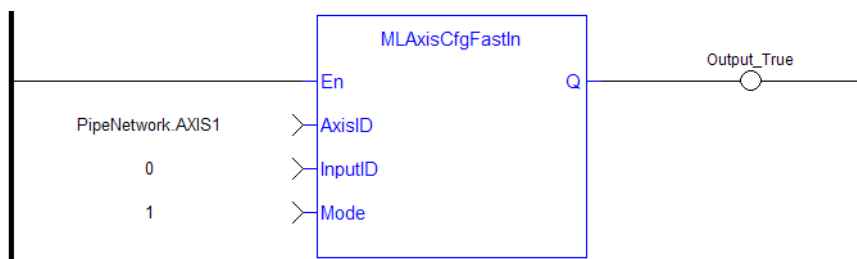
- [Fast Inputs with Pipe Network Motion](#)
- [Fast Inputs with PLCOpen](#)
- [Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block](#)
- [Fast Homing Example with the PLCOpen Motion Engine](#)
- [Pipe Network Registration and Fast Homing](#)
- [Registration Position Capture Example with Pipe Network Trigger Block](#)

#### 2.1.4.50.7 Example

##### 2.1.4.51.8.1 Structured Text

```
MLAxisCfgFastIn( PipeNetwork.Axis1, 0, 1 ) ;
```

##### 2.1.4.52.9.2 Ladder Diagram



##### 2.1.4.53.10.3 Function Block Diagram



See also "Fast inputs" for more details.

**2.1.4.54 MLAxisCmdPos** **2.1.4.55.1 Description**

Returns the reference position of the axis.

**2.1.4.56.2 Arguments****2.1.4.57.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.58.4.2 Output**

<b>OK</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Position</b>	<b>Description</b>	Returns the Axis reference position
	<b>Data type</b>	LREAL
	<b>Unit</b>	<a href="#">User unit</a>

**2.1.4.59.5 Related Functions**

[MLAxisReadActPos](#)

[MLAxisFBackPos](#)

[MLAxisGenPos](#)

[MLAxisPipePos](#)

[MLAxisWritePipPos](#)

**2.1.4.60.6 Previous Function Name**

MLAxisRefPos

**2.1.4.61.7 Example****2.1.4.62.8.1 Structured Text**

```
Axis1_ReferencePosition := MLAxisCmdPos(PipeNetwork.AXIS1);
```

**2.1.4.63.9.2 Ladder Diagram**



### 2.1.4.64.10.3 Function Block Diagram



**2.1.4.65 MLAxisCreate** Pipe Network ✓**2.1.4.66.1 Description**

Creates a new axis object. Returns the ID of the newly created axis object or 0 if the function failed

**TIP**

This function must be called or executed before "[MLMotionStart](#)" (p. 574) is called.

**2.1.4.67.2 Arguments****2.1.4.68.3.1 Input**

<b>Name</b>	<b>Description</b>	Name of the created Axis
	<b>Data type</b>	STRING
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>DriverName</b>	<b>Description</b>	Is the Motion bus driver name or Simulated
	<b>Data type</b>	STRING
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Address</b>	<b>Description</b>	Axis motion bus address
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ID</b>	<b>Description</b>	ID name of the Axis Block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.69.4.2 Output**

<b>OK</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL



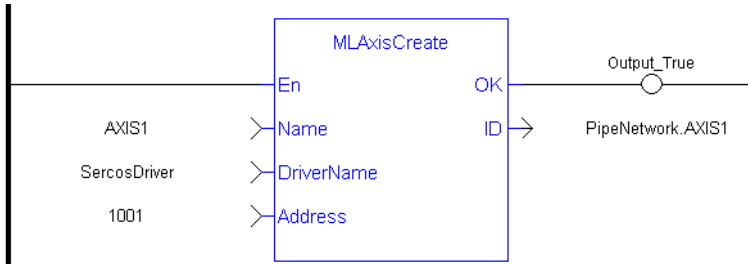
<b>Unit</b>	n/a
-------------	-----

**2.1.4.70.5 Example**

**2.1.4.71.6.1 Structured Text**

```
PipeNetwork.AXIS1 := MAxisCreate('AXIS1', 'SercosDriver',
1001);
```

**2.1.4.72.7.2 Ladder Diagram**



**2.1.4.73.8.3 Function Block Diagram**



**2.1.4.74 MLAxisFBackPos** Pipe Network ✓

**2.1.4.75.1 Description**

Returns the Feedback Position of the axis

**2.1.4.76.2 Arguments**

**2.1.4.77.3.1 Input**

<b>ID</b>	<b>Description</b>	ID name of the Axis Block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.78.4.2 Output**

<b>OK</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Position</b>	<b>Description</b>	Returns the Feedback Position of the axis
	<b>Data type</b>	LREAL
	<b>Unit</b>	<u>User unit</u>

**2.1.4.79.5 Related Functions**

[MLAxisReadActPos](#)

[MLAxisGenPos](#)

[MLAxisPipePos](#)

[MLAxisCmdPos](#)

[MLAxisWritePipPos](#)

**2.1.4.80.6 Example**

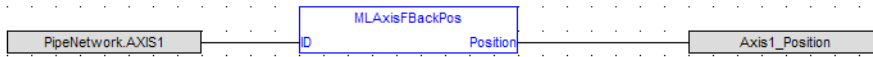
**2.1.4.81.7.1 Structured Text**

```
Axis1_Position := MLAxisFBackPos( PipeNetwork.Axis1 ) ;
```

**2.1.4.82.8.2 Ladder Diagram**



### 2.1.4.83.9.3 Function Block Diagram



**2.1.4.84 MLAxisGenEN** Pipe Network ✓

**2.1.4.85.1 Description**

Enables or disables the internal TMP generator of the axis.

**2.1.4.86.2 Arguments**

**2.1.4.87.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Enable</b>	<b>Description</b>	Boolean switch to activate the generator
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.88.4.2 Output**

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

**2.1.4.89.5 Related Functions**

[MLAxisGensEN](#)

**2.1.4.90.6 Example**

**2.1.4.91.7.1 Structured Text**

```
MLAxisGenEN( PipeNetwork.Axis1, true) ;
```

**2.1.4.92.8.2 Ladder Diagram**



### 2.1.4.93.9.3 Function Block Diagram



**2.1.4.94 MLAxisGenIsEN** Pipe Network ✓

**2.1.4.95.1 Description**

Check if the internal TMP generator of the axis is enable. Returns TRUE if the internal generator is enabled.

**2.1.4.96.2 Arguments**

**2.1.4.97.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.98.4.2 Output**

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

**2.1.4.99.5 Related Functions**

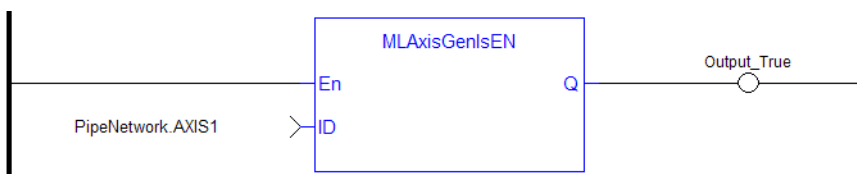
[MLAxisGenIsRdy](#)

**2.1.4.100.6 Example**

**2.1.4.101.7.1 Structured Text**

```
MLAxisGenIsEN(PipeNetwork.Axis1 ) ;
```

**2.1.4.102.8.2 Ladder Diagram**



**2.1.4.103.9.3 Function Block Diagram**



**2.1.4.104 MLAxisGenIsRdy** Pipe Network ✓**2.1.4.105.1 Description**

Check if an axis is ready. Returns TRUE if the internal generator axis is ready.

**2.1.4.106.2 Arguments****2.1.4.107.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.108.4.2 Output**

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

**2.1.4.109.5 Related Functions**

[MLAxisGenIsEN](#)

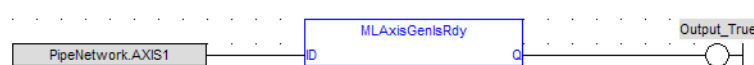
[MLAxisStatus](#)

**2.1.4.110.6 Example**

See "Usage Example of Axis Functions" (p. 208) for additional examples.

**2.1.4.111.7.1 Structured Text**

```
MLAxisGenIsRdy(PipeNetwork.Axis1 );
```

**2.1.4.112.8.2 Ladder Diagram****2.1.4.113.9.3 Function Block Diagram**

**2.1.4.114 MLAxisGenPos** Pipe Network ✓

**2.1.4.115.1 Description**

Returns the generator position of the axis Returns TRUE if the internal generator axis is ready.

**2.1.4.116.2 Arguments**

**2.1.4.117.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.118.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Position</b>	<b>Description</b>	Returns Axis generator position value
	<b>Data type</b>	LREAL
	<b>Unit</b>	<u>User unit</u>

**2.1.4.119.5 Related Functions**

[MLAxisReadActPos](#)

[MLAxisFBackPos](#)

[MLAxisPipePos](#)

[MLAxisCmdPos](#)

[MLAxisWritePipPos](#)

**2.1.4.120.6 Example**

**2.1.4.121.7.1 Structured Text**

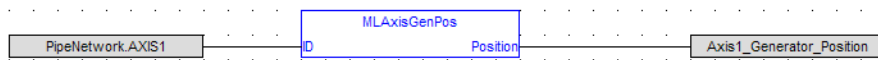
```
Axis1_Generator_Position := MLAxisGenPos (PipeNetwork.Axis1 ) ;
```

**2.1.4.122.8.2 Ladder Diagram**





### 2.1.4.123.9.3 Function Block Diagram



**2.1.4.124 MAxisGenReadAcc** Pipe Network ✓

**2.1.4.125.1 Description**

Get the acceleration of the internal generator of an axis.

**2.1.4.126.2 Arguments**

**2.1.4.127.3.1 Input**

<b>AxisID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.128.4.2 Output**

<b>OK</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Acceleration</b>	<b>Description</b>	Returns Axis Acceleration value
	<b>Data type</b>	LREAL
	<b>Unit</b>	<u>User unit</u> /sec <sup>2</sup>

**2.1.4.129.5 Related Functions**

[MLAxisGenReadDec](#)

[MLAxisGenReadSpd](#)

**2.1.4.130.6 Example**

**2.1.4.131.7.1 Structured Text**

```
Axis1_Acceleration := MAxisGenReadAcc( PipeNetwork.Axis1 );
```

**2.1.4.132.8.2 Ladder Diagram**



### 2.1.4.133.9.3 Function Block Diagram



**2.1.4.134 MLAxisGenReadDec** Pipe Network ✓

**2.1.4.135.1 Description**

Get the Deceleration of the internal generator of an axis.

**2.1.4.136.2 Arguments**

**2.1.4.137.3.1 Input**

<b>AxisID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.138.4.2 Output**

<b>OK</b>	<b>Description</b>	
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Deceleration</b>	<b>Description</b>	Returns Axis Deceleration value
	<b>Data type</b>	LREAL
	<b>Unit</b>	<u>User unit</u> /sec <sup>2</sup>

**2.1.4.139.5 Related Functions**

[MLAxisGenReadAcc](#)

[MLAxisGenReadSpd](#)

**2.1.4.140.6 Example**

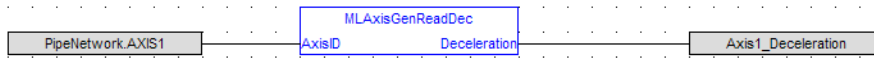
**2.1.4.141.7.1 Structured Text**

```
Axis1_Deceleration := MLAxisGenReadDec ( PipeNetwork.Axis1 );
```

**2.1.4.142.8.2 Ladder Diagram**



**2.1.4.143.9.3 Function Block Diagram**



**2.1.4.144 MLAxisGenReadSpd** Pipe Network ✓

**2.1.4.145.1 Description**

Get the speed of the internal generator of an axis.

**2.1.4.146.2 Arguments**

**2.1.4.147.3.1 Input**

<b>AxisID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.148.4.2 Output**

<b>OK</b>	<b>Description</b>	
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Speed</b>	<b>Description</b>	Returns Axis Speed value
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit/sec

**2.1.4.149.5 Related Functions**

[MLAxisGenReadDec](#)

[MLAxisGenReadAcc](#)

**2.1.4.150.6 Example**

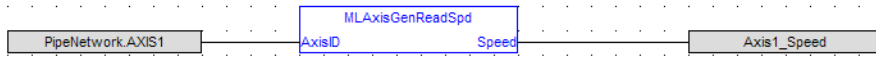
**2.1.4.151.7.1 Structured Text**

```
Axis1_Speed := MLAxisGenReadSpd( PipeNetwork.Axis1 ) ;
```

**2.1.4.152.8.2 Ladder Diagram**



**2.1.4.153.9.3 Function Block Diagram**



**2.1.4.154 MLAxisGenWriteAcc** **2.1.4.155.1 Description**

Set the acceleration of the internal generator of an axis Returns TRUE if the internal generator axis is ready.

**2.1.4.156.2 Arguments****2.1.4.157.3.1 Input**

<b>AxisID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Sets the generator Acceleration value
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a> /sec <sup>2</sup>
	<b>Default</b>	—

**2.1.4.158.4.2 Output**

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

**2.1.4.159.5 Related Functions**

[MLAxisGenWriteDec](#)

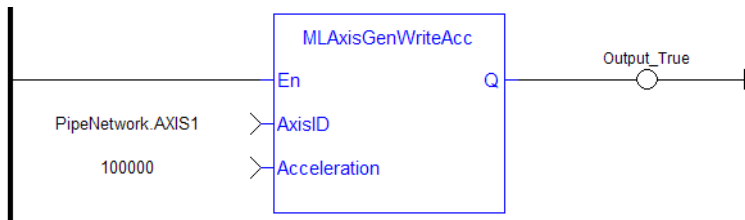
[MLAxisGenWriteSpd](#)

**2.1.4.160.6 Example****2.1.4.161.7.1 Structured Text**

```
MLAxisGenWriteAcc(PipeNetwork.Axis1, 100000 ) ;
```

**2.1.4.162.8.2 Ladder Diagram**





### 2.1.4.163.9.3 Function Block Diagram



**2.1.4.164 MLAxisGenWriteDec** Pipe Network ✓**2.1.4.165.1 Description**

Set the Deceleration of the internal generator of an axis Returns TRUE if the internal generator axis is ready.

**2.1.4.166.2 Arguments****2.1.4.167.3.1 Input**

<b>AxisID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Sets the generator Deceleration value.  The axis deceleration rate is limited such that the velocity cannot change by more than the value of the declared velocity limit in a single iteration.  The Pipe Network Axis block uses the TRAVEL_SPEED parameter to scale this limit. The maximum deceleration is therefore affected by the Pipe Network Axis Block parameter “TRAVEL_SPEED”, as well as the axis update rate.
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a> /sec <sup>2</sup>
	<b>Default</b>	—

**2.1.4.168.4.2 Output**

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

**2.1.4.169.5 Related Functions**

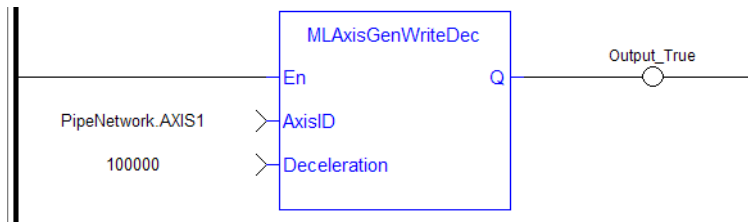
[MLAxisGenWriteAcc](#)

[MLAxisGenWriteSpd](#)

**2.1.4.170.6 Example****2.1.4.171.7.1 Structured Text**

```
MLAxisGenWriteDec(PipeNetwork.Axis1, 100000 ) ;
```

**2.1.4.172.8.2 Ladder Diagram**



### 2.1.4.173.9.3 Function Block Diagram



**2.1.4.174 MAxisGenWriteSpd** **2.1.4.175.1 Description**

Set the speed of the internal generator of an axis. Returns TRUE if the function succeeded. This function does not generate any motion.

**2.1.4.176.2 Arguments****2.1.4.177.3.1 Input**

<b>AxisID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Speed</b>	<b>Description</b>	Sets the generator Speed value
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a>
	<b>Default</b>	—

**2.1.4.178.4.2 Output**

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

**2.1.4.179.5 Related Functions**

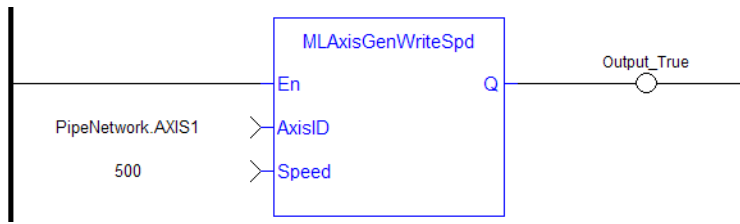
[MAxisGenWriteAcc](#)

[MAxisGenWriteDec](#)

**2.1.4.180.6 Example****2.1.4.181.7.1 Structured Text**

```
MAxisGenWriteSpd(PipeNetwork.Axis1, 500 ) ;
```

**2.1.4.182.8.2 Ladder Diagram**



### 2.1.4.183.9.3 Function Block Diagram



**2.1.4.184 MLAxisInit** **2.1.4.185.1 Description**

Initializes an axis object. Returns TRUE if the function succeeded

**2.1.4.186.2 Arguments****2.1.4.187.3.1 Input**

<b>AxisID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ModuloPosition</b>	<b>Description</b>	Value of the period of a cyclic system expressed in user units. The parameter is defined to correctly manage the periodicity (modulo) of the input values
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>UserUnitPerTurn</b>	<b>Description</b>	Define the unit which is equivalent to one revolution of the physical motor
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>FeedbackUnitPerTurn</b>	<b>Description</b>	
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Speed</b>	<b>Description</b>	Sets the Axis Speed
	<b>Data type</b>	LREAL
	<b>Range</b>	—

	<b>Unit</b>	<a href="#">User unit</a>
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Sets the Axis Acceleration value
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a> /sec <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Sets the Axis Deceleration value
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a> /sec <sup>2</sup>
	<b>Default</b>	—
<b>InitialPosition</b>	<b>Description</b>	Initial position value expressed in user logical units. Used only at the pipe activation to initialize the position starting point
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a>
	<b>Default</b>	—
<b>Modulo</b>	<b>Description</b>	Define the mode which can be Modulo (True) or not (False)
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 2.1.4.188.4.2 Output

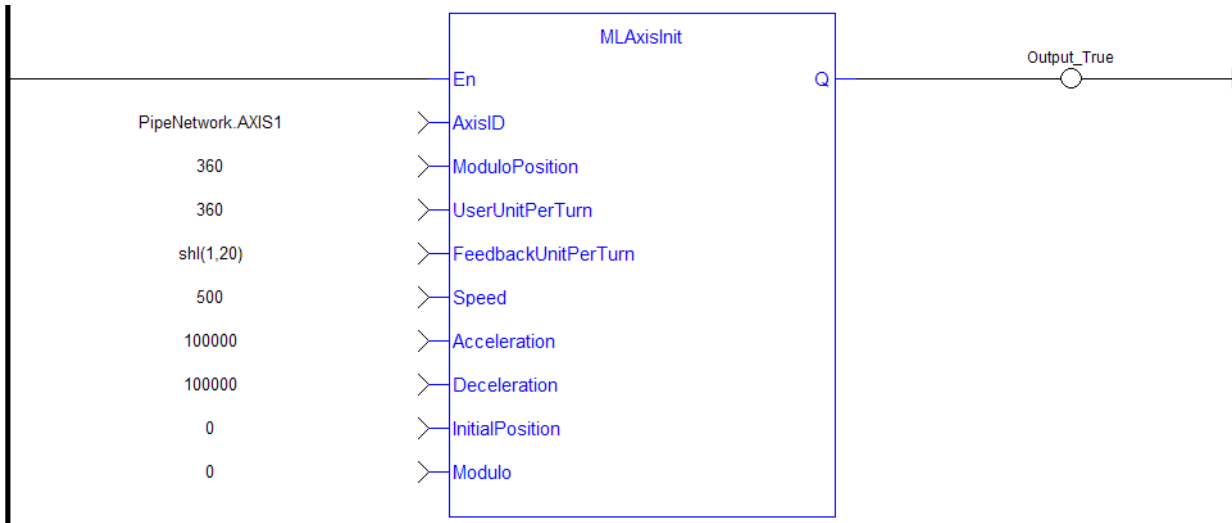
<b>Default (.Q)</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

#### 2.1.4.189.5 Example

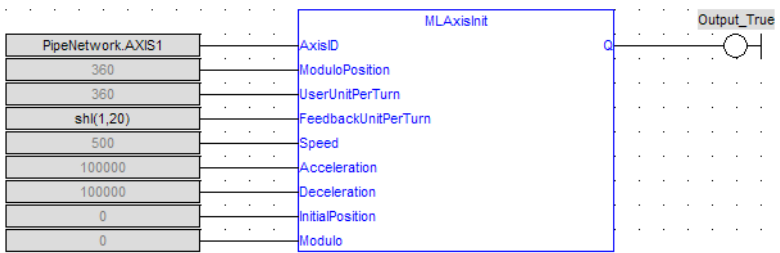
#### 2.1.4.190.6.1 Structured Text

```
MLAxisInit( PipeNetwork.Axis1, 360.0, 360.0, SHL(1,20), 500.0,
100000.0, 100000.0, 0.0, true ) ;
```

**2.1.4.191.7.2 Ladder Diagram**



**2.1.4.192.8.3 Function Block Diagram**





**2.1.4.193 MAxisIsCnctd** Pipe Network ✓

**2.1.4.194.1 Description**

Check if a pipe is currently connected to the axis. Returns TRUE if a pipe is connected.

**2.1.4.195.2 Arguments**

**2.1.4.196.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.197.4.2 Output**

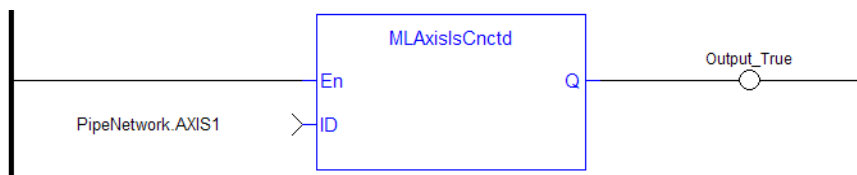
<b>Default (.Q)</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.4.198.5 Example**

**2.1.4.199.6.1 Structured Text**

```
MAxisIsCnctd(PipeNetwork.Axis1 ) ;
```

**2.1.4.200.7.2 Ladder Diagram**



**2.1.4.201.8.3 Function Block Diagram**



### 2.1.4.202 MLAxisIsTriggered

#### 2.1.4.203.1 Description

Checks if the axis got a trigger event. Returns TRUE if the Fast Input event has been **triggered** and not yet been reset. MLAxisCfgFastIn

#### 2.1.4.204.2 Arguments

##### 2.1.4.205.3.1 Input

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>InputID</b>	<b>Description</b>	ID of the triggered Fast input of an axis (ie IN1 and IN2 on S300)  0 = Capture Engine 0 1 = Capture Engine 1 Range is [0,1] For information on configuring the capture engines, refer to <a href="#">AKD Capture Engine Configuration</a> .
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>edge</b>	<b>Description</b>	Configures the Inputs as 0= Disabled, 1=Rising Edge, 2=Falling edge
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

##### 2.1.4.206.4.2 Output

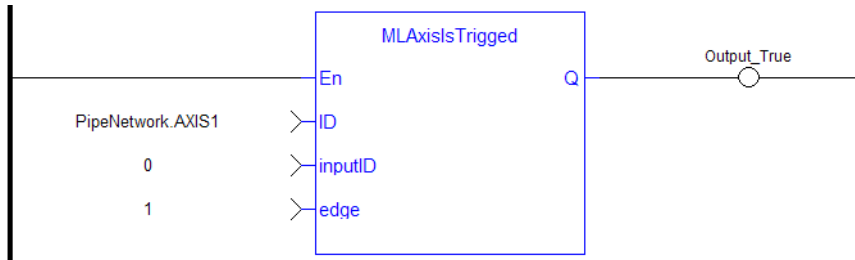
<b>Default (.Q)</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

#### 2.1.4.207.5 Related Functions

[MLAxisRstFastIn](#)

**2.1.4.208.6 Example****2.1.4.209.7.1 Structured Text**

```
MLAxisIsTriggered (PipeNetwork.Axis1, 0,1 ) ;
```

**2.1.4.210.8.2 Ladder Diagram****2.1.4.211.9.3 Function Block Diagram**

**2.1.4.212 MLAxisMoveVel** **2.1.4.213.1 Description**

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

**2.1.4.214.2 Arguments****2.1.4.215.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Speed</b>	<b>Description</b>	Sets the Axis Speed
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a> /sec
	<b>Default</b>	—

**2.1.4.216.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns true when function successfully executes, after the motion has reached jog speed
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.4.217.5 Related Functions**

[MLAxisGenWriteSpd](#)

[MLAxisGenWriteDec](#)

[MLAxisGenWriteAcc](#)

**2.1.4.218.6 Previous Function Name**

MLAxisRun

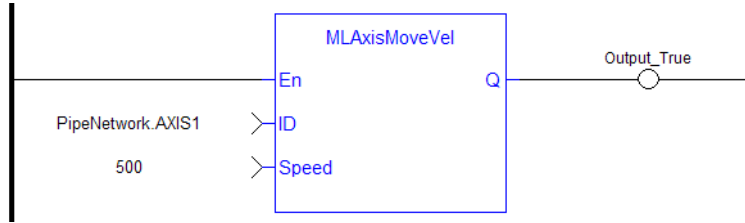
**2.1.4.219.7 Example**

See "[Usage Example of Axis Functions](#)" (p. 208) for additional examples.

**2.1.4.220.8.1 Structured Text**

```
MLAxisMoveVel (PipeNetwork.Axis1, 500 ) ;
```

### 2.1.4.221.9.2 Ladder Diagram



### 2.1.4.222.10.3 Function Block Diagram



**2.1.4.223 MLAxisPipePos** **2.1.4.224.1 Description**

Returns the pipe position of the axis.

**2.1.4.225.2 Arguments****2.1.4.226.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.227.4.2 Output**

<b>OK</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Position</b>	<b>Description</b>	
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a>

**2.1.4.228.5 Related Functions**

[MLAxisReadActPos](#)

[MLAxisFBackPos](#)

[MLAxisGenPos](#)

[MLAxisCmdPos](#)

[MLAxisWritePipPos](#)

**2.1.4.229.6 Example****2.1.4.230.7.1 Structured Text**

```
Axis1_Pipe_Position := MLAxisPipePos (PipeNetwork.Axis1 ) ;
```

**2.1.4.231.8.2 Ladder Diagram**



### 2.1.4.232.9.3 Function Block Diagram



**2.1.4.233 MLAxisPower** **2.1.4.234.1 Description**

Powers up or down the axis. Enable or disabled Axis Servo Drive.

When the axis is powered up, the **ReferencePosition** is modified to equal the **ActualPosition**. For that, KAS updates the **GeneratorPosition**.

**2.1.4.235.2 Arguments****2.1.4.236.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>On</b>	<b>Description</b>	Flag to power up (True) or down (False) the Axis
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.237.4.2 Output**

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

**2.1.4.238.5 Related Functions**

[MLAxisPowerDOff](#)

**2.1.4.239.6 Previous Function Name**

MLAxisPowerOn

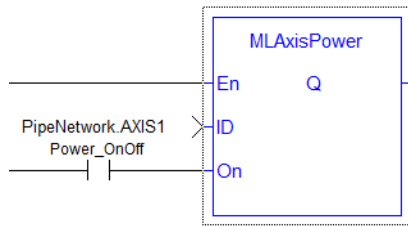
MLAxisPowerOff

**2.1.4.240.7 Example****2.1.4.241.8.1 Structured Text**

```
(* If Power_OnOff is TRUE then power in ON, otherwise OFF*)
MLAxisPower( PipeNetwork.Axis1, Power_OnOff) ;
```

**2.1.4.242.9.2 Ladder Diagram**





### 2.1.4.243.10.3 Function Block Diagram



**2.1.4.244 MLAxisPowerDOff** Pipe Network ✓

**2.1.4.245.1 Description**

Returns the adjustment of position done by the last power on to avoid bumps

**2.1.4.246.2 Arguments**

**2.1.4.247.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.248.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>PowerONDeltaOffset</b>	<b>Description</b>	
	<b>Data type</b>	LREAL
	<b>Unit</b>	<a href="#">User unit</a>

**2.1.4.249.5 Related Functions**

[MLAxisPower](#)

**2.1.4.250.6 Example**

**2.1.4.251.7.1 Structured Text**

```
Axis1_Power_On_Delta_Offset := MLAxisPowerDOff
(PipeNetwork.Axis1 ) ;
```

**2.1.4.252.8.2 Ladder Diagram**



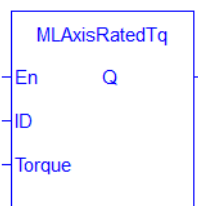
**2.1.4.253.9.3 Function Block Diagram**



### 2.1.4.254 MAxisRatedTq Pipe Network ✓

#### 2.1.4.255.1 Description

Allows conversion of drive torque values from rated torque units (1000 = rated motor continuous torque) to N.m (Newton meter).



#### 2.1.4.256.2 Arguments

##### 2.1.4.257.3.1 Input

<b>ID</b>	<b>Description</b>	Pipe network identifier of the axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Torque</b>	<b>Description</b>	<p>Actual torque applied by the drive associated to the axis</p> <p>Rated torque = Nominal Drive Current * Torque factor = <b>DRV.ICONT</b> * MOTOR.KT</p> <p><b>About SDO</b></p> <p>DRV.ICONT is obtained by SDO parameter: index 5083h (sub-index 0)</p> <p>MOTOR.KT is obtained by SDO parameter: index 3593h (sub-index 0)</p> <p>For more details, refer to:</p> <ul style="list-style-type: none"> <li>• Communication SDOs</li> <li>• Manufacturer specific SDOs</li> <li>• Profile specific SDOs</li> </ul> <p>To read/write an SDO object with an index greater than 16#7FFF (32767), the value must be entered in the form <code>any_to_int(index # in hex format)</code>. For example <code>any_to_int(16#8321)</code>.</p> <p>The actual units of DRV.ICONT and MOTOR.KT are 1/1000 of the actual values if obtained by SDO. So the formula, if using the SDO values, is:</p> <p>Rated Torque = Torque = (SDO(DRV.ICONT)/1000) * (SDO(MOTOR.KT)/1000)</p>
	<b>Data type</b>	LREAL
	<b>Unit</b>	N.m (Newton meter)

**2.1.4.258.4.2 Output**

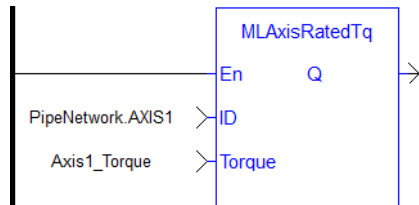
<b>Default (.Q)</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.4.259.5 Related Functions**

[MLAxisReadTq](#)

**2.1.4.260.6 Example****2.1.4.261.7.1 Structured Text**

```
MLAxisRatedTq(PipeNetwork.Axis1, Axis1_Torque ) ;
```

**2.1.4.262.8.2 Ladder Diagram****2.1.4.263.9.3 Function Block Diagram**

**2.1.4.264 MLAxisRead2ndFB** Pipe Network ✓

**2.1.4.265.1 Description**

Return the position given by the secondary feedback device of the drive mapped to the specified axis.

**2.1.4.266.2 Arguments**

**2.1.4.267.3.1 Input**

<b>ID</b>	<b>Description</b>	Pipe network identifier of the axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.268.4.2 Output**

<b>Position</b>	<b>Description</b>	Position value returned by the secondary feedback
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit

**2.1.4.269.5 Related Functions**

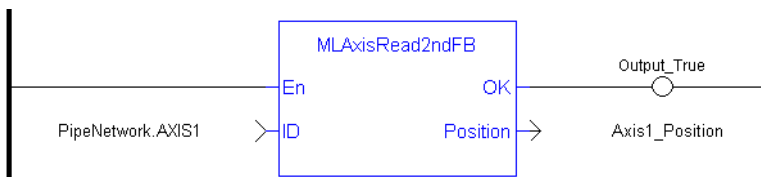
[MLAxisReadActPos](#)

**2.1.4.270.6 Example**

**2.1.4.271.7.1 Structured Text**

```
Axis1_Position := MLAxisRead2ndFB ( PipeNetwork.Axis1 ) ;
```

**2.1.4.272.8.2 Ladder Diagram**



**2.1.4.273.9.3 Function Block Diagram**



**2.1.4.274 MLAxisReadActPos** Pipe Network ✓**2.1.4.275.1 Description**

Returns the Actual Position of the axis

**2.1.4.276.2 Arguments****2.1.4.277.3.1 Input**

<b>ID</b>	<b>Description</b>	ID name of the Axis Block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.278.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Position</b>	<b>Description</b>	Returns the absolute position of the axis
	<b>Data type</b>	LREAL
	<b>Unit</b>	<a href="#">User unit</a>

**2.1.4.279.5 Related Functions**

[MLAxisFBackPos](#)

[MLAxisGenPos](#)

[MLAxisPipePos](#)

[MLAxisCmdPos](#)

[MLAxisWritePipPos](#)

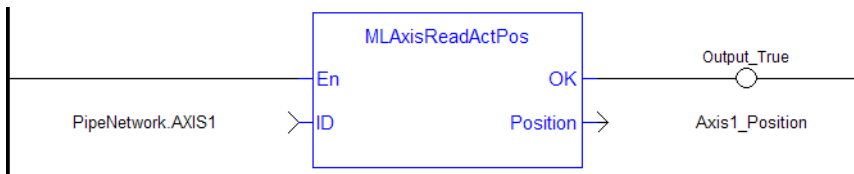
**2.1.4.280.6 Previous Function Name**

MLAxisActualPos

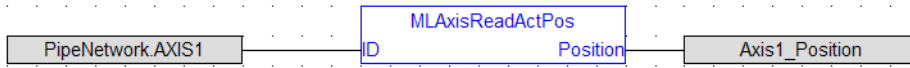
**2.1.4.281.7 Example****2.1.4.282.8.1 Structured Text**

```
Axis1_Position := MLAxisReadActPos( PipeNetwork.Axis1 ) ;
```

**2.1.4.283.9.2 Ladder Diagram**



### 2.1.4.284.10.3 Function Block Diagram





**2.1.4.285 MAxisReadFBUnit** Pipe Network ✓

**2.1.4.286.1 Description**

Get the feedback units per revolution value of the axis

**2.1.4.287.2 Arguments**

**2.1.4.288.3.1 Input**

<b>AxisID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.289.4.2 Output**

<b>OK</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>FBUnitsPerRev</b>	<b>Description</b>	Returns the Axis Feedback Units per revolution
	<b>Data type</b>	LREAL
	<b>Unit</b>	n/a

**2.1.4.290.5 Example**

**2.1.4.291.6.1 Structured Text**

```
Axis1_Feedback_Units := MAxisReadFBUnit(PipeNetwork.Axis1 ) ;
```

**2.1.4.292.7.2 Ladder Diagram**



**2.1.4.293.8.3 Function Block Diagram**



**2.1.4.294 MLAxisReadFEUU** Pipe Network ✓

**2.1.4.295.1 Description**

Return the difference between the reference position and the actual position of the drive mapped to the specified axis

**2.1.4.296.2 Arguments**

**2.1.4.297.3.1 Input**

<b>ID</b>	<b>Description</b>	Pipe network identifier of the axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.298.4.2 Output**

<b>Error</b>	<b>Description</b>	Difference between the reference position and the actual position of the drive associated to the axis
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit

**2.1.4.299.5 Related Functions**

[MLAxisReadActPos](#)

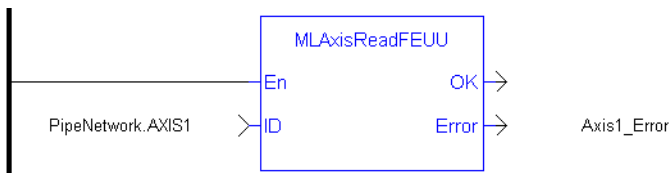
[ECATGetStatus](#)

**2.1.4.300.6 Example**

**2.1.4.301.7.1 Structured Text**

```
Axis1_Error := MLAxisReadFEUU (PipeNetwork.Axis1 ) ;
```

**2.1.4.302.8.2 Ladder Diagram**



**2.1.4.303.9.3 Function Block Diagram**



**2.1.4.304 MLAxisReadGenStatus** **2.1.4.305.1 Description**

Returns the status of the internal generator of the axis.

<b>0</b>	RUN mode (acceleration)
<b>1</b>	RUNNING or STOPPED
<b>2</b>	MOVE: Changing move destination
<b>3</b>	MOVE: Changing move destination
<b>4</b>	MOVE: Acceleration
<b>5</b>	MOVE: Constant speed (travel speed)
<b>6</b>	MOVE: Deceleration
<b>7</b>	MOVE: Single step (micro movement)

**2.1.4.306.2 Arguments****2.1.4.307.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.308.4.2 Output**

<b>OK</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Default (.Q)</b>	<b>Description</b>	Shows the status of the internal generator based on the table at the top of this topic
	<b>Data type</b>	DINT
	<b>Unit</b>	n/a

**2.1.4.309.5 Related Functions**

[MLAxisGenIsRdy](#)

[MLAxisStatus](#)

**2.1.4.310.6 Previous Function Name**

MLAxisGenStatus

**2.1.4.311.7 Example****2.1.4.312.8.1 Structured Text**

```
MLAxisReadGenStatus (PipeNetwork.Axis1 ) ;
```

### 2.1.4.313.9.2 Ladder Diagram



### 2.1.4.314.10.3 Function Block Diagram



**2.1.4.315 MAxisReadModPos** Pipe Network ✓

**2.1.4.316.1 Description**

Get the value period of the axis.

**2.1.4.317.2 Arguments**

**2.1.4.318.3.1 Input**

<b>AxisID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.319.4.2 Output**

<b>OK</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>ModuloPosition</b>	<b>Description</b>	Returns the Axis Value Period
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit

**2.1.4.320.5 Example**

**2.1.4.321.6.1 Structured Text**

```
Axis1_Value_Period := MAxisReadModPos (PipeNetwork.Axis1 ) ;
```

**2.1.4.322.7.2 Ladder Diagram**



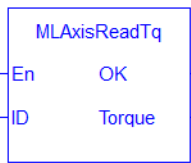
**2.1.4.323.8.3 Function Block Diagram**



**2.1.4.324 MLAxisReadTq** Pipe Network ✓

**2.1.4.325.1 Description**

Return the actual torque applied by the drive which is mapped to the specified axis.



**2.1.4.326.2 Arguments**

**2.1.4.327.3.1 Input**

<b>ID</b>	<b>Description</b>	Pipe network identifier of the axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.328.4.2 Output**

<b>Torque</b>	<b>Description</b>	Actual torque applied by the drive associated to the axis in N.m (Newton meter). If you have <b>not</b> previously invoked the " <a href="#">MLAxisRatedTq</a> " (p. 172) function, the Output value is rated motor continuous torque (where 1000.0 = rated torque)
	<b>Data type</b>	LREAL
	<b>Unit</b>	N.m (Newton meter)

**2.1.4.329.5 Related Functions**

[MLAxisRatedTq](#)

[MLAxisReadActPos](#)

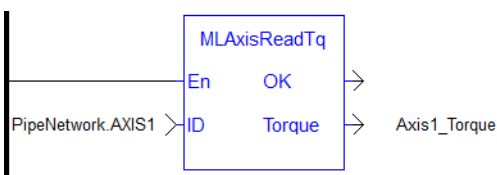
[MLAxisReadVel](#)

**2.1.4.330.6 Example**

**2.1.4.331.7.1 Structured Text**

```
Axis1_Torque := MLAxisReadTq(PipeNetwork.Axis1 ) ;
```

**2.1.4.332.8.2 Ladder Diagram**



### 2.1.4.333.9.3 Function Block Diagram



**2.1.4.334 MAxisReadUUnits** Pipe Network ✓

**2.1.4.335.1 Description**

Get the User units per revolution value of the axis

**2.1.4.336.2 Arguments**

**2.1.4.337.3.1 Input**

<b>AxisID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.338.4.2 Output**

<b>OK</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>UserUnitsPerRev</b>	<b>Description</b>	Returns the Axis User Units per revolution
	<b>Data type</b>	LREAL
	<b>Unit</b>	n/a

**2.1.4.339.5 Example**

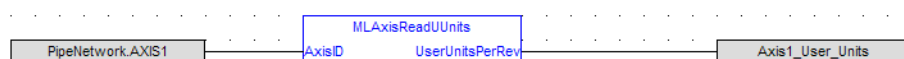
**2.1.4.340.6.1 Structured Text**

```
Axis1_User_Units := MAxisReadUUnits (PipeNetwork.Axis1 ) ;
```

**2.1.4.341.7.2 Ladder Diagram**



**2.1.4.342.8.3 Function Block Diagram**





**2.1.4.343 MLAxisReadVel** Pipe Network ✓**2.1.4.344.1 Description**

Return the actual velocity of the axis as calculated internally by the drive mapped to it, based on the data provided by the feedback device of the drive.

**2.1.4.345.2 Arguments****2.1.4.346.3.1 Input**

<b>ID</b>	<b>Description</b>	Pipe network identifier of the axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.347.4.2 Output**

<b>Velocity</b>	<b>Description</b>	Actual velocity returned by the drive associated to the axis
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit/sec

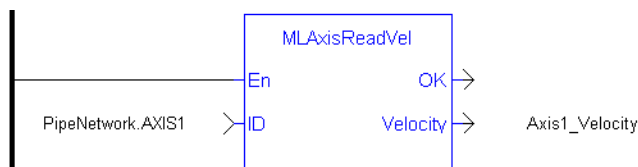
**2.1.4.348.5 Related Functions**

[MLAxisReadActPos](#)

[MLAxisReadTq](#)

**2.1.4.349.6 Example****2.1.4.350.7.1 Structured Text**

```
Axis1_Velocity := MLAxisReadVel(PipeNetwork.Axis1 ) ;
```

**2.1.4.351.8.2 Ladder Diagram****2.1.4.352.9.3 Function Block Diagram**

**2.1.4.353 MLAxisReAlgnRdy** Pipe Network ✓

**2.1.4.354.1 Description**

Check if an axis is ready. Returns TRUE if the internal realignment axis is ready.

**2.1.4.355.2 Arguments**

**2.1.4.356.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.357.4.2 Output**

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

**2.1.4.358.5 Related Functions**

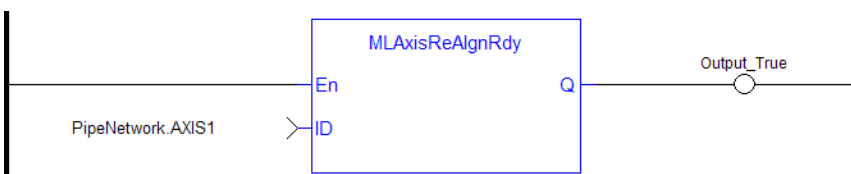
[MLAxisReAlign](#)

**2.1.4.359.6 Example**

**2.1.4.360.7.1 Structured Text**

```
MLAxisReAlgnRdy(PipeNetwork.Axis1 ) ;
```

**2.1.4.361.8.2 Ladder Diagram**



**2.1.4.362.9.3 Function Block Diagram**



**2.1.4.363 MLAxisReAlign** **2.1.4.364.1 Description**

When stopping the drive a motion profile is applied to decelerate. During the deceleration, the Reference position changes. Calling MLAxisReAlign realigns the actual position with the reference position by moving the axis by the specified delta position, which is typically calculated by the application code. After a [MLAxisStop](#) is executed, a MLAxisReAlign is required for the Pipe Position to be used again.

The function returns TRUE if it succeeds.

**NOTE**

The realign function do not work properly if the [MLAxisStop](#) function is continuously executed via its Start input

**2.1.4.365.2 Arguments****2.1.4.366.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Sets the Realign Acceleration
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a> /sec <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Sets the Realign Deceleration rate
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a> /sec <sup>2</sup>
	<b>Default</b>	—
<b>Speed</b>	<b>Description</b>	Sets the Axis Speed
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a> /sec
	<b>Default</b>	—

<b>DeltaPos</b>	<b>Description</b>	Sets the Axis Delta Position, or the relative distance to be moved
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a>
	<b>Default</b>	—

**2.1.4.367.4.2 Output**

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

**2.1.4.368.5 Related Functions**

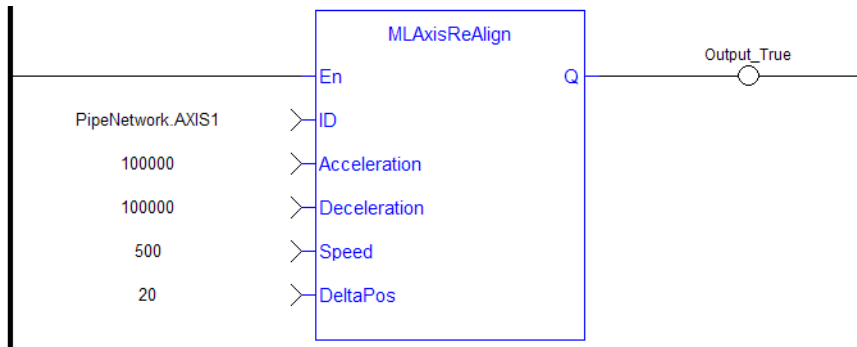
[MLAxisReAlignRdy](#)

**2.1.4.369.6 Example**

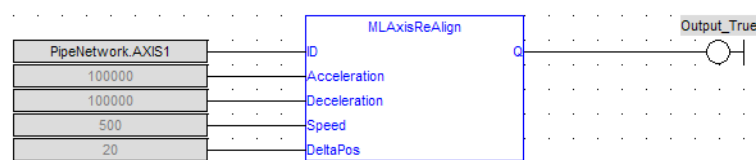
**2.1.4.370.7.1 Structured Text**

```
MLAxisReAlign(PipeNetwork.Axis1, 100000, 100000, 500, 20 ) ;
```

**2.1.4.371.8.2 Ladder Diagram**



**2.1.4.372.9.3 Function Block Diagram**



**2.1.4.373 MAxisRel** **2.1.4.374.1 Description**

A selected Axis performs a move for a specified distance relative to the current position. The DeltaPosition input is signed so that the move can be in the positive or negative direction, and the Axis moves this distance in user units. The travel speed, acceleration, deceleration, and User Units of the move are values inherited from the selected Axis. The default settings are entered when an Axis is created and initiated, and can be changed with other MAxis commands such as [MLAxisGenWriteSpd](#), [MLAxisGenWriteAcc](#), and [MLAxisWriteUUnits](#).

**NOTE**

If you wish to know when a move has completed, we recommend using [MLAxisGenIsRdy](#). The output of MAxisRel can occur before moves have finished.

**2.1.4.375.2 Arguments****2.1.4.376.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>DeltaPosition</b>	<b>Description</b>	Sets the Axis Delta Position, or the relative distance to be moved
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a>
	<b>Default</b>	—

**2.1.4.377.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns true when function successfully executes. This occurs immediately after the function is called; the function does not wait for the motion profile to be completed.
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.4.378.5 Related Functions**

[MLAxisGenWriteAcc](#)

[MLAxisGenWriteDec](#)

[MLAxisGenWriteSpd](#)

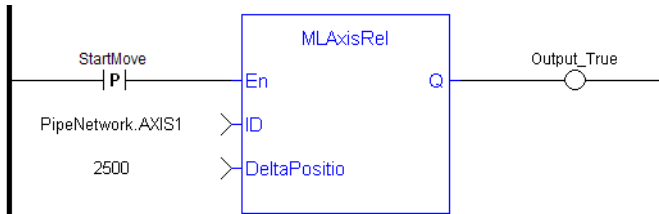
**2.1.4.379.6 Example**

See "[Usage Example of Axis Functions](#)" (p. 208) for additional examples.

### 2.1.4.380.7.1 Structured Text

```
MLAxisRel (PipeNetwork.Axis1, 2500 ) ;
```

### 2.1.4.381.8.2 Ladder Diagram



#### NOTE

You must use a [pulse contact](#) to start the FB

### 2.1.4.382.9.3 Function Block Diagram



**2.1.4.383 MAxisResetErrors** Pipe Network ✓**2.1.4.384.1 Description**

Clears errors of the specified axis

**2.1.4.385.2 Arguments****2.1.4.386.3.1 Input**

<b>ID</b>	<b>Description</b>	ID name of the Axis Block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.387.4.2 Output**

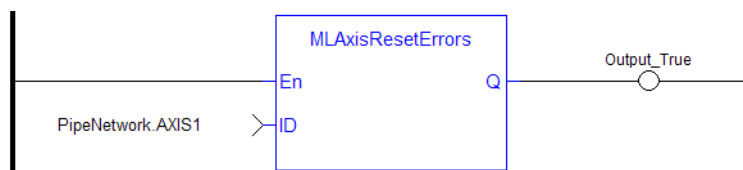
<b>Default (.Q)</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.4.388.5 Previous Function Name**

MAxisClrErrors

**2.1.4.389.6 Example****2.1.4.390.7.1 Structured Text**

```
MAxisResetErrors( PipeNetwork.Axis1 ) ;
```

**2.1.4.391.8.2 Ladder Diagram****2.1.4.392.9.3 Function Block Diagram**

**2.1.4.393 MLAxisRstFastIn** **2.1.4.394.1 Description**

Write in the Latch Control Word to reset the Fast Input.

**2.1.4.395.2 Arguments****2.1.4.396.3.1 Input**

<b>AxisID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>InputID</b>	<b>Description</b>	ID name of the Fast input to be reset on an axis, (ie IN1 and IN2 on S300) 0 = Capture Engine 0 1 = Capture Engine 1 Range is [0,1] For information on configuring the capture engines, refer to <a href="#">AKD Capture Engine Configuration</a> .
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.397.4.2 Output**

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

**2.1.4.398.5 Related Functions**

[MLAxisCfgFastIn](#)

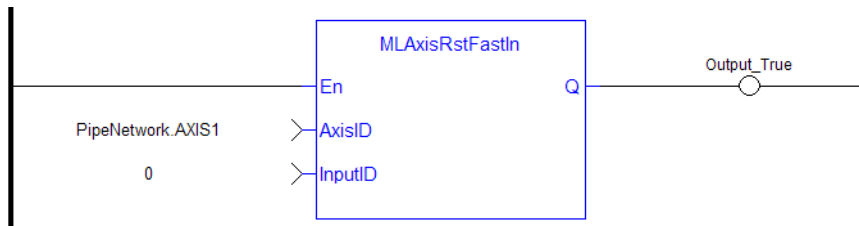
[MLAxisIsTriggered](#)

**2.1.4.399.6 Example****2.1.4.400.7.1 Structured Text**

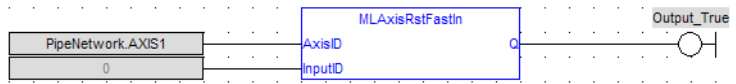
```
MLAxisRstFastIn (PipeNetwork.Axis1, 0 ) ;
```

**2.1.4.401.8.2 Ladder Diagram**





**2.1.4.402.9.3 Function Block Diagram**



**2.1.4.403 MLAxisStatus** Pipe Network ✓

**2.1.4.404.1 Description**

Returns the status of the axis.

**2.1.4.405.2 Arguments**

**2.1.4.406.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.407.4.2 Output**

<b>OK</b>	<b>Description</b>	Returns true when function successfully executes																												
	<b>Data type</b>	BOOL																												
	<b>Unit</b>	n/a																												
<b>Default (.Q)</b>	<b>Description</b>	Returns the status of the axis																												
		<table border="1"> <thead> <tr> <th>Bit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Initialized (1 if initialized)</td> </tr> <tr> <td>1</td> <td>Power (1 if power is on) Is linked to bit 1 (Switched on) of the Status Word For more information on the status machine</td> </tr> <tr> <td>2</td> <td>Enabled (1 if enabled) Is linked to bit 0 (Ready to switch on) of the Status Word</td> </tr> <tr> <td>3</td> <td>Found (1 if found on the network). EtherCAT state is Pre-Operational, see <a href="#">State Machine</a>.</td> </tr> <tr> <td>4</td> <td>Configured (1 if configured) EtherCAT state is Safe-Operational, see <a href="#">State Machine</a>.</td> </tr> <tr> <td>5</td> <td>Running (1 if running) EtherCAT state is Operational, see <a href="#">State Machine</a>.</td> </tr> <tr> <td>6</td> <td>Error (1 if in error)</td> </tr> <tr> <td>7</td> <td>Simulated (1 if working with a simulated axis)</td> </tr> <tr> <td>8</td> <td>Connected (1 if a pipe is connected)</td> </tr> <tr> <td>9</td> <td>Warning (1 if the drive signals a warning)</td> </tr> <tr> <td>10</td> <td>Stopping (1 if the drive is performing a Stop)</td> </tr> <tr> <td>11</td> <td>Stopped (1 if the drive has finished the Stop)</td> </tr> <tr> <td>12 to 31</td> <td>Reserved</td> </tr> </tbody> </table>	Bit	Description	0	Initialized (1 if initialized)	1	Power (1 if power is on) Is linked to bit 1 (Switched on) of the Status Word For more information on the status machine	2	Enabled (1 if enabled) Is linked to bit 0 (Ready to switch on) of the Status Word	3	Found (1 if found on the network). EtherCAT state is Pre-Operational, see <a href="#">State Machine</a> .	4	Configured (1 if configured) EtherCAT state is Safe-Operational, see <a href="#">State Machine</a> .	5	Running (1 if running) EtherCAT state is Operational, see <a href="#">State Machine</a> .	6	Error (1 if in error)	7	Simulated (1 if working with a simulated axis)	8	Connected (1 if a pipe is connected)	9	Warning (1 if the drive signals a warning)	10	Stopping (1 if the drive is performing a Stop)	11	Stopped (1 if the drive has finished the Stop)	12 to 31	Reserved
Bit	Description																													
0	Initialized (1 if initialized)																													
1	Power (1 if power is on) Is linked to bit 1 (Switched on) of the Status Word For more information on the status machine																													
2	Enabled (1 if enabled) Is linked to bit 0 (Ready to switch on) of the Status Word																													
3	Found (1 if found on the network). EtherCAT state is Pre-Operational, see <a href="#">State Machine</a> .																													
4	Configured (1 if configured) EtherCAT state is Safe-Operational, see <a href="#">State Machine</a> .																													
5	Running (1 if running) EtherCAT state is Operational, see <a href="#">State Machine</a> .																													
6	Error (1 if in error)																													
7	Simulated (1 if working with a simulated axis)																													
8	Connected (1 if a pipe is connected)																													
9	Warning (1 if the drive signals a warning)																													
10	Stopping (1 if the drive is performing a Stop)																													
11	Stopped (1 if the drive has finished the Stop)																													
12 to 31	Reserved																													
	<b>Data type</b>	DINT																												

<b>Unit</b>	n/a
-------------	-----

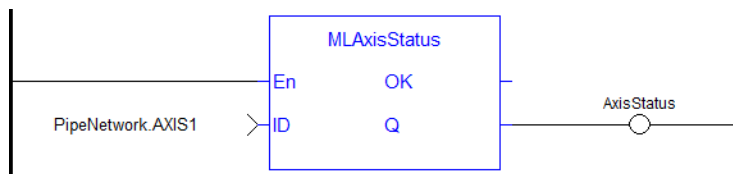
### 2.1.4.408.5 Example

#### 2.1.4.409.6.1 Structured Text

```
AxisStatus := MAxisStatus(PipeNetwork.AXI_A1_Axis) ;
IF AxisStatus.11 THEN
    MAxisStop(PipeNetwork.AXI_A1_Axis, FALSE, DEF_A1_StopDec) ;
END_IF;
```

```
AxisStatus := MAxisStatus(PipeNetwork.AXIS1);
If AxisStatus.0 Then
    (*Axis is initialized*)
ElsIf AxisStatus.1 Then
    (*Axis' power is ON*)
ElsIf AxisStatus.2 Then
    (*Axis is READY to be enabled*)
End_If;
```

#### 2.1.4.410.7.2 Ladder Diagram



#### 2.1.4.411.8.3 Function Block Diagram



**2.1.4.412 MLAxisStop** **2.1.4.413.1 Description**

Stop with the specified deceleration.

After stopping the drive, you need to restart the motion by realigning the actual position with the reference position

The purpose of the MLAxisStop Command is not to remove the input source, but to stop the drive from continuing to move.

When the stop occurs, the master keeps moving and the axis starts ignoring the Pipe Position value and begins a controlled stop based on the input parameters. Also at that point, any Axis Block level profile (issued from FB like MLAxisAbs, MLAxisRel...) are aborted. When the stop is complete, it is up to the application to decide how to move the axis, master, or both to a position where they can be realigned, and the master restarted.

The [realign](#) function is used to move the axis to a restart position in order to enable synchronized machine motion to start again. Once the realign function is successfully completed, the Pipe Position is again summed with the Generator Position to create the Reference Position.

**2.1.4.414.2 Arguments****2.1.4.415.3.1 Input**

<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Start</b>	<b>Description</b>	
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a> /sec <sup>2</sup>
	<b>Default</b>	—

**2.1.4.416.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Comes true when the Axis is completely stopped.
---------------------	--------------------	---

	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>PipePos</b>	<b>Description</b>	Corresponds to the Pipe Position input to the axis at the time the stop is triggered.
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit
<b>GenPos</b>	<b>Description</b>	Corresponds to the Generator Position input to the axis at the time the stop is triggered.
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit
<b>RealignPos</b>	<b>Description</b>	Realign Position is the Reference Position at which the stop is triggered. The Realign Position is obtained by converting the last value sent to the drive from drive interface units into user units. The Realign Position is useful if you want to return to the point at which the trajectory was abandoned, or in case you need to realign the master to the slave.
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit
<b>StopPos</b>	<b>Description</b>	Corresponds to the last Reference Position sent to the drive at the time when the Axis is completely stopped. It is functionally different than the Actual Position because that position is the drive position converted to user units. The correct delta for the realign move to get in sync with the trajectory in order to realign the slave to the master is the current Reference Position minus the <b>Stop Position</b> for the realign move. After stopping, if the axis is disabled and the motor position is manually altered, this distance must be taken into account when performing the realign.
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit

#### 2.1.4.417.5 Related Functions

[MLAxisReAlign](#)

#### 2.1.4.418.6 Example

#### 2.1.4.419.7.1 Structured Text

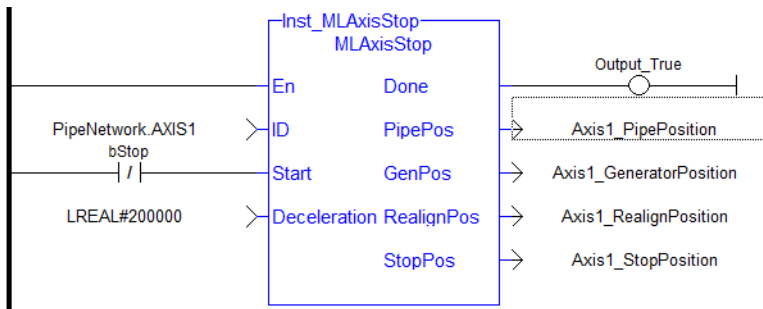
```

Inst_MLAxisStop(PipeNetwork.AXIS1, bStop, 200000);
If Inst_MLAxisStop.Done Then
  Axis1_PipePosition      := Inst_MLAxisStop.PipePos;
  Axis1_GeneratorPosition := Inst_MLAxisStop.GenPos;
  Axis1_RealignPosition   := Inst_MLAxisStop.RealignPos;

```

```
Axis1_StopPosition := Inst_MLAxisStop.StopPos;
End_if;
```

### 2.1.4.420.8.2 Ladder Diagram



### 2.1.4.421.9.3 Function Block Diagram



**2.1.4.422 MLAxisTimeStamp** **2.1.4.423.1 Description**

Returns the timestamp of the triggered axis.

**2.1.4.424.2 Arguments****2.1.4.425.3.1 Input**

<b>En</b>	<b>Description</b>	Enables execution
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ID</b>	<b>Description</b>	ID Name of the Axis block
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>InputID</b>	<b>Description</b>	ID of the triggered Fast input of an axis, 0=first , 1=second (ie IN1 and IN2 on S300)
	<b>Data type</b>	DINT
	<b>Range</b>	[0, 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>edge</b>	<b>Description</b>	Configures the Inputs as 0= Disabled, 1=Rising Edge, 2=Falling edge
	<b>Data type</b>	DINT
	<b>Range</b>	[0, 2]
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.4.426.4.2 Output**

<b>OK</b>	<b>Description</b>	Returns true when function successfully executes.
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

<b>Q</b>	<b>Description</b>	Returns the time stamp value. This value is explained in <a href="#">How to interpret the timestamp.</a>
	<b>Data type</b>	DINT
	<b>Unit</b>	microseconds

**2.1.4.427.5 Related Functions**

[MLAxisCfgFastIn](#)

[MLAxisRstFastIn](#)

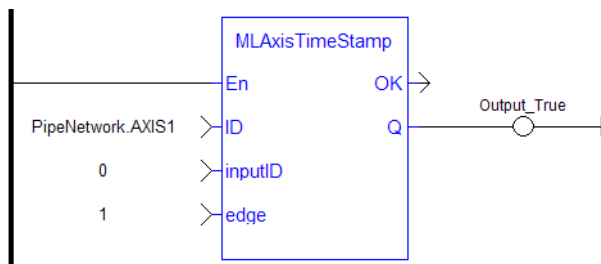
[MLAxisIsTriggered](#)

**2.1.4.428.6 Example**

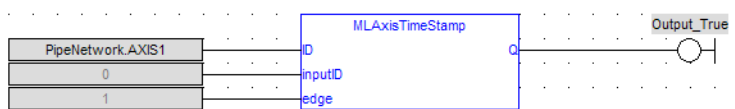
**2.1.4.429.7.1 Structured Text**

```
MLAxisTimeStamp (PipeNetwork.Axis1, 0, 1 ) ;
```

**2.1.4.430.8.2 Ladder Diagram**



**2.1.4.431.9.3 Function Block Diagram**





**2.1.4.432 MAxisWriteModPos** Pipe Network ✓

**2.1.4.433.1 Description**

Set the value period of the axis. Returns TRUE if the function succeeded.

**2.1.4.434.2 Arguments**

**2.1.4.435.3.1 Input**

AxisID	Description Data type Range Unit Default	ID Name of the Axis block DINT — n/a —
ModuloPosition	Description Data type Range Unit Default	Sets the Axis Period Value when Mode is set to Modulo. LREAL — User unit —

**2.1.4.436.4.2 Output**

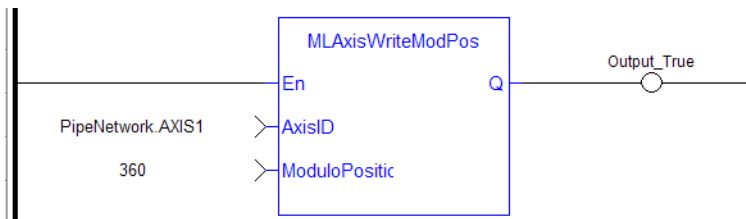
Default (.Q)	Description Data type Unit	Returns true when function successfully executes BOOL n/a
--------------	----------------------------------	---

**2.1.4.437.5 Example**

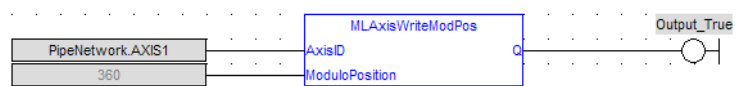
**2.1.4.438.6.1 Structured Text**

```
MAxisWriteModPos (PipeNetwork.Axis1, 360) ) ;
```

**2.1.4.439.7.2 Ladder Diagram**



**2.1.4.440.8.3 Function Block Diagram**



**2.1.4.441 MLAxisWritePipPos** Pipe Network ✓

**2.1.4.442.1 Description**

Force the pipe position internal value. This function is working only when no pipe is connected.

**2.1.4.443.2 Arguments**

**2.1.4.444.3.1 Input**

AxisID	Description Data type Range Unit Default	ID Name of the Axis block DINT — n/a —
PipePosition	Description Data type Range Unit Default	Sets the Axis Pipe Position LREAL — <a href="#">User unit</a> —

**2.1.4.445.4.2 Output**

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

**2.1.4.446.5 Related Functions**

[MLAxisReadActPos](#)

[MLAxisFBackPos](#)

[MLAxisGenPos](#)

[MLAxisPipePos](#)

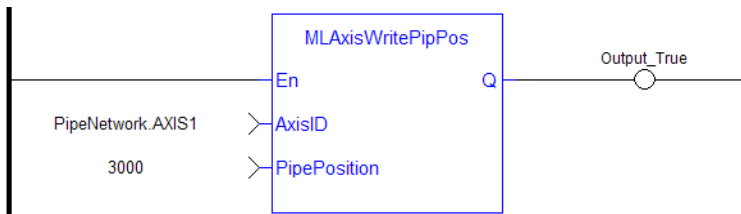
[MLAxisCmdPos](#)

**2.1.4.447.6 Example**

**2.1.4.448.7.1 Structured Text**

```
MLAxisWritePipPos (PipeNetwork.Axis1, 3000 ) ;
```

**2.1.4.449.8.2 Ladder Diagram**



**2.1.4.450.9.3 Function Block Diagram**



### 2.1.4.451 MLAxisWritePos

#### 2.1.4.452.1 Description

Used to set a position offset at the Axis when the Pipe Network:

- Pipe Position and Pipe Offset are set to zero
- Generator Position is set to equal to Zero Position
- Then Reference Position equals Pipe Position + Generator Position

#### About associated data on Positions

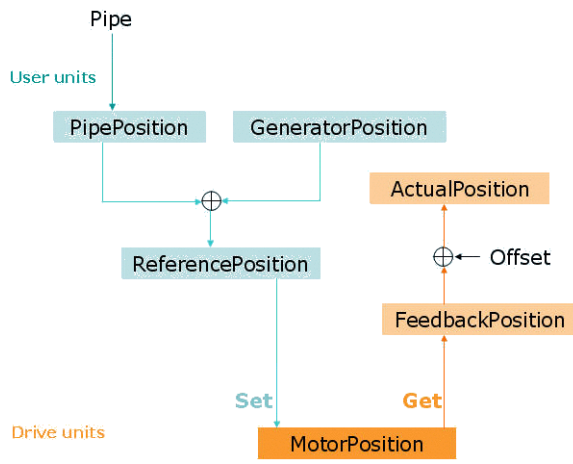
The following data are illustrated in the figure below.

##### NOTE

All positions are in user units with Modulo applied if active, unless specified.

Position / Offset	Description
<b>ActualPosition</b>	<p><b>Actual</b> refers to the actual position of the underlying Drive. It is the current position of the drive in user units. It is the sum of the feedback value (Position actual value) returned from the communication link to the drive, the <b>Power ON Delta Offset</b>, and any zero-offset due to an MLWritePos function ("<a href="#">MLAxisWritePipPos</a>" (p. 202), "<a href="#">MLAxisWritePos</a>" (p. 204)). Normally the value of power on delta offset is zero.</p> <pre>ActualPos := FeedbackPos + ZeroOffset</pre>
<b>CurrentPosition</b>	<p><b>Current</b> position is the actual command value being sent to the drive. It is an unsigned 32-bit integer value (fraction = zero). When in the power on condition this value is the command value that represents the target value in the communication link (Position demand value). It is not in user units, but in Drive units of 2**20 units per revolution of the drive.</p> <pre>CurrentPos := ReferencePosition + ZeroOffset</pre>
<b>FeedbackPosition</b>	<p><b>Feedback</b> Position is the "Position actual value" read from the drive. FeedbackPos relates to the TxPDO value of 'Actual position value'</p>
<b>GeneratorPosition</b>	<p><b>Generator</b> position is the summation of all previous commands to the Axis internal trapezoidal motion generator. It is also a collector of uncompensated motion due to "<a href="#">MLAxisWritePos</a>" (p. 204)() being used to modify actual position via the zero offset value and the adjustment in commanded value to insure no steps in the <b>Current</b> position command. It also accumulates changes in pipe position due to activate and deactivation of the pipe and convertor output to pipe position of the axis.</p>
<b>MotorPosition</b>	<p><b>Motor</b> position relates to the RxPDO value of 'Position demand value'</p> <pre>MotorPosition = CurrentPos + PowerOnDeltaOffset</pre>
<b>PipePosition</b>	<p>The output of the convertor block is written into the <b>PipePosition</b> value whenever the convertor block is connected to the axis and the pipe is active.</p>

Position / Offset	Description
<b>Power ON Delta Offset</b>	A change was made a long time ago to allow absolute feedback to be passed into the axis rather than always starting at zero actual position. Units are in Drive units of 2**20 units per revolution. On Drive Power On this value is set to be the difference between the “ <b>ActualPosition</b> value” and the “Position demand value” last sent to the drive. It is then added to the <b>Current</b> position value when the “Position demand value” is updated. It is read in User Units without periodicity applied.
<b>ReferencePosition</b>	Reference position is the summation of <b>PipePosition</b> and <b>GeneratorPosition</b> .  <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <math display="block">\text{ReferencePosition} = \text{Pipe Position} + \text{Generator Position}</math> </div>
<b>Zero Offset</b>	Affected by the “ <b>MLAxisWritePos</b> ” (p. 204)() function to adjust the actual position to the desired value of the command by setting zero offset to the difference between the desired and actual position, and applying the change to modify the generator position so that the reference position tracks the change in reference.



### 2.1.4.453.2 Arguments

#### 2.1.4.454.3.1 Input

ID	Description	ID Name of the Axis block
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—
Position	Description	Position offset.
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

#### 2.1.4.455.4.2 Output

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

#### 2.1.4.456.5 Previous Function Name

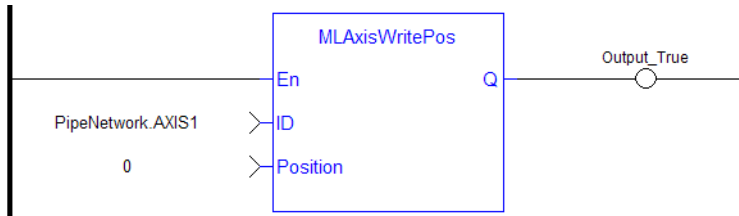
MLAxisSetZero

### 2.1.4.457.6 Example

#### 2.1.4.458.7.1 Structured Text

```
MLAxisWritePos ( PipeNetwork.Axis1, 0 ) ;
```

#### 2.1.4.459.8.2 Ladder Diagram



#### 2.1.4.460.9.3 Function Block Diagram



**2.1.4.461 MAxisWriteUUnits** Pipe Network ✓**2.1.4.462.1 Description**

Set the user units per revolution value of the axis. Returns TRUE if the function succeeded. User units are user-defined position units used within the KAS application. Selected units must be as natural as possible and must make sense for the machine. It must be related to the final moving object (e.g. the driven belt rather than the axis shaft). The same unit must be used for all related axes for simplicity reasons. Speeds are defined in [user units / second] and accelerations in [user units / second<sup>2</sup>].

**2.1.4.463.2 Arguments****2.1.4.464.3.1 Input**

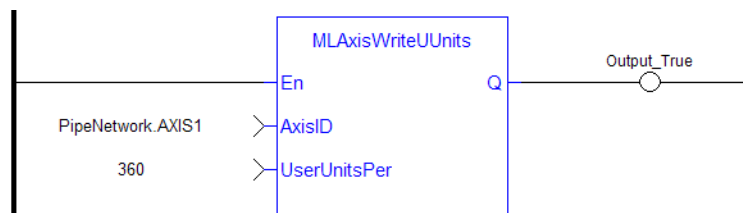
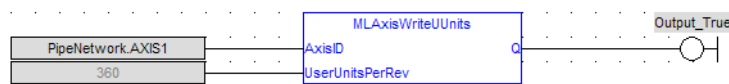
AxisID	Description	ID Name of the Axis block
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—

**2.1.4.465.4.2 Output**

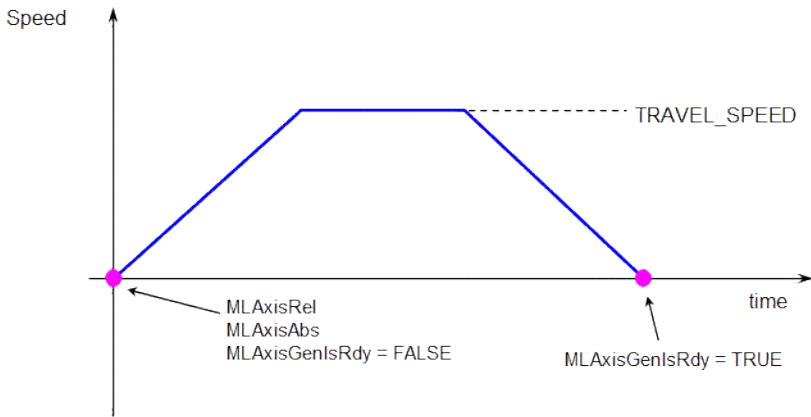
Default (.Q)	Description	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a
UserUnitsPerRev	Description	Sets the Axis User Units per revolution
	Data type	LREAL
	Unit	n/a

**2.1.4.466.5 Example****2.1.4.467.6.1 Structured Text**

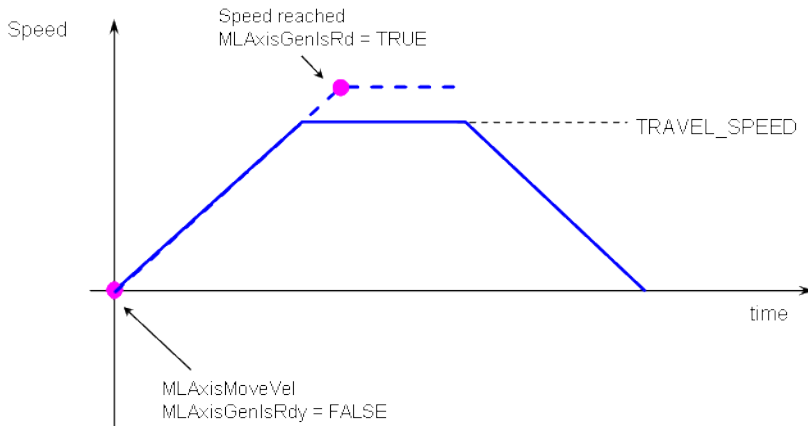
```
MAxisWriteUUnits (PipeNetwork.Axis1, 360 ) ;
```

**2.1.4.468.7.2 Ladder Diagram****2.1.4.469.8.3 Function Block Diagram**

### 2.1.4.470 Usage Example of Axis Functions



**MLAxisMoveVel(Speed)** starts to run the axis. Then **MLAxisGensRdy** returns TRUE when the Speed is reached.



**MLAxisMoveVel(0.0)** reduces the speed down to 0. Then **MLAxisGensRdy** returns TRUE once the axis is ready.

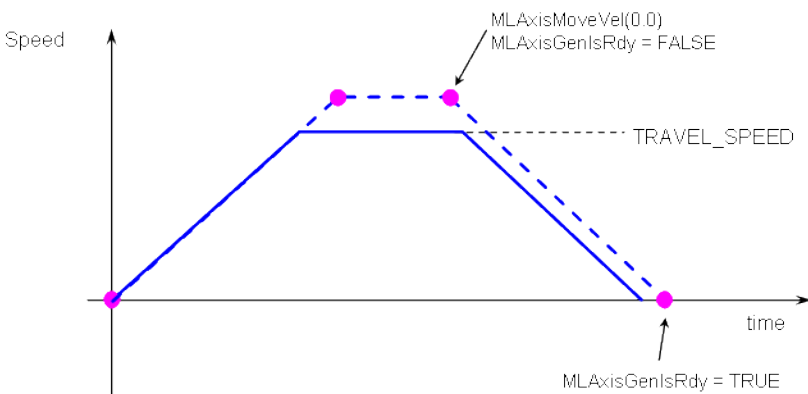


Figure 1-19: Axis Functions Usage



### 2.1.5 Motion Library - Cam Profile

Name	Description	Return type
"MLCamInit" (p. 210)	Initializes a cam Pipe Block with user-defined settings	BOOL
"MLCamSwitch" (p. 212)	Switches profiles of the selected cam object	BOOL
"MLPrfReadIOffset" (p. 214)	Returns the Input Offset value of a selected cam profile	None
"MLPrfReadIScale" (p. 216)	Returns the Input Ratio value of a selected cam profile	None
"MLPrfReadOOffset" (p. 218)	Returns the Output Offset value of a selected cam profile	None
"MLPrfReadOScale" (p. 220)	Returns the Output Ratio value of a selected cam profile	None
"MLPrfWriteIOffset" (p. 222)	Sets the Input Offset value of a selected cam profile	BOOL
"MLPrfWriteIScale" (p. 224)	Sets the Input Ratio value of a selected cam profile	BOOL
"MLPrfWriteOOffset" (p. 226)	Sets the Output Offset value of a selected cam profile	BOOL
"MLPrfWriteOScale" (p. 228)	Sets the Output Ratio value of a selected cam profile	BOOL
"MLProfileBuild" (p. 552)	Builds a cam profile from application data	See "Output" (p. 555)
"MLProfileCreate" (p. 561)	Creates a new cam profile object	None
"MLProfileInit" (p. 563)	Initializes a previously created cam profile object	BOOL
"MLProfileRelease" (p. 566)	Removes a Profile so the Profile ID may be used by a different or new Profile.	See "Output" (p. 567)

### 2.1.5.1 MLCamInit

#### 2.1.5.2.1 Description

Initializes a Cam Pipe Block for use in a PLC Program. Function block is automatically called if a Cam Block is added to the Pipe Network, with user-defined settings then entered in the Pipe Blocks Properties screen.

The Cam Pipe Block is used to generate motion profiles of any shape. These profiles are created and initiated separately and the shape is modified with the Cam Editor. With the Editor profiles can be changed graphically or by manually changing values in a numeric table relating input and output values with specific slopes. The Cam Editor software tool provides the capability to visualize, analyze, edit, and smooth profiles.

With the PipeNetwork (PN) Cam block:

- the Cam block's profile is in reference to the input positions coming into the PN Cam block (Master Absolute)
- the PN Cam block output positions are in reference to PN Cam block's output position at the end of the last cam cycle (Slave Relative)

Profile switching can be done on the fly, without losing synchronization and without dead time. In addition, the offsets and ratios of Cam Profiles can be changed on the fly. See [Cam Profile Switching](#) for more information.

#### NOTE

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

#### 2.1.5.3.2 Arguments

##### 2.1.5.4.3.1 Input

<b>BlockID</b>	<b>Description</b>	ID number of a created Pipe Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	CAM
<b>ProfileName</b>	<b>Description</b>	Name of the current profile assigned to the cam. It must be a declared profile object
	<b>Data type</b>	STRING
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>ModuloPosition</b>	<b>Description</b>	Value of the period of the cam output values expressed in user units, for a cyclic system
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	360.0

**2.1.5.5.4.2 Output**

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

**2.1.5.6.5.3 Return Type**

BOOL

**2.1.5.7.6 Related Functions**

"MLProfileCreate" (p. 561)

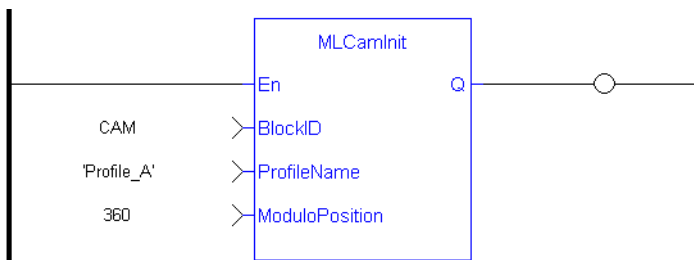
"MLProfileInit" (p. 563)

**2.1.5.8.7 Example**

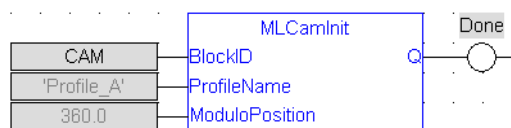
**2.1.5.9.8.1 Structured Text**

```
//Initialize a Pipe Network block named "CAM" with a profile
named "Profile_A", set the cam modulo position to 360
CAM := MLBlkCreate( 'CAM', 'CAM' );
MLCamInit( CAM, 'Profile_A', 360.0 );
```

**2.1.5.10.9.2 Ladder Diagram**



**2.1.5.11.10.3 Function Block Diagram**



### 2.1.5.12 MLCamSwitch

#### 2.1.5.13.1 Description

Switches the CAM Profile in a selected CAM object. Can be used in combination with a comparator to check that profiles are switched at a time where the input and output values of both the old and new profiles are equal, so an Axis receives continuous position values and does not jump.

These profiles are created and initiated separately and the shape is created with the CAM Editor. With the Editor profiles can be changed graphically or by manually changing values in a numeric table relating input and output values with specific slopes. The Cam Editor software tool provides the capability to visualize, analyze, edit, and smooth profiles.

See [Cam Profile Switching](#) for more information.

#### 2.1.5.14.2 Arguments

##### 2.1.5.15.3.1 Input

<b>BlockID</b>	<b>Description</b>	ID number of an initialized CAM Pipe Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ProfileID</b>	<b>Description</b>	Name of the new CAM profile which is assigned to the CAM Pipe Block. It must be a declared profile object.
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—

##### 2.1.5.16.4.2 Output

<b>Default (.Q)</b>	<b>Description</b>	Returns TRUE if the CAM Profile is changed
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

##### 2.1.5.17.5.3 Return Type

BOOL

##### 2.1.5.18.6 Related Functions

["MLProfileCreate"](#) (p. 561)

["MLProfileInit"](#) (p. 563)

["MLPrfWriteOffset"](#) (p. 222)

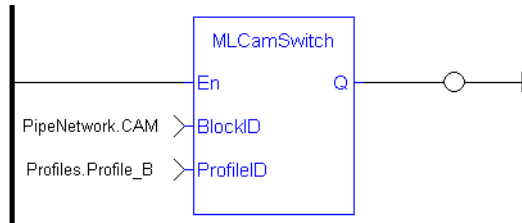
["MLPrfWriteOScale"](#) (p. 228)

##### 2.1.5.19.7 Example

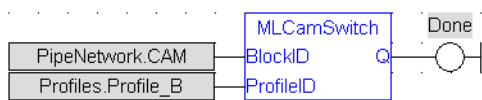
### 2.1.5.20.8.1 Structured Text

```
//Switch CAM Profile  
MLCamSwitch(PipeNetwork.CAM, Profiles.Profile_B);
```

### 2.1.5.21.9.2 Ladder Diagram



### 2.1.5.22.10.3 Function Block Diagram



**2.1.5.23 MLPrfReadIOffset** Pipe Network ✓

**2.1.5.24.1 Description**

Returns the Input Offset value of a selected CAM Profile. Offsets can be changed on the fly to modify the CAM Profile while maintaining its shape. A change in input offset is equivalent to shifting the CAM Profile on the x or Input Axis.

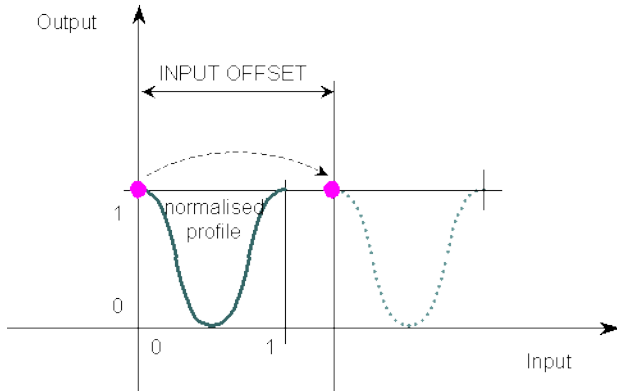


Figure 1-20: MLPrfReadIOffset

**2.1.5.25.2 Arguments**

**2.1.5.26.3.1 Input**

ProfileID	Description Data type Range Unit Default	Name of an initialized CAM Profile DINT [-2147483648, 2147483648] n/a —
-----------	--	---

**2.1.5.27.4.2 Output**

OK	Description Data type	Returns true when function successfully executes BOOL
Offset	Description Data type Unit	Returns the Input Offset of the selected CAM Profile LREAL n/a

**2.1.5.28.5 Related Functions**

["MLPrfWriteIOffset" \(p. 222\)](#)

["MLProfileCreate" \(p. 561\)](#)

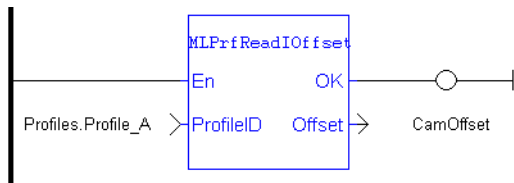
["MLProfileInit" \(p. 563\)](#)

**2.1.5.29.6 Example**

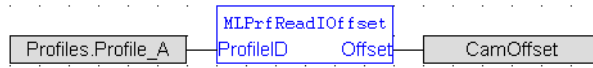
**2.1.5.30.7.1 Structured Text**

```
//Save value of input offset
CamOffset := MLPrfReadIOffset( Profiles.Profile_A );
```

**2.1.5.31.8.2 Ladder Diagram**



### 2.1.5.32.9.3 Function Block Diagram



### 2.1.5.33 MLPrfReadIScale Pipe Network ✓

#### 2.1.5.34.1 Description

Returns the Input Ratio value of a selected CAM Profile. Ratios can be changed on the fly to modify the CAM Profile while maintaining its basic shape. A change in input ratio is equivalent to stretching the CAM Profile on the X (or Input) Axis. A negative value is not allowed.

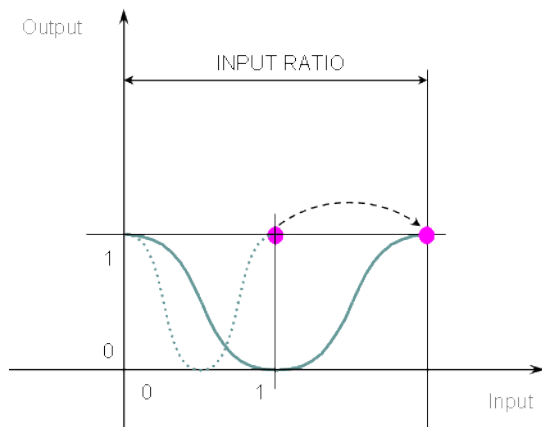


Figure 1-21: MLPrfReadIScale

#### 2.1.5.35.2 Arguments

##### 2.1.5.36.3.1 Input

	Description	ID number of an initialized CAM Profile
	Data type	DINT
ProfileID	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

##### 2.1.5.37.4.2 Output

	Description	Returns the Input Ratio of the selected CAM Profile
Ratio	Data type	LREAL
	Unit	n/a

#### 2.1.5.38.5 Related Functions

["MLPrfWriteIScale" \(p. 224\)](#)

["MLProfileCreate" \(p. 561\)](#)

["MLProfileInit" \(p. 563\)](#)

#### 2.1.5.39.6 Previous Function Name

MLPrfGetIRatio

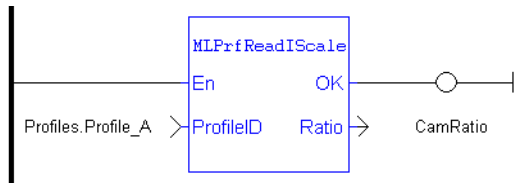
#### 2.1.5.40.7 Example

##### 2.1.5.41.8.1 Structured Text

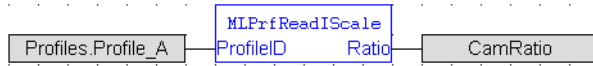
```
//Save value of input ratio
CamRatio := MLPrfReadIScale( Profiles.Profile_A );
```



### 2.1.5.42.9.2 Ladder Diagram



### 2.1.5.43.10.3 Function Block Diagram



**2.1.5.44 MLPrfReadOOffset** Pipe Network ✓**2.1.5.45.1 Description**

Returns the Output Offset value of a selected CAM Profile. Offsets can be changed on the fly to modify the CAM Profile while maintaining its shape. A change in output offset is equivalent to shifting the CAM Profile on the Y (or Output) Axis.

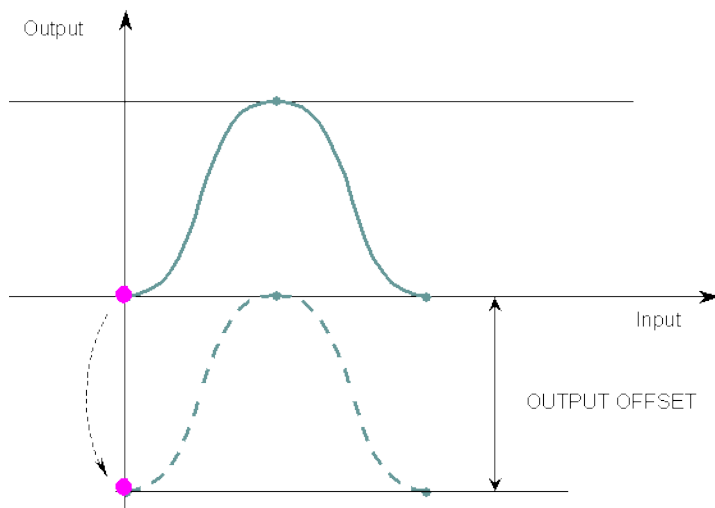


Figure 1-22: MLPrfReadOOffset

**2.1.5.46.2 Arguments****2.1.5.47.3.1 Input**

ProfileID	Description	ID number of an initialized CAM Profile
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

**2.1.5.48.4.2 Output**

Offset	Description	Returns the Output Offset of the selected CAM Profile
	Data type	LREAL
	Unit	n/a

**2.1.5.49.5 Related Functions**

["MLPrfWriteOOffset" \(p. 226\)](#)

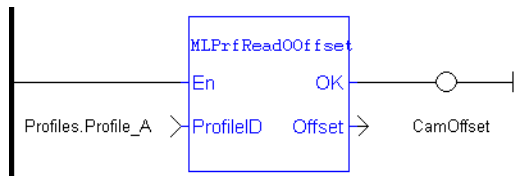
["MLProfileCreate" \(p. 561\)](#)

["MLProfileInit" \(p. 563\)](#)

**2.1.5.50.6 Example****2.1.5.51.7.1 Structured Text**

```
//Save value of output offset
CamOffset := MLPrfReadOOffset( Profiles.Profile_A );
```

**2.1.5.52.8.2 Ladder Diagram**



### 2.1.5.53.9.3 Function Block Diagram



**2.1.5.54 MLPrfReadOScale** Pipe Network ✓

**2.1.5.55.1 Description**

Returns the Output Ratio value of a selected CAM Profile. Ratios can be changed on the fly to modify the CAM Profile while maintaining its basic shape. A change in output ratio is equivalent to stretching, and flipping if negative, the CAM Profile on the Y (or Output) Axis.

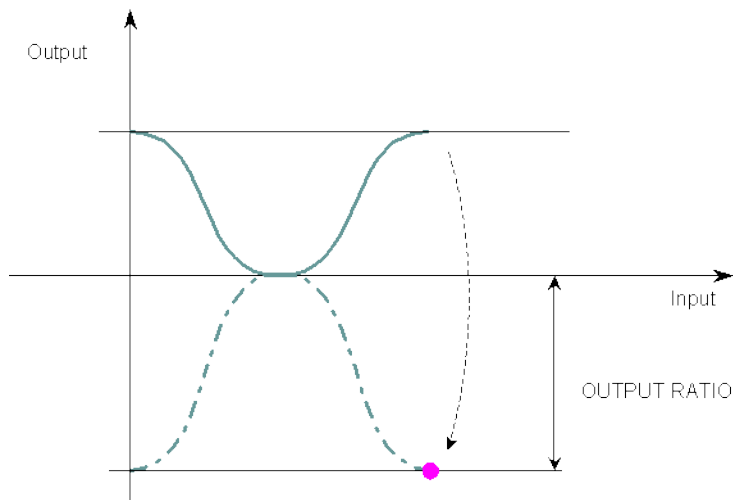


Figure 1-23: MLPrfReadOScale

**2.1.5.56.2 Arguments**

**2.1.5.57.3.1 Input**

ProfileID	Description Data type Range Unit Default	ID number of an initialized CAM Profile DINT [-2147483648, 2147483648] n/a —
-----------	--	--

**2.1.5.58.4.2 Output**

Ratio	Description Data type Unit	Returns the Output Ratio of the selected CAM Profile LREAL n/a
-------	----------------------------------	--

**2.1.5.59.5 Related Functions**

["MLPrfWriteOScale" \(p. 228\)](#)

["MLProfileCreate" \(p. 561\)](#)

["MLProfileInit" \(p. 563\)](#)

**2.1.5.60.6 Previous Function Name**

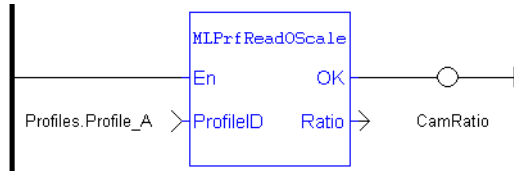
MLPrfGetORatio

**2.1.5.61.7 Example**

**2.1.5.62.8.1 Structured Text**

```
//Save value of output ratio  
CamRatio := MLPrfReadOScale( Profiles.Profile_A );
```

### 2.1.5.63.9.2 Ladder Diagram



### 2.1.5.64.10.3 Function Block Diagram



**2.1.5.65 MLPrfWriteOffset** Pipe Network ✓

**2.1.5.66.1 Description**

Set the Input Offset value of a selected CAM Profile. Offsets are changed on the fly to modify the CAM Profile while maintaining its shape. A change in input offset is equivalent to shifting the CAM Profile on the X (or Input) Axis.

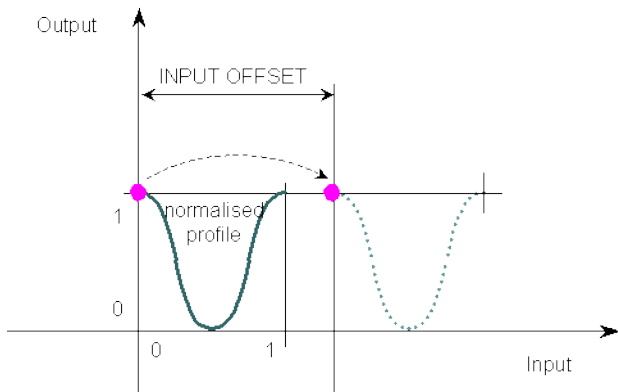


Figure 1-24: MLPrfWriteOffset

**2.1.5.67.2 Arguments**

**2.1.5.68.3.1 Input**

ProfileID	Description Data type Range Unit Default	ID number of an initialized CAM Profile DINT [-2147483648, 2147483648] n/a —
Offset	Description Data type Range Unit Default	Desired new value of Input Offset LREAL — n/a —

**2.1.5.69.4.2 Output**

Default (.Q)	Description Data type Unit	Returns TRUE if the Input Offset is changed to the new value BOOL n/a
--------------	----------------------------------	---

**2.1.5.70.5.3 Return Type**

BOOL

**2.1.5.71.6 Related Functions**

["MLPrfReadIOffset" \(p. 214\)](#)

["MLProfileCreate" \(p. 561\)](#)

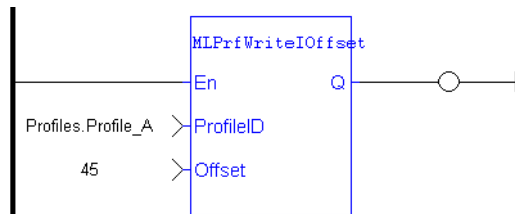
["MLProfileInit" \(p. 563\)](#)

**2.1.5.72.7 Example**

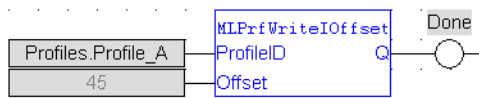
**2.1.5.73.8.1 Structured Text**

```
//Change the value of input offset  
MLPrfWriteIOffset( Profiles.Profile_A , 45 );
```

### 2.1.5.74.9.2 Ladder Diagram



### 2.1.5.75.10.3 Function Block Diagram



**2.1.5.76 MLPrfWriteIScale** Pipe Network ✓

**2.1.5.77.1 Description**

Set the Input Ratio value of a selected CAM Profile. Ratios are changed on the fly to modify the CAM Profile while maintaining its basic shape. A change in input ratio is equivalent to stretching the CAM Profile on the X (or Input) Axis.

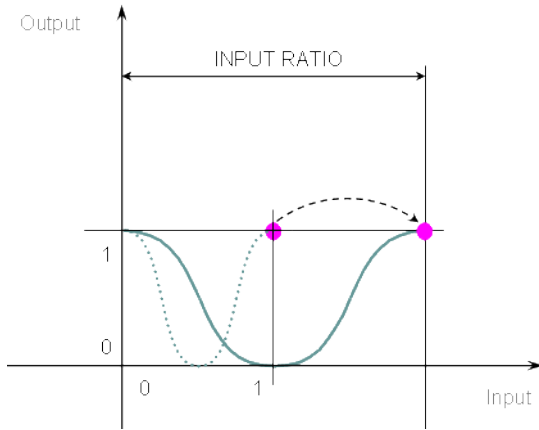


Figure 1-25: MLPrfWriteIScale

**2.1.5.78.2 Arguments**

**2.1.5.79.3.1 Input**

ProfileID	Description	ID number of initialized CAM Profile
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
Ratio	Description	Desired new value for Input Ratio
	Data type	LREAL
	Range	Positive
	Unit	n/a
	Default	—

**2.1.5.80.4.2 Output**

Default (.Q)	Description	Returns TRUE if the Input Ratio is changed
	Data type	BOOL
	Unit	n/a

**2.1.5.81.5.3 Return Type**

BOOL

**2.1.5.82.6 Related Functions**

["MLPrfReadIScale" \(p. 216\)](#)

["MLProfileCreate" \(p. 561\)](#)

["MLProfileInit" \(p. 563\)](#)

**2.1.5.83.7 Previous Function Name**

MLPrfSetIRatio

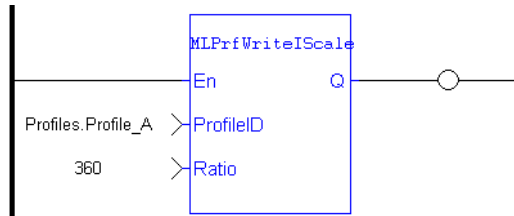
**2.1.5.84.8 Example**



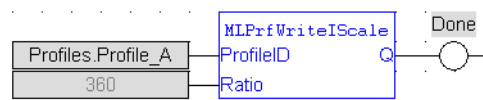
### 2.1.5.85.9.1 Structured Text

```
//Change value of input ratio
MLPrfWriteIScale( Profiles.Profile_A, 360 );
```

### 2.1.5.86.10.2 Ladder Diagram



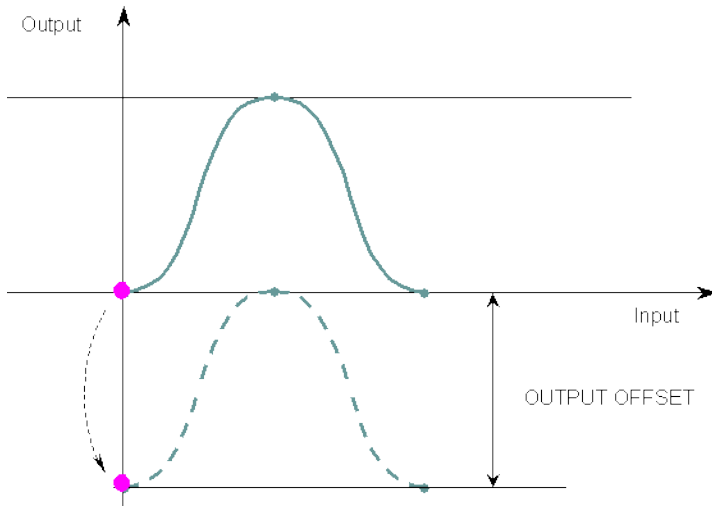
### 2.1.5.87.11.3 Function Block Diagram



**2.1.5.88 MLPrfWriteOOffset** Pipe Network ✓

**2.1.5.89.1 Description**

Changes the Output Offset value of a selected CAM Profile. Offsets are changed on the fly to modify the CAM Profile while maintaining its shape. A change in output offset is equivalent to shifting the CAM Profile on the Y (or Output) Axis.



**Figure 1-26:** MLPrfWriteOOffset

**2.1.5.90.2 Arguments**

**2.1.5.91.3.1 Input**

ProfileID	Description Data type Range Unit Default	ID number of an initialized CAM Profile DINT [-2147483648, 2147483648] n/a —
Offset	Description Data type Range Unit Default	Desired new value of Output Offset LREAL — n/a —

**2.1.5.92.4.2 Output**

Default (.Q)	Description Data type Unit	Returns TRUE if the Output Offset value is changed BOOL n/a
--------------	----------------------------------	---

**2.1.5.93.5.3 Return Type**

BOOL

**2.1.5.94.6 Related Functions**

["MLPrfReadOOffset" \(p. 218\)](#)

["MLProfileCreate" \(p. 561\)](#)

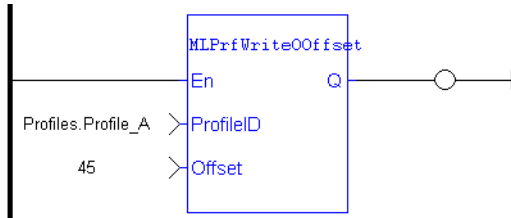
["MLProfileInit" \(p. 563\)](#)

**2.1.5.95.7 Example**

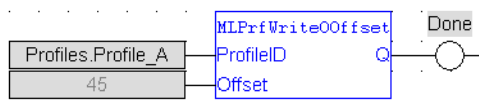
**2.1.5.96.8.1 Structured Text**

```
//Change value of output offset  
MLPrfWriteOOffset( Profiles.Profile_A , 45 );
```

### 2.1.5.97.9.2 Ladder Diagram



### 2.1.5.98.10.3 Function Block Diagram



**2.1.5.99 MLPrfWriteOScale** Pipe Network ✓**2.1.5.100.1 Description**

Set the Output Ratio value of a selected CAM Profile. Ratios are changed on the fly to modify the CAM Profile while maintaining its basic shape. A change in output ratio is equivalent to stretching, and flipping if negative (as shown on figure below), the CAM Profile on the Y (or Output) Axis.

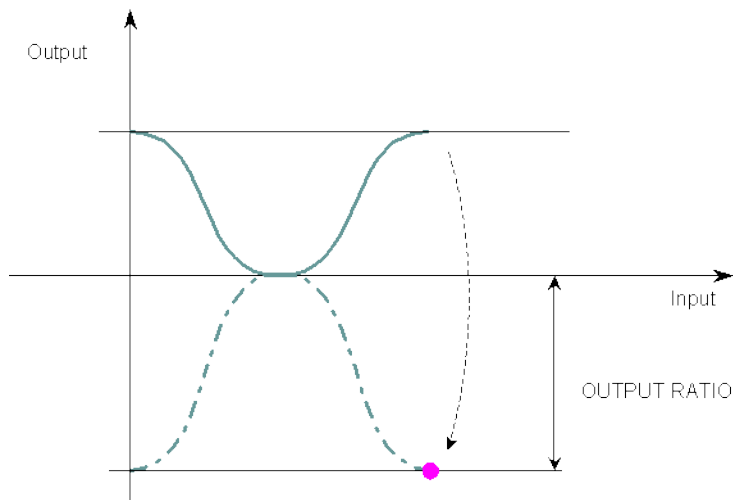


Figure 1-27: MLPrfWriteOScale

**2.1.5.101.2 Arguments****2.1.5.102.3.1 Input**

ProfileID	Description	ID number of an initialized CAM Profile
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
Ratio	Description	Desired new value of Output Ratio
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—

**2.1.5.103.4.2 Output**

Default (.Q)	Description	Returns TRUE if the Output Ratio is changed
	Data type	BOOL
	Unit	n/a

**2.1.5.104.5.3 Return Type**

BOOL

**2.1.5.105.6 Related Functions**

["MLPrfReadOScale" \(p. 220\)](#)

["MLProfileCreate" \(p. 561\)](#)

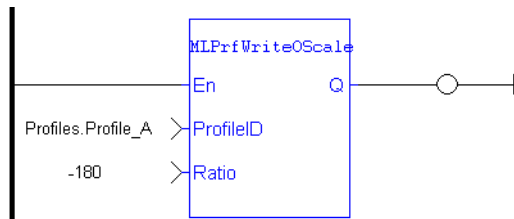
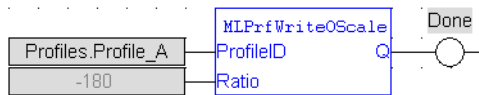
["MLProfileInit" \(p. 563\)](#)

**2.1.5.106.7 Previous Function Name**

MLPrfSetORatio

**2.1.5.107.8 Example****2.1.5.108.9.1 Structured Text**

```
//Change value of output ratio
MLPrfWriteOScale( Profiles.Profile_A , -180 );
```

**2.1.5.109.10.2 Ladder Diagram****2.1.5.110.11.3 Function Block Diagram**

## 2.1.6 Motion Library - Comparator

**TIP**

- For a Comparator function example, see "Usage example of Comparator Functions" (p. 240)

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

### 2.1.6.1 MLCompCheck Pipe Network ✓

#### 2.1.6.2.1 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.

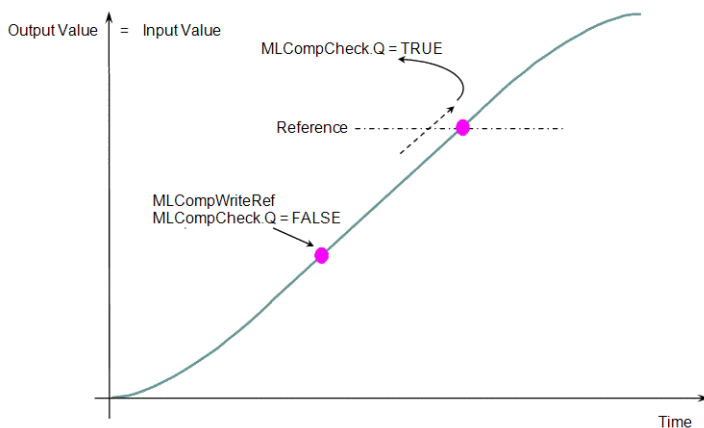


Figure 1-28: MLCompCheck

#### 2.1.6.3.2 Arguments

##### 2.1.6.4.3.1 Input

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

##### 2.1.6.5.4.2 Output

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

### 2.1.6.6.5.3 Return Type

BOOL

### 2.1.6.7.6 Related Functions

[MLCompReset](#)

[MLCompWriteRef](#)

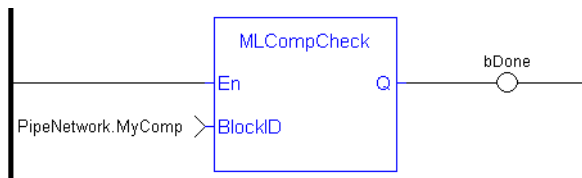
[MLCompReadRef](#)

### 2.1.6.8.7 Example

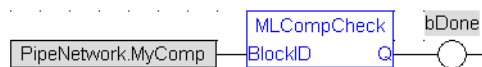
#### 2.1.6.9.8.1 Structured Text

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

### 2.1.6.10.9.2 Ladder Diagram



### 2.1.6.11.10.3 Function Block Diagram



### 2.1.6.12 MLComplnit

#### 2.1.6.13.1 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.

#### 2.1.6.14.2 Arguments

##### 2.1.6.15.3.1 Input

<b>BlockID</b>	<b>Description</b>	ID number of a created Comparator Pipe Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ModuloPosition</b>	<b>Description</b>	Value of the period of a cyclic system
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>ThroughZero</b>	<b>Description</b>	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.
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Reference</b>	<b>Description</b>	Set the reference position in the new Comparator object



<b>Data type</b>	LREAL
<b>Range</b>	—
<b>Unit</b>	User unit
<b>Default</b>	—

#### 2.1.6.16.4.2 Output

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

#### 2.1.6.17.5.3 Return Type

BOOL

#### 2.1.6.18.6 Related Functions

[MLBlkCreate](#)

[MLCompCheck](#)

[MLCompReset](#)

[MLCompWriteRef](#)

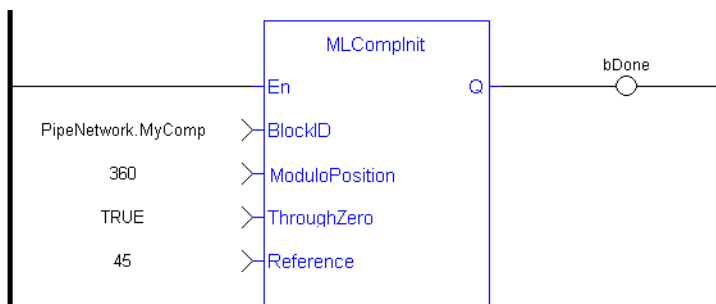
#### 2.1.6.19.7 Example

##### 2.1.6.20.8.1 Structured Text

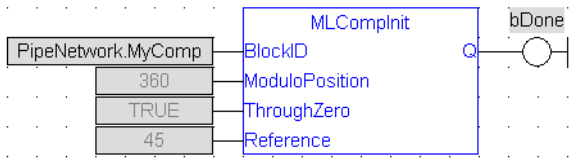
```
//Initiate a created Comparator Block named "MyComp" to:
// Modulo of 360
// Require the input position to first cross 0 before the
// MLCompCheck output is triggered
// Input compared position to 45

MyComp := MLBlkCreate( 'MyComp', 'COMPARATOR' );
MLCompInit( MyComp, 360.0, TRUE, 45.0 );
```

##### 2.1.6.21.9.2 Ladder Diagram



##### 2.1.6.22.10.3 Function Block Diagram



**2.1.6.23 MLCompReadRef** Pipe Network ✓

**2.1.6.24.1 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.

**2.1.6.25.2 Arguments**

**2.1.6.26.3.1 Input**

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

**2.1.6.27.4.2 Output**

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

**2.1.6.28.5 Related Functions**

[MLCompWriteRef](#)

[MLCompReset](#)

[MLCompCheck](#)

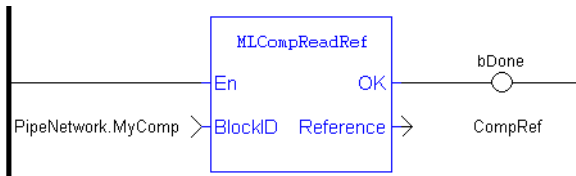
**2.1.6.29.6 Example**

**2.1.6.30.7.1 Structured Text**

```

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

**2.1.6.31.8.2 Ladder Diagram**



**2.1.6.32.9.3 Function Block Diagram**



**2.1.6.33 MLCompReset** Pipe Network ✓

**2.1.6.34.1 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.

**2.1.6.35.2 Arguments**

**2.1.6.36.3.1 Input**

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

**2.1.6.37.4.2 Output**

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

**2.1.6.38.5.3 Return Type**

BOOL

**2.1.6.39.6 Related Functions**

[MLCompCheck](#)

[MLCompReadRef](#)

[MLCompWriteRef](#)

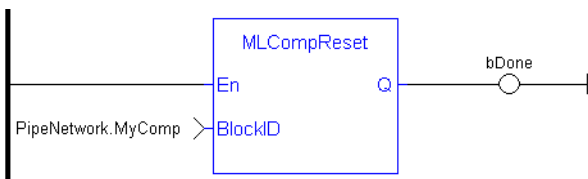
**2.1.6.40.7 Example**

**2.1.6.41.8.1 Structured Text**

```

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

**2.1.6.42.9.2 Ladder Diagram**



**2.1.6.43.10.3 Function Block Diagram**



**2.1.6.44 MLCompWriteRef** Pipe Network ✓

**2.1.6.45.1 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.

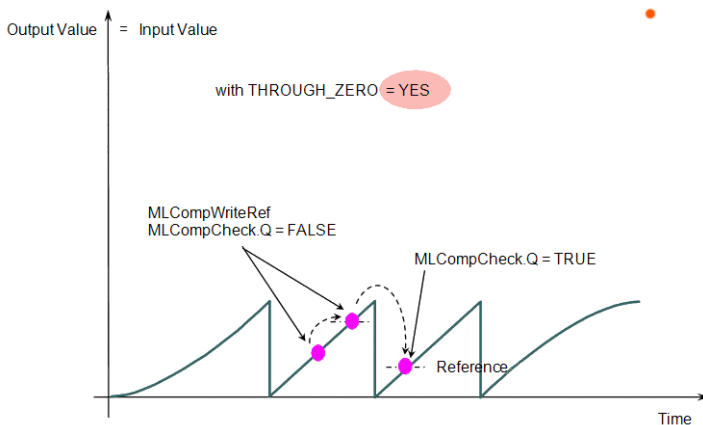


Figure 1-29: MLCompWriteRef

**2.1.6.46.2 Arguments**

**2.1.6.47.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID number of an initiated Comparator object
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ThroughZero</b>	<b>Description</b>	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.
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Reference</b>	<b>Description</b>	New reference position to set in the selected Comparator object
	<b>Data type</b>	LREAL

<b>Range</b>	—
<b>Unit</b>	User unit
<b>Default</b>	—

### 2.1.6.48.4.2 Output

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

### 2.1.6.49.5.3 Return Type

BOOL

### 2.1.6.50.6 Related Functions

[MLCompCheck](#)

[MLCompReadRef](#)

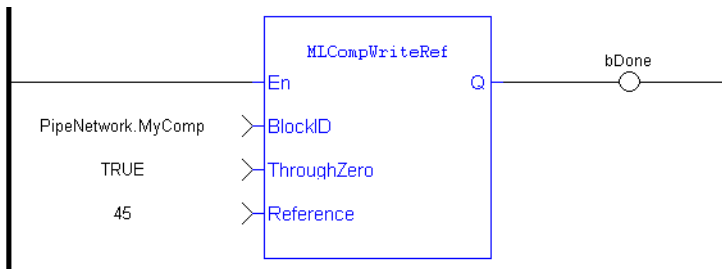
[MLCompReset](#)

### 2.1.6.51.7 Example

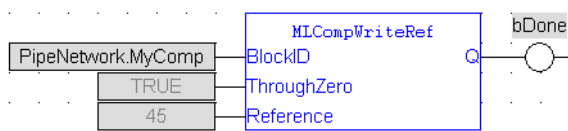
#### 2.1.6.52.8.1 Structured Text

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

### 2.1.6.53.9.2 Ladder Diagram

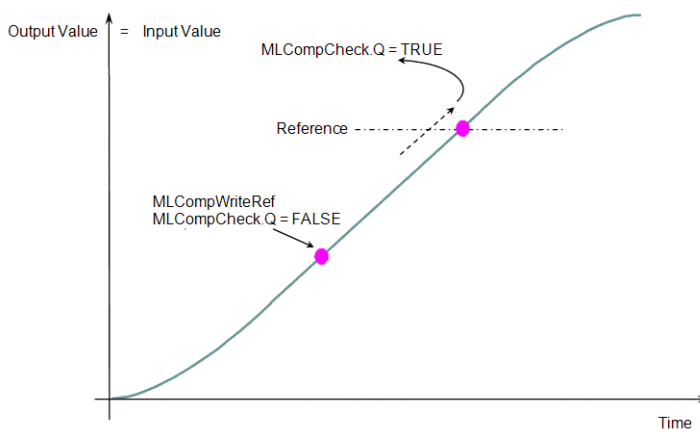
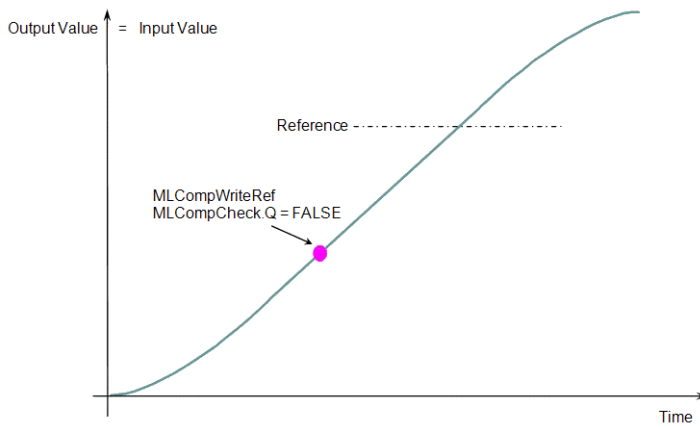


### 2.1.6.54.10.3 Function Block Diagram

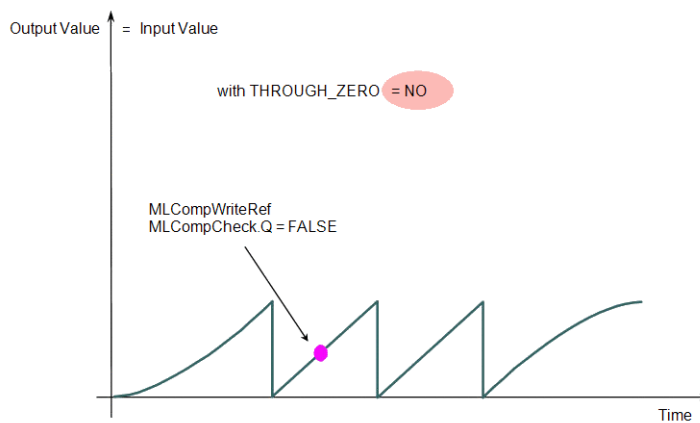


### 2.1.6.55 Usage example of Comparator Functions

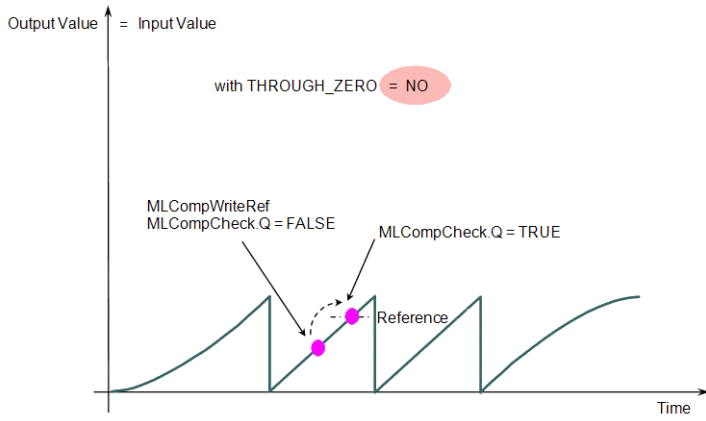
When you call the **MLCompWriteRef** function, the output for **MLCompCheck** becomes True as soon as the input value reaches the reference.



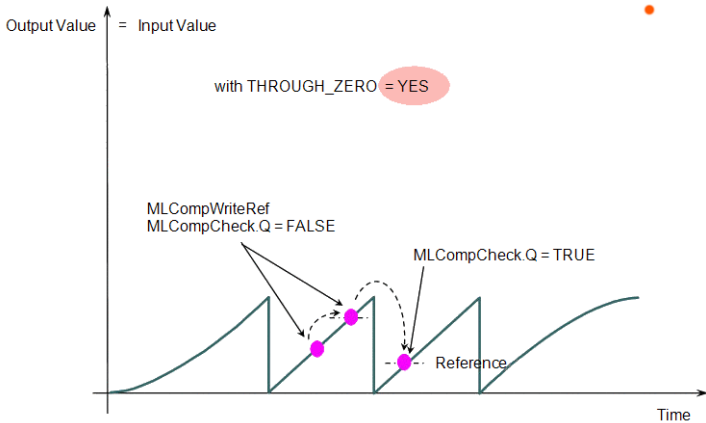
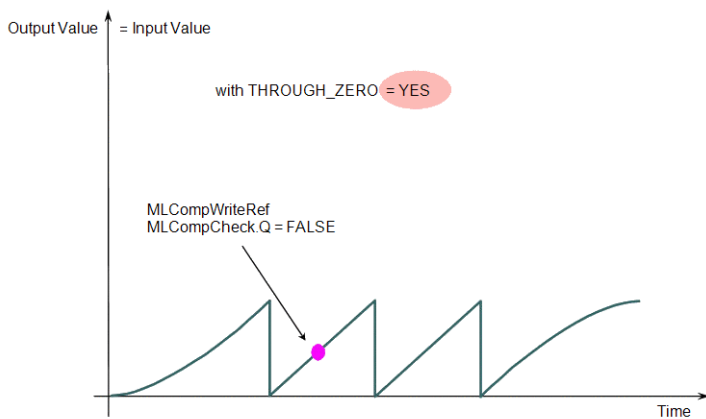
The same function can also be called for a cyclic input value.







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 1-30:** Comparator Functions Usage

### 2.1.7 Motion Library - Convertor

Name	Description	Return type
"MLCNVConnect" (p. 242)	Connects a converter Pipe Block to the specified axis	BOOL
"MLCNVConnectEx" (p. 244)	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" (p. 249)	Disconnects a converter Pipe Block from its associated axis	BOOL
"MLCNVInit" (p. 251)	Initializes a converter Pipe Block in Position or Speed mode	BOOL

#### 2.1.7.1 MLCNVConnect

##### 2.1.7.2.1 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.

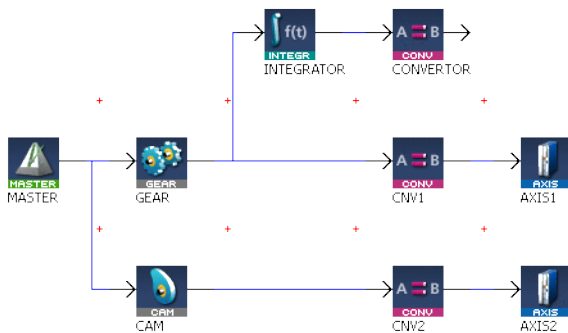


Figure 1-31: MLCNVConnect

#### 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.

#### 2.1.7.3.2 Arguments

##### 2.1.7.4.3.1 Input

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

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

#### 2.1.7.5.4.2 Output

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

#### 2.1.7.6.5.3 Return Type

BOOL

#### 2.1.7.7.6 Related Functions

["MLCNVConnectEx" \(p. 244\)](#)

["MLCNVDisconnect" \(p. 249\)](#)

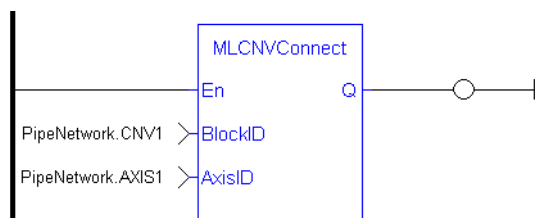
["MLCNVInit" \(p. 251\)](#)

#### 2.1.7.8.7 Example

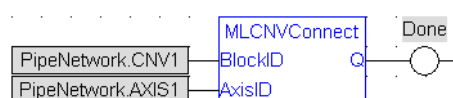
##### 2.1.7.9.8.1 Structured Text

```
//Connect a converter Pipe Block named "CNV1" to Pipe Block
//AXIS1
MLCNVConnect( PipeNetwork.CNV1, AXIS1 );
```

#### 2.1.7.10.9.2 Ladder Diagram



#### 2.1.7.11.10.3 Function Block Diagram



## 2.1.7.12 MLCNVConnectEx

### 2.1.7.13.1 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.

### 2.1.7.14.2 Arguments

#### 2.1.7.15.3.1 Input

<b>BlockID</b>	<b>Description</b>	ID number of an initiated Converter object
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxisID</b>	<b>Description</b>	ID number of an initiated Axis object
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ValueID</b>	<b>Description</b>	Specify the following constant: <ul style="list-style-type: none"> <li>• EC_ADDITIVE_TORQUE_VALUE (for torque feed-forward)</li> <li>• <a href="#">EC_ANALOG_OUTPUT</a> (for control of Analog Output: AKD parameter: "<a href="#">AOUT.VALUEU</a>")</li> </ul> <p>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 <a href="#">AOUT.MODE</a> must be set to "User Mode (mode = 0)". See the TIP below for more information.</p>
	<b>Data type</b>	DINT

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

### 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.

#### 2.1.7.16.4.2 Output

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

#### 2.1.7.17.5.3 Return Type

BOOL

#### 2.1.7.18.6 Related Functions

["MLCNVConnect" \(p. 242\)](#)

["MLCNVDisconnect" \(p. 249\)](#)

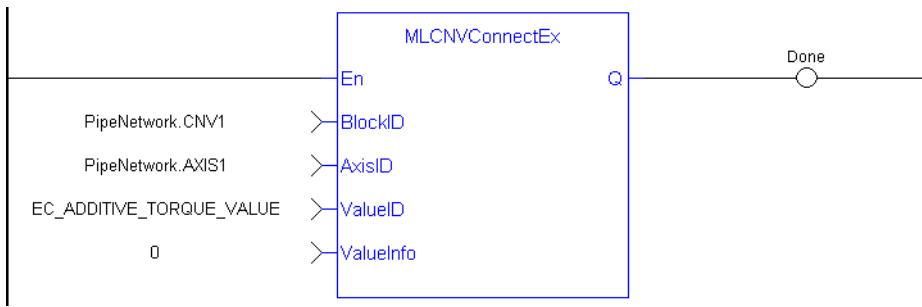
["MLCNVInit" \(p. 251\)](#)

#### 2.1.7.19.7 Example

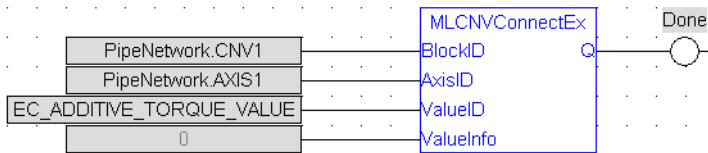
##### 2.1.7.20.8.1 Structured Text

```
//Connect a converter Pipe Block named "CNV1" to the pipe block
named AXIS1, And send feed-forward (EC_ADDITIVE_TORQUE_VALUE) to
the drive
MLCNVConnectEx( PipeNetwork.CNV1, PipeNetwork.AXIS1, EC_
ADDITIVE_TORQUE_VALUE, 0 );
```

#### 2.1.7.21.9.2 Ladder Diagram



### 2.1.7.22.10.3 Function Block Diagram



### 2.1.7.23 MLCNVConECAT

#### 2.1.7.24.1 Description

This function will connect the output of a pipe convertor block to an EtherCAT Output (Rx) PDO object. The output value of the convertor block will then be written to the PDO object every update of the convertor block. The pipe block is specified by the BlockID input and the PDO object is specified by the DeviceAddr, Index, and SubIndex inputs.

#### 2.1.7.25.2 Arguments

##### 2.1.7.26.3.1 Input

<b>BlockID</b>	<b>Description</b>	The convertor block whose output value will be written to the PDO object. For example: PipeNetwork:CNV1
	<b>Data type</b>	DINT
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>DeviceAddr</b>	<b>Description</b>	The device address of the PDO object to be written. EtherCAT devices are numbered in order with the first device being 1001, the second 1002, etc.
	<b>Data type</b>	INT
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Index</b>	<b>Description</b>	The index of the PDO object to be written. The index can be determined from the table located in the “PDO Selection/Mapping” tab of the EtherCAT device page. (In Project Explorer, under EtherCAT, select the device, then select the PDO Selection/Mapping tab.)
	<b>Data type</b>	UINT
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>SubIndex</b>	<b>Description</b>	The sub index of the PDO object to be written. The sub index can be determined from the table located in the “PDO Selection/Mapping” tab of the EtherCAT device page. (In Project Explorer, under EtherCAT, select the device, then select the PDO Selection/Mapping tab.)
	<b>Data type</b>	USINT
	<b>Range</b>	n/a

<b>Unit</b>	n/a
<b>Default</b>	—

**2.1.7.27.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns TRUE if this function has successfully connected the output of the pipe convertor block to the EtherCAT Output (Rx) PDO Object.
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.7.28.5 Related Functions**

"MLCNVDisconnect" (p. 249)

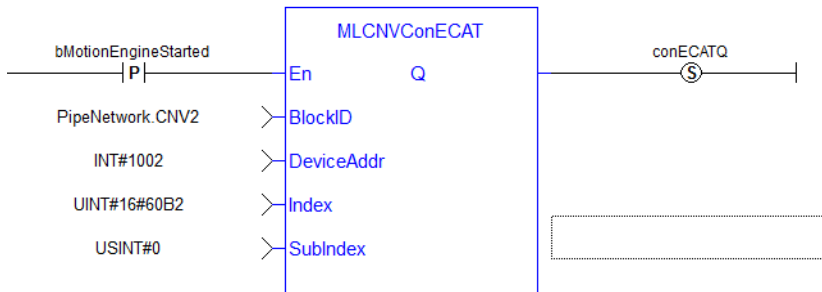
"MLCNVInit" (p. 251)

**2.1.7.29.6 Example**

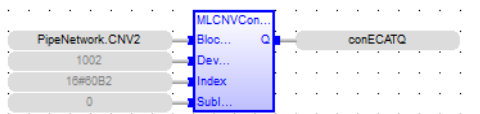
**2.1.7.30.7.1 Structured Text**

```
//Connect a converter Pipe Block named "CNV2" to PDO 16#60B2
(Accel FF) on ECAT address 1002.
MLCNVConECAT( PipeNetwork.CNV2, 1002, 16#60B2, 0 );
```

**2.1.7.31.8.2 Ladder Diagram**



**2.1.7.32.9.3 Function Block Diagram**





### 2.1.7.33 MLCNVDisconnect

#### 2.1.7.34.1 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.

#### 2.1.7.35.2 Arguments

##### 2.1.7.36.3.1 Input

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

##### 2.1.7.37.4.2 Output

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

##### 2.1.7.38.5.3 Return Type

BOOL

##### 2.1.7.39.6 Related Functions

["MLCNVConnect" \(p. 242\)](#)

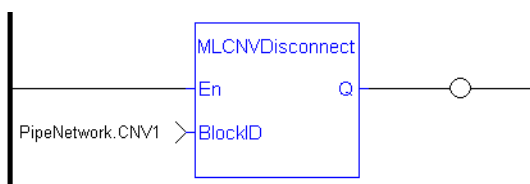
["MLCNVInit" \(p. 251\)](#)

##### 2.1.7.40.7 Example

##### 2.1.7.41.8.1 Structured Text

```
//Disconnect a converter Pipe Block name " CNV1" from its
present connection
MLCNVDisconnect( PipeNetwork.CNV1);
```

##### 2.1.7.42.9.2 Ladder Diagram



### 2.1.7.43.10.3 Function Block Diagram



## 2.1.7.44 MLCNVInit

### 2.1.7.45.1 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.

### 2.1.7.46.2 Arguments

#### 2.1.7.47.3.1 Input

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

#### 2.1.7.48.4.2 Output

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

#### 2.1.7.49.5.3 Return Type

BOOL

#### 2.1.7.50.6 Related Functions

[MLBlkCreate](#)

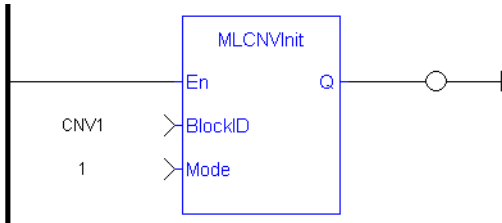
"MLCNVConnect" (p. 242)

#### 2.1.7.51.7 Example

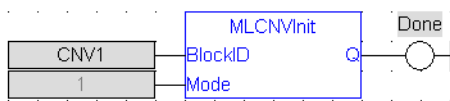
#### 2.1.7.52.8.1 Structured Text

```
// Initiate a created convertor block named "CNV1"
CNV1 := MLBlkCreate( 'CNV1', 'CONVERTOR' );
MLCNVInit( CNV1, 1 );
```

### 2.1.7.53.9.2 Ladder Diagram



### 2.1.7.54.10.3 Function Block Diagram



## 2.1.8 Motion Library - Delay

Name	Description	Return type
"MLDelayInit" (p. 253)	Initializes a delay object	BOOL

### 2.1.8.1 MLDelayInit Pipe Network ✓

#### 2.1.8.2.1 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.

#### 2.1.8.3.2 Arguments

##### 2.1.8.4.3.1 Input

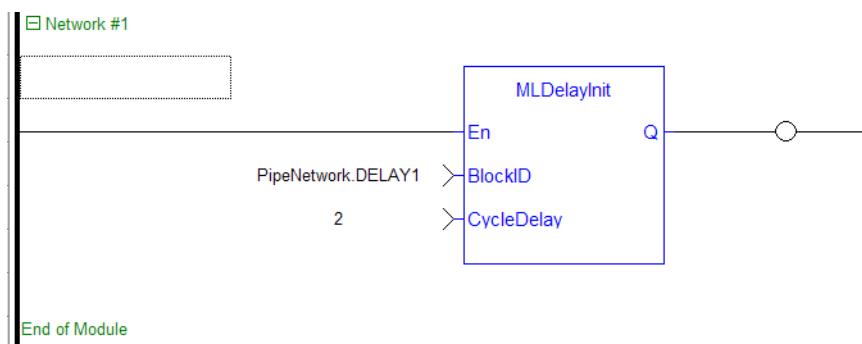
<b>BlockID</b>	<b>Description</b>	ID number of a created Pipe Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>CycleDelay</b>	<b>Description</b>	Number of delay cycles
	<b>Data type</b>	DINT
	<b>Range</b>	[0, 9]
	<b>Unit</b>	Cycle
	<b>Default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

##### 2.1.8.5.4.2 Example

##### 2.1.8.6.5.3 Structured Text

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

##### 2.1.8.7.6.4 Ladder Diagram



### 2.1.8.8.7.5 Function Block Diagram



## 2.1.9 Motion Library - Derivator

Name	Description	Return type
<a href="#">MLDerInit</a>	Initializes a derivator object	BOOL
<a href="#">MLDerReadInModPos</a>	Returns the input MODULO_POSITION of the Derivator block	None
<a href="#">MLDerWriteInModPos</a>	Sets the input MODULO_POSITION of the Derivator block	BOOL

### 2.1.9.1 MLDerInit

#### 2.1.9.2.1 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.

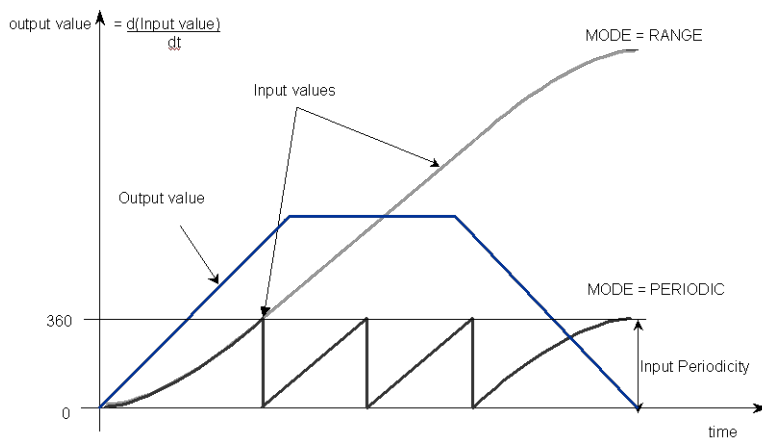


Figure 1-32: MLDerInit

#### 2.1.9.3.2 Arguments

##### 2.1.9.4.3.1 Input

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

<b>Unit</b>	User unit
<b>Default</b>	360.0

### 2.1.9.5.4.2 Output

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

### 2.1.9.6.5.3 Return Type

BOOL

### 2.1.9.7.6 Related Functions

[MLBlkCreate](#)

[MLDerReadInModPos](#)

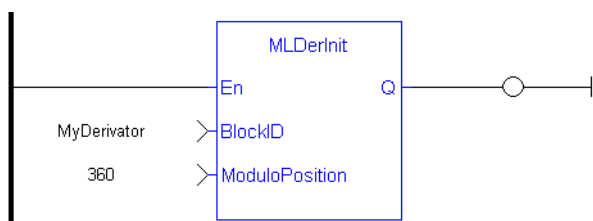
[MLDerWriteInModPos](#)

### 2.1.9.8.7 Example

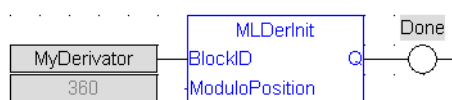
#### 2.1.9.9.8.1 Structured Text

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

#### 2.1.9.10.9.2 Ladder Diagram



#### 2.1.9.11.10.3 Function Block Diagram





### 2.1.9.12 MLDerReadInModPos Pipe Network ✓

#### 2.1.9.13.1 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

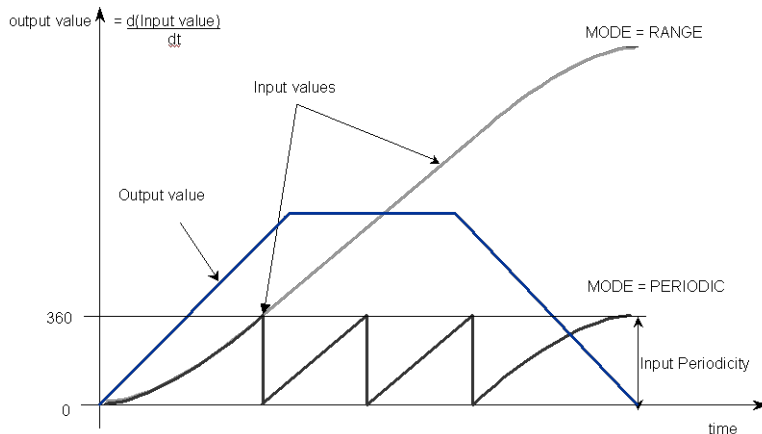


Figure 1-33: MLDerReadInModPos

#### NOTE

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

#### 2.1.9.14.2 Arguments

##### 2.1.9.15.3.1 Input

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

##### 2.1.9.16.4.2 Output

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

### 2.1.9.17.5 Related Functions

[MLDerWriteInModPos](#)

[MLDerInit](#)

### 2.1.9.18.6 Example

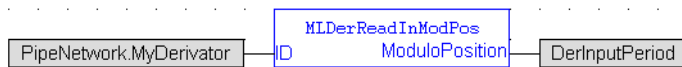
#### 2.1.9.19.7.1 Structured Text

```
//save the current input MODULO_POSITION of a Derivator object
DerInputPeriod := MLDerReadInModPos ( PipeNetwork.MyDerivator
);
```

#### 2.1.9.20.8.2 Ladder Diagram



#### 2.1.9.21.9.3 Function Block Diagram



**2.1.9.22 MLDerWriteInModPos** Pipe Network ✓

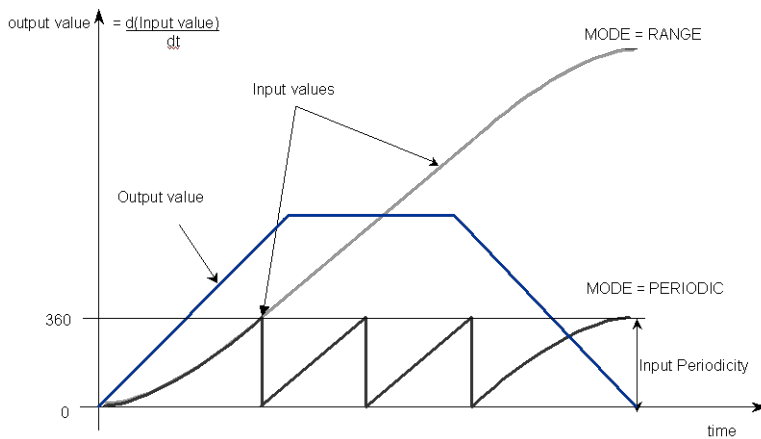
**2.1.9.23.1 Description**

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 1-34:** MLDerWriteInModPos

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

**2.1.9.24.2 Arguments**

**2.1.9.25.3.1 Input**

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

<b>Default</b>	—
----------------	---

**2.1.9.26.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns TRUE if the Input ModuloPosition value is changed
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.9.27.5.3 Return Type**

BOOL

**2.1.9.28.6 Related Functions**

[MLDerReadInModPos](#)

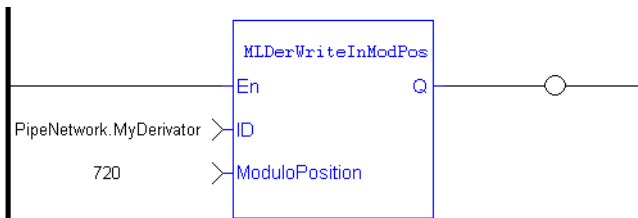
[MLDerInit](#)

**2.1.9.29.7 Example**

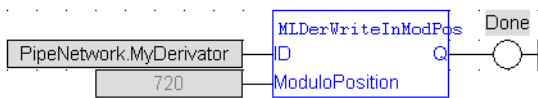
**2.1.9.30.8.1 Structured Text**

```
//change the input MODULO_POSITION of a Derivator object to 720
MLDerWriteInModPos ( PipeNetwork.MyDerivator, 720 );
```

**2.1.9.31.9.2 Ladder Diagram**



**2.1.9.32.10.3 Function Block Diagram**

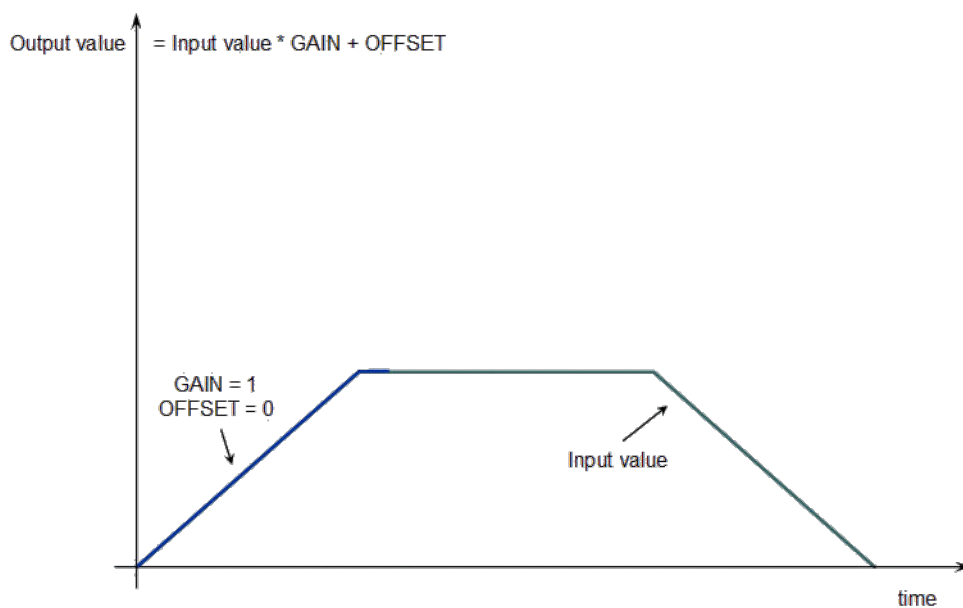


## 2.1.10 Motion Library - Gear

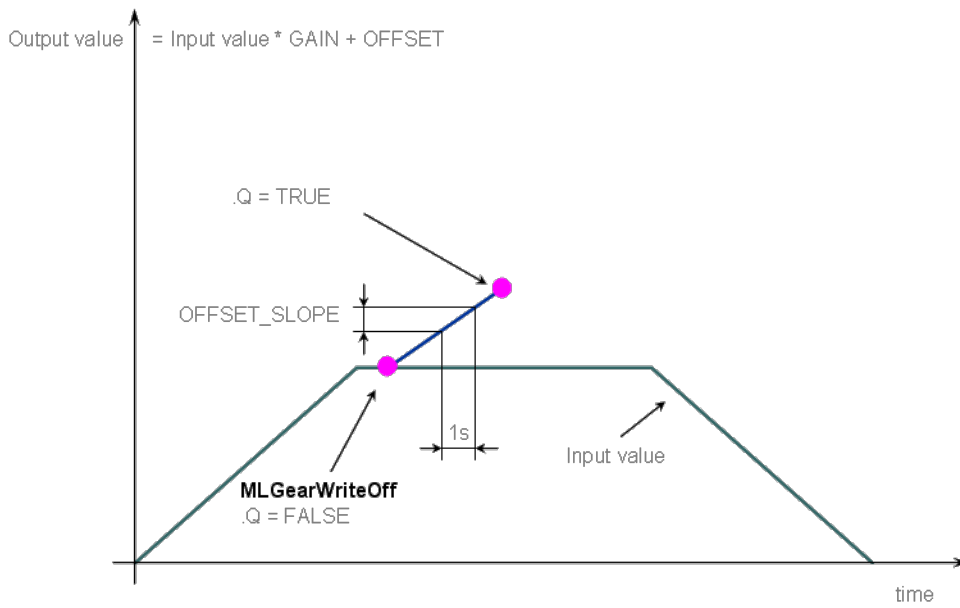
Name	Description	Return type
"MLGearInit" (p. 264)	Initializes a Gear Pipe Block with user-defined settings	BOOL
"MLGearReadOffset" (p. 268)	Returns the offset value of selected Gear Block	None
"MLGearReadOffSlp" (p. 269)	Returns the Offset Slope value of selected Gear Block	None
"MLGearReadRatio" (p. 271)	Returns the ratio value of a gear block	None
"MLGearReadRatSlp" (p. 272)	Returns the ratio slope value of a gear block	None
"MLGearWriteOff" (p. 273)	Sets the Offset value of a selected Gear Pipe Block	BOOL
"MLGearWriteOSlp" (p. 275)	Sets the Offset Slope value of a selected Gear Pipe Block	BOOL
"MLGearWriteRatio" (p. 277)	Sets the Ratio value of a selected Gear Pipe Block	BOOL
"MLGearWriteRatSlp" (p. 279)	Sets the Ratio Slope value of a selected Gear Pipe Block	BOOL

### 2.1.10.1 Usage example of Gear Functions

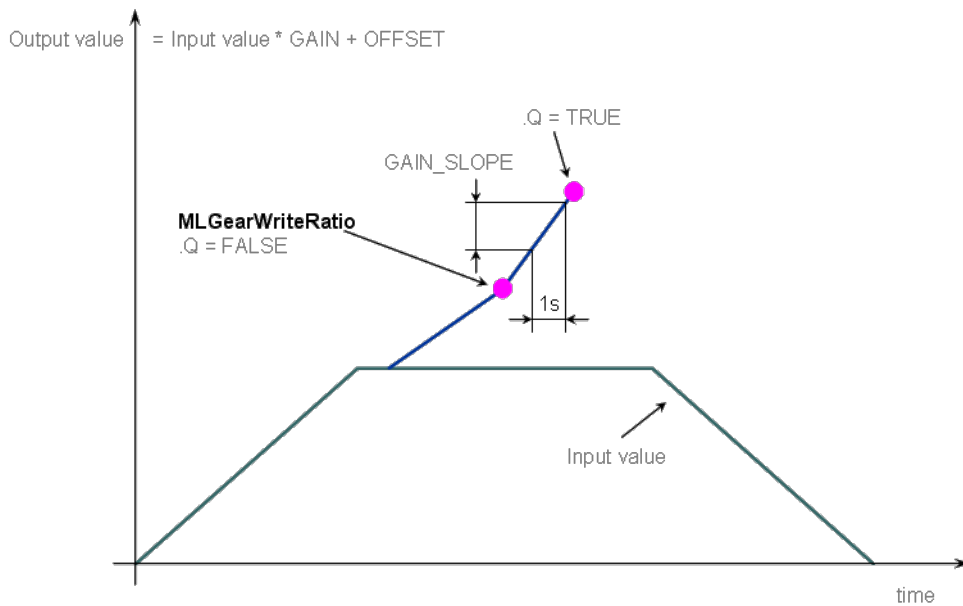
The output value starts with offset = 0 and gain = 1 (blue line)



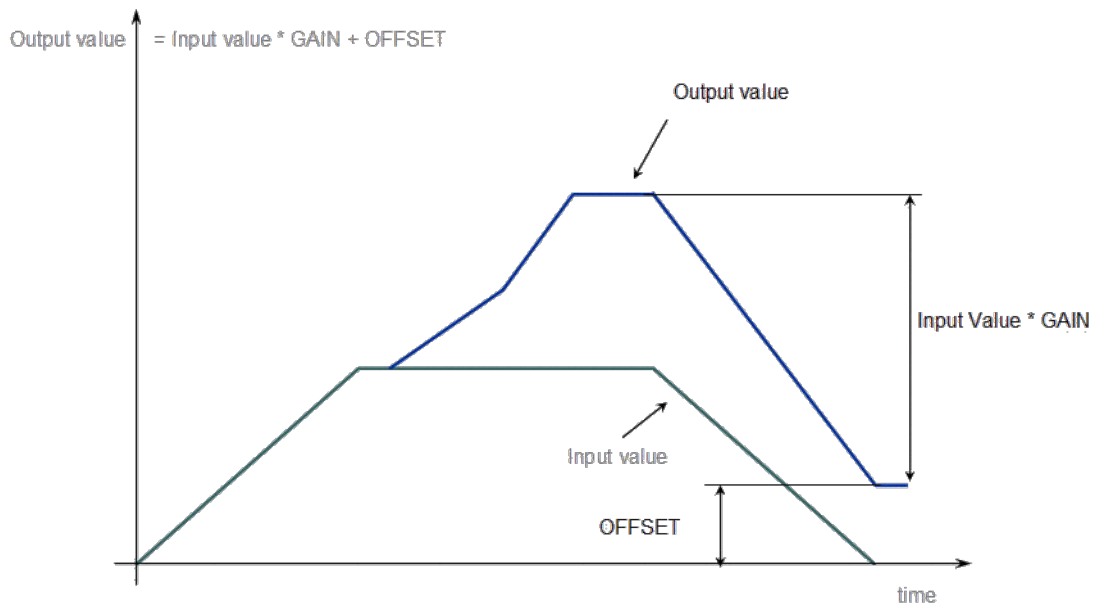
You can call the **MLGearWriteOff** function to modify the Offset (where Offset\_Slope is set with the **MLGearWriteOSlp** function).



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 1-35:** Gear Functions Usage

### 2.1.10.2 MLGearInit

#### 2.1.10.3.1 Description

Initializes a Gear Pipe Block for use in a PLC Program. Function block is automatically called if a Gear Block is added to the Pipe Network, with user-defined settings entered in the Pipe Blocks Properties screen.

The Pipe Block is assigned a **Name**, **Ratio**, **Offset**, and **Slopes** for changes in Ratio and Offset values. You can also choose between Modulo or Not modulo mode. Slopes set the limit at which step changes in Ratio and Offset are implemented.

The output of a Gear Block = Input value \* Ratio + Offset

#### NOTE

Be sure to set  $\text{RatioSlope} < (\text{Ratio} * \text{EtherCAT Update Rate})$ . The Gear block will make a jump (without a ramp) from one gear to the next when the RatioSlope is greater than the Ratio change factor multiplied by the update rate scale factor.

#### TIP

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

#### 2.1.10.4.2 Arguments

##### 2.1.10.5.3.1 Input

<b>BlockID</b>	<b>Description</b>	ID number of a created Pipe Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	GEAR
<b>Ratio</b>	<b>Description</b>	Ratio of new Gear Pipe Block. Values lower than 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	1.0
<b>Offset</b>	<b>Description</b>	Offset of new Gear Pipe Block. Values lower than 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a



	<b>Default</b>	0.0
<b>UseUserRatioSlope</b>	<b>Description</b>	FALSE to use the maximum Slope, causing an instantaneous gear change within one cycle. TRUE to use user-defined RatioSlope
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	FALSE
<b>RatioSlope</b>	<b>Description</b>	User-defined limit at which step changes in Ratio are implemented. Values lower than 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	1/sec
	<b>Default</b>	0.0
<b>UseUserOffsetSlope</b>	<b>Description</b>	FALSE to use the maximum Slope, causing an instantaneous gear change within one cycle. TRUE to use user-defined OffsetSlope
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	FALSE
<b>OffsetSlope</b>	<b>Description</b>	User-defined limit at which step changes in Offset are implemented. Values lower than 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	0.0
<b>Modulo</b>	<b>Description</b>	TRUE when mode is modulo. Modulo mode adapts the output values according to the ModuloPosition (modulo)
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1

<b>Unit</b>	n/a
<b>Default</b>	FALSE

**2.1.10.6.4.2 Output**

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

**2.1.10.7.5.3 Return Type**

BOOL

**2.1.10.8.6 Related Functions**

[MLBlkCreate](#)

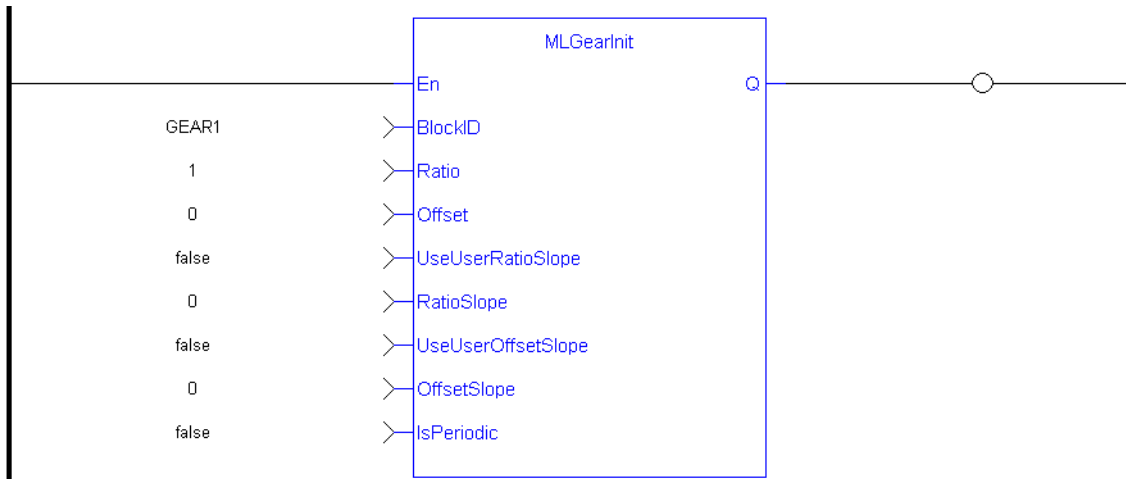
"MLGearWriteRatio" (p. 277)

**2.1.10.9.7 Example**

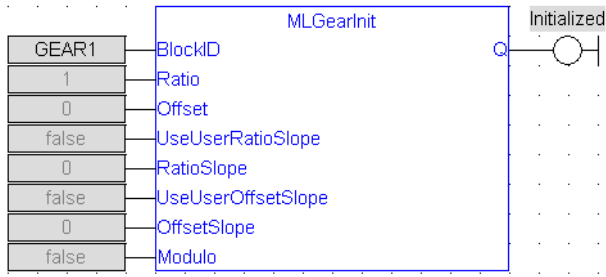
**2.1.10.10.8.1 Structured Text**

```
//Initialize a Gear Pipe Block named GEAR1 with:
// Ratio = 1,Offset = 0, User Ratio Slope OFF, User Ratio
// Slope = 0, Offset Slope = 0, and no Modulo
GEAR1 := MLBlkCreate( 'GEAR1', 'GEAR' );
MLGearInit( GEAR1, 1.0, 0.0, false, 0.0, false, 0.0, false);
```

**2.1.10.11.9.2 Ladder Diagram**



**2.1.10.12.10.3 Function Block Diagram**



**2.1.10.13 MLGearReadOffset** Pipe Network ✓

**2.1.10.14.1 Description**

Returns the Offset value of a selected Gear Block from the Pipe Network.

The output of a Gear Block = Input value \* Ratio + Offset

**2.1.10.15.2 Arguments**

**2.1.10.16.3.1 Input**

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

**2.1.10.17.4.2 Output**

<b>Offset</b>	<b>Description</b>	The offset value currently assigned to the selected Gear Pipe Block
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit

**2.1.10.18.5 Related Functions**

"MLGearWriteOff" (p. 273)

"MLGearInit" (p. 264)

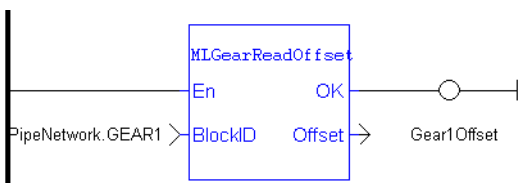
**2.1.10.19.6 Example**

**2.1.10.20.7.1 Structured Text**

```

//Find the Offset value of Gear1 Pipe Block
Gear1Offset := MLGearReadOffset( PipeNetwork.GEAR1 );
    
```

**2.1.10.21.8.2 Ladder Diagram**



**2.1.10.22.9.3 Function Block Diagram**



**2.1.10.23 MLGearReadOffSlp** Pipe Network ✓

**2.1.10.24.1 Description**

Returns the Offset Slope value of a selected Gear Block from the Pipe Network. Offset Slope sets the limit in User Units per Second at which step changes in offset are implemented. The default value when creating a Gear Block is OFFSET\_SLOPE\_MAX or infinite.

**2.1.10.25.2 Arguments**

**2.1.10.26.3.1 Input**

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

**2.1.10.27.4.2 Output**

<b>Slope</b>	<b>Description</b>	The offset slope value currently assigned to the selected Gear Pipe Block, which may be a different sign than what is programmed with MLGearWriteOSlp.
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit/sec

**2.1.10.28.5 Related Functions**

"MLGearWriteOSlp" (p. 275)

"MLGearInit" (p. 264)

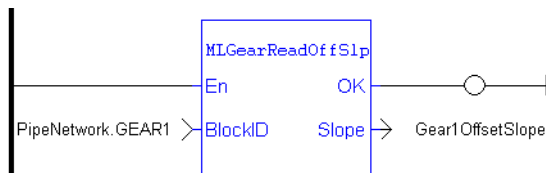
**2.1.10.29.6 Example**

**2.1.10.30.7.1 Structured Text**

```

//Find the Offset Slope value of Gear1 Pipe Block
Gear1OffsetSlope := MLGearReadOffSlp(PipeNetwork.GEAR1);
    
```

**2.1.10.31.8.2 Ladder Diagram**



**2.1.10.32.9.3 Function Block Diagram**





**2.1.10.33 MLGearReadRatio** **2.1.10.34.1 Description**

Returns the Ratio value of a selected Gear Block from the Pipe Network. The output of a Gear Block = Input value \* Ratio + Offset

**2.1.10.35.2 Arguments****2.1.10.36.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID number of an initialized Gear Pipe Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.10.37.4.2 Output**

<b>Ratio</b>	<b>Description</b>	The Ratio value currently assigned to the selected Gear Pipe Block
	<b>Data type</b>	LREAL
	<b>Unit</b>	n/a

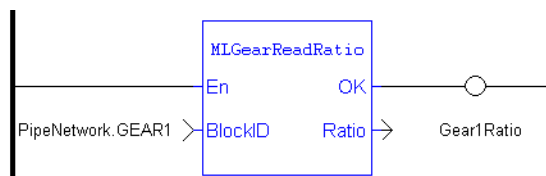
**2.1.10.38.5 Related Functions**

"MLGearWriteRatio" (p. 277)

"MLGearInit" (p. 264)

**2.1.10.39.6 Example****2.1.10.40.7.1 Structured Text**

```
//Find the Ratio value of Gear1 Pipe Block
Gear1Ratio := MLGearReadRatio(PipeNetwork.GEAR1);
```

**2.1.10.41.8.2 Ladder Diagram****2.1.10.42.9.3 Function Block Diagram**

**2.1.10.43 MLGearReadRatSlp** Pipe Network ✓

**2.1.10.44.1 Description**

Returns the Ratio Slope value of a selected Gear Block from the Pipe Network. Ratio Slope sets the limit in 1/Seconds (or s<sup>-1</sup>) at which step changes in Ratio are implemented.

**2.1.10.45.2 Arguments**

**2.1.10.46.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID number of an initialized Gear Pipe Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.10.47.4.2 Output**

<b>Slope</b>	<b>Description</b>	The Ratio Slope value currently assigned to the selected Gear Pipe Block, , which may be a different sign than what is programmed with MLGearWriteRatSlp.
	<b>Data type</b>	LREAL
	<b>Unit</b>	1/sec (or s <sup>-1</sup> )

**2.1.10.48.5 Related Functions**

["MLGearWriteRatSlp" \(p. 279\)](#)

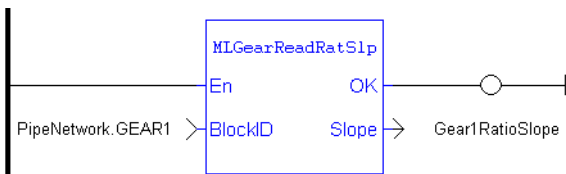
["MLGearInit" \(p. 264\)](#)

**2.1.10.49.6 Example**

**2.1.10.50.7.1 Structured Text**

```
//Find the Ratio Slope value of Gear1 Pipe Block
Gear1RatioSlope := MLGearReadRatSlp(PipeNetwork.GEAR1);
```

**2.1.10.51.8.2 Ladder Diagram**



**2.1.10.52.9.3 Function Block Diagram**





**2.1.10.53 MLGearWriteOff** **2.1.10.54.1 Description**

Sets the Offset value of a selected Gear Pipe Block.

The output of a Gear Block = Input value \* Ratio + Offset

**TIP**

Values lower than 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)

**2.1.10.55.2 Arguments****2.1.10.56.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID number of an initialized Gear Pipe Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Offset</b>	<b>Description</b>	New Offset value to be assigned to selected Gear Pipe Block. Values lower than 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—

**2.1.10.57.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns TRUE if Offset value is changed in the selected Gear Pipe Block
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.10.58.5.3 Return Type**

BOOL

**2.1.10.59.6 Related Functions**

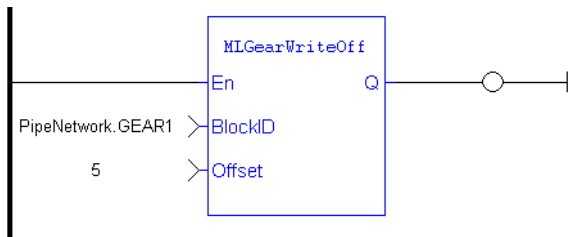
["MLGearReadOffset" \(p. 268\)](#)

["MLGearInit" \(p. 264\)](#)

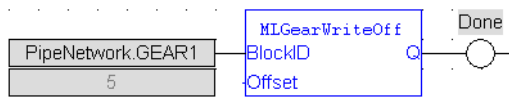
**2.1.10.60.7 Example****2.1.10.61.8.1 Structured Text**

```
//Set the Offset value of Gear1 Pipe Block to 5 User Units  
MLGearWriteOff(PipeNetwork.GEAR1, 5.0);
```

### 2.1.10.62.9.2 Ladder Diagram



### 2.1.10.63.10.3 Function Block Diagram



**2.1.10.64 MLGearWriteOSIp** Pipe Network ✓**2.1.10.65.1 Description**

Sets the Offset Slope value of a selected Gear Pipe Block. Offset Slope sets the limit in User Units per Second at which step changes in offset are implemented. The default value when creating a Gear Block is OFFSET\_SLOPE\_MAX or infinite.

**TIP**

Values lower than 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)

**2.1.10.66.2 Arguments****2.1.10.67.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID number of an initialized Gear Pipe Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Slope</b>	<b>Description</b>	New Offset Slope value to be assigned to selected Gear Pipe Block. Values lower than 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	—

**2.1.10.68.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns TRUE if Offset Slope value is changed in the selected Gear Pipe Block
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.10.69.5.3 Return Type**

BOOL

**2.1.10.70.6 Related Functions**

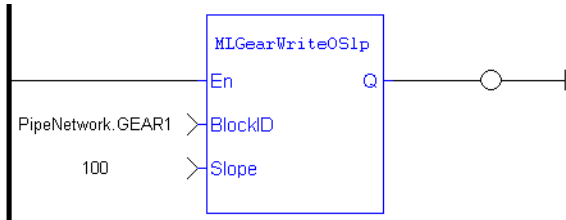
["MLGearReadOffSlp" \(p. 269\)](#)

["MLGearInit" \(p. 264\)](#)

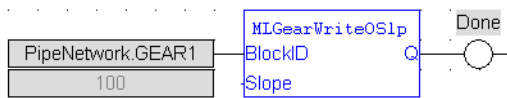
**2.1.10.71.7 Example****2.1.10.72.8.1 Structured Text**

```
//Set the Offset Slope value of Gear1 Pipe Block to 100  
MLGearWriteOSlp(PipeNetwork.GEAR1, 100.0);
```

### 2.1.10.73.9.2 Ladder Diagram



### 2.1.10.74.10.3 Function Block Diagram



**2.1.10.75 MLGearWriteRatio** Pipe Network ✓**2.1.10.76.1 Description**

Set the Ratio value of a selected Gear Pipe Block.

The output of a Gear Block = Input value \* Ratio + Offset

**TIP**

Values lower than 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)

**2.1.10.77.2 Arguments****2.1.10.78.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID number of an initialized Gear Pipe Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Ratio</b>	<b>Description</b>	New Ratio value to be assigned to selected Gear Pipe Block. Values lower than 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.10.79.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns TRUE if Ratio value is changed in the selected Gear Pipe Block
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.10.80.5.3 Return Type**

BOOL

**2.1.10.81.6 Related Functions**

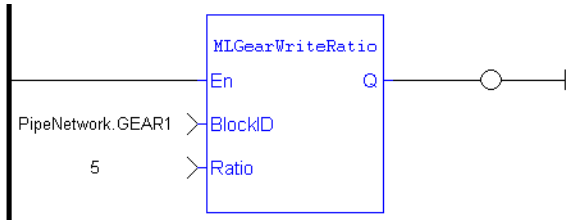
["MLGearReadRatio" \(p. 271\)](#)

["MLGearInit" \(p. 264\)](#)

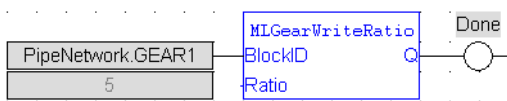
**2.1.10.82.7 Example****2.1.10.83.8.1 Structured Text**

```
//Set the Ratio value of Gear1 Pipe Block to 5  
MLGearWriteRatio(PipeNetwork.GEAR1, 5.0);
```

### 2.1.10.84.9.2 Ladder Diagram



### 2.1.10.85.10.3 Function Block Diagram



### 2.1.10.86 MLGearWriteRatSlp

#### 2.1.10.87.1 Description

Set the Ratio Slope value of a selected Gear Pipe Block. Ratio Slope sets the limit at which step changes in ratio are implemented.

#### NOTE

Be sure to set  $\text{RatioSlope} < (\text{Ratio} * \text{EtherCAT Update Rate})$ . The Gear block will make a jump (without a ramp) from one gear to the next when the RatioSlope is greater than the Ratio change factor multiplied by the update rate scale factor.

#### TIP

Values lower than 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)

#### NOTE

The GEAR block output will add a position offset to the GEAR block input when using a RatioSlope. See "[RatioSlope Offset](#)" (p. 280) in the Examples below.

#### 2.1.10.88.2 Arguments

##### 2.1.10.89.3.1 Input

<b>BlockID</b>	<b>Description</b>	ID number of an initialized Gear Pipe Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Slope</b>	<b>Description</b>	New Ratio Slope value to be assigned to selected Gear Pipe Block. Values lower than 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	1/sec
	<b>Default</b>	—

##### 2.1.10.90.4.2 Output

<b>Default (.Q)</b>	<b>Description</b>	Returns TRUE if Ratio Slope value is changed in the selected Gear Pipe Block
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

##### 2.1.10.91.5.3 Return Type

BOOL

### 2.1.10.92.6 Related Functions

"MLGearReadOffSlp" (p. 269)

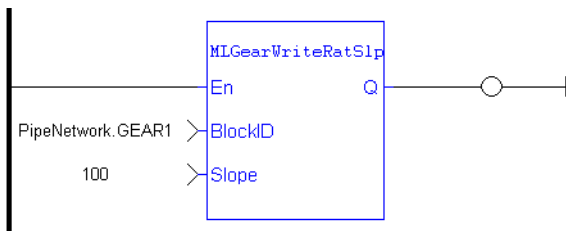
"MLGearInit" (p. 264)

### 2.1.10.93.7 Example

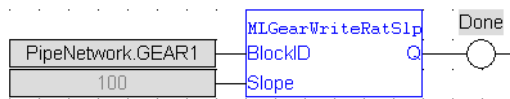
#### 2.1.10.94.8.1 Structured Text

```
//Set the Ratio Slope value of Gear1 Pipe Block to 100
MLGearWriteRatSlp(PipeNetwork.GEAR1, 100.0);
```

#### 2.1.10.95.9.2 Ladder Diagram



#### 2.1.10.96.10.3 Function Block Diagram



#### 2.1.10.97.11.4 RatioSlope Offset

If MLGearWriteRatSlp is set as `MLGearWriteRatSlp( PipeNetwork.GEAR1 12 , Gear1RatioSlope 500.0 )`; to generate a ramp (instead of a step) when going from a gear ratio of 1 to 2, then there will be a position offset when the gear ratio settles as 2. In the image below the ratio goes from 1.0 to 2.0; Green is PN Gear Block Output and Red is Gearbox Input.



1. Green line: PN Gear Block Output
2. Red line: PN Gearbox Input
3. When the ratio is changed
4. Phase difference

If MLGearWriteRatSlp is set without a ramp,

```
MLGearWriteRatSlp( PipeNetwork.GEAR1 12 , Gear1RatioSlope 1e+301 );
```

, then there will not be an offset.



1. Green line: PN Gear Block Output
2. Red line: PN Gearbox Input
3. When the ratio changes
4. Synced



## 2.1.11 Motion Library - Integrator

Name	Description	Return type
<a href="#">MLIntInit</a>	Initializes an integrator object	BOOL
<a href="#">MLIntWriteOutVal</a>	Sets the output value of an integrator object	BOOL

### 2.1.11.1 MLIntInit Pipe Network ✓

#### 2.1.11.2.1 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.

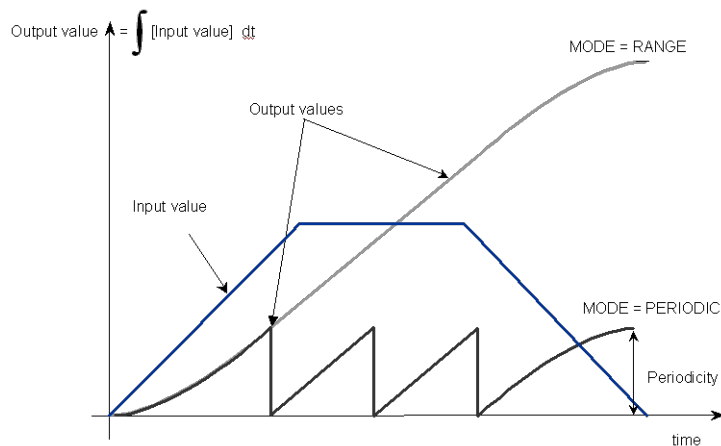


Figure 1-36: MLIntInit

#### 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.

#### 2.1.11.3.2 Arguments

##### 2.1.11.4.3.1 Input

<b>BlockID</b>	<b>Description</b>	ID number of a created Pipe Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ModuloPosition</b>	<b>Description</b>	Output ModuloPosition of Integrator object
	<b>Data type</b>	LREAL
	<b>Range</b>	—

	<b>Unit</b>	User unit
	<b>Default</b>	360.0
<b>Modulo</b>	<b>Description</b>	TRUE when mode is modulo. Modulo mode adapts the output values according to the ModuloPosition (modulo)
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	TRUE

**2.1.11.5.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns TRUE if the Integrator object is initialized
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.11.6.5.3 Return Type**

BOOL

**2.1.11.7.6 Related Functions**

[MLBlkCreate](#)

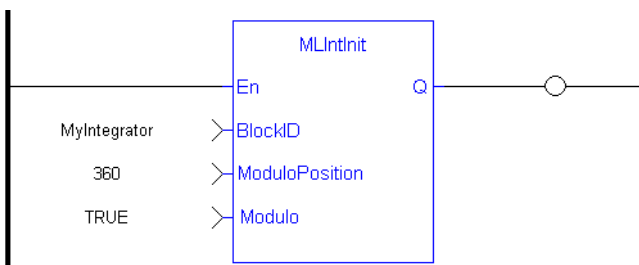
[MLIntWriteOutVal](#)

**2.1.11.8.7 Example**

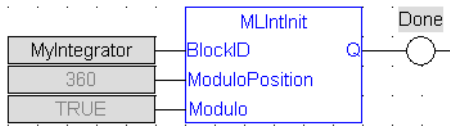
**2.1.11.9.8.1 Structured Text**

```
//Initiate an Integrator Pipe Block named "MyIntegrator" with a
Modulo of 360
MLIntInit(MyIntegrator, 360.0, true );
```

**2.1.11.10.9.2 Ladder Diagram**



**2.1.11.11.10.3 Function Block Diagram**



**2.1.11.12 MLIntWriteOutVal** **2.1.11.13.1 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.

**2.1.11.14.2 Arguments****2.1.11.15.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID number of an initiated Integrator object
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Value</b>	<b>Description</b>	Desired new output value of the selected Integrator object
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—

**2.1.11.16.4.2 Output**

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

**2.1.11.17.5.3 Return Type**

BOOL

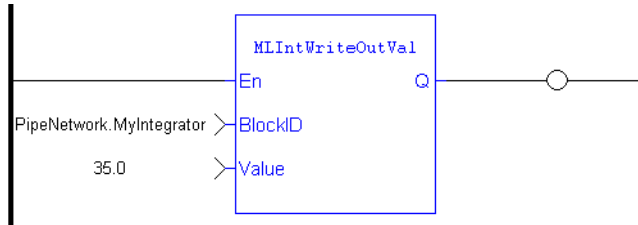
**2.1.11.18.6 Related Functions**

[MLIntInit](#)

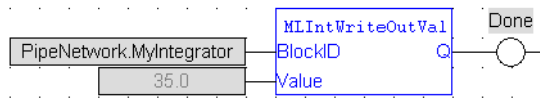
**2.1.11.19.7 Example****2.1.11.20.8.1 Structured Text**

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

### 2.1.11.21.9.2 Ladder Diagram



### 2.1.11.22.10.3 Function Block Diagram



## 2.1.12 Motion Library - Master

### TIP

- For an example of Master Functions, see "[Usage example of Master Functions](#)" (p. 286)

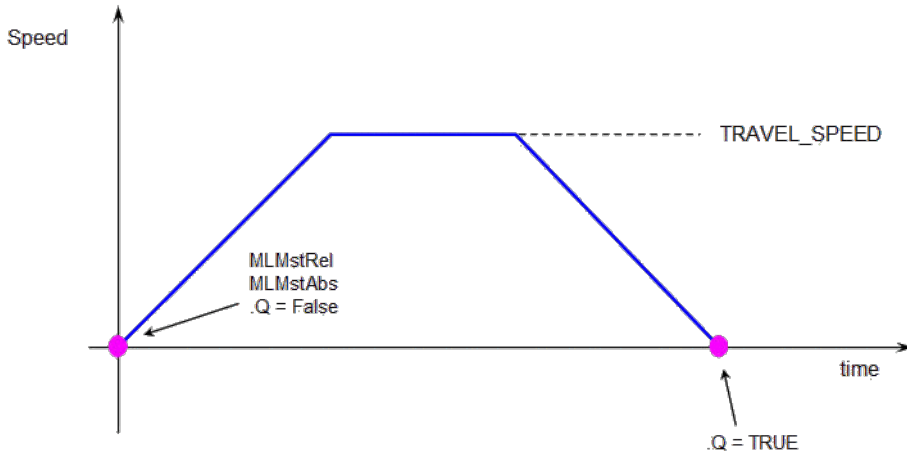
#### Functions sorted by types:

Motion Control	Inquiry Functions	Position setting
"MLMstInit" (p. 294)	"MLMstReadAccel" (p. 297)	"MLMstAbs" (p. 287)
"MLMstRun" (p. 307)	"MLMstReadDecel" (p. 299)	"MLMstAdd" (p. 290)
"MLMstWriteAccel" (p. 311)	"MLMstReadInitPos" (p. 301)	"MLMstForcePos" (p. 292)
"MLMstWriteDecel" (p. 313)	"MLMstReadSpeed" (p. 303)	"MLMstRel" (p. 305)
"MLMstWriteSpeed" (p. 317)	"MLMstStatus" (p. 309)	

#### Functions sorted in alphabetical order:

Name	Description	Return type
"MLMstAbs" (p. 287)	Does an absolute move	BOOL
"MLMstAdd" (p. 290)	Does an additive move relative for a specified distance from the endpoint of the previous move	BOOL
"MLMstForcePos" (p. 292)	Forces the specified position. Possible only when the block is <b>not</b> moving.	BOOL
"MLMstInit" (p. 294)	Initializes a master object (TMP generator)	BOOL
"MLMstReadAccel" (p. 297)	Gets the present acceleration value of a master block	None
"MLMstReadDecel" (p. 299)	Gets the present deceleration value of a master block	None
"MLMstReadInitPos" (p. 301)	Gets the initial position of a master block	None
"MLMstReadSpeed" (p. 303)	Gets the speed of a master block	None
"MLMstRel" (p. 305)	Does an Relative move for a specified distance from the current position	BOOL
"MLMstRun" (p. 307)	Jogs at the specified speed. Returns TRUE if the function succeeded	BOOL
"MLMstStatus" (p. 309)	Returns the status of the generator	DINT
"MLMstWriteAccel" (p. 311)	Sets the acceleration of a master block	BOOL
"MLMstWriteDecel" (p. 313)	Sets the deceleration of a master block	BOOL
"MLMstWriteInitPos" (p. 315)	Sets the initial position of a master block	BOOL
"MLMstWriteSpeed" (p. 317)	Sets the speed of a master block	BOOL

### 2.1.12.1 Usage example of Master Functions



**MLMstRun(0.0)** reduce the speed down to 0.

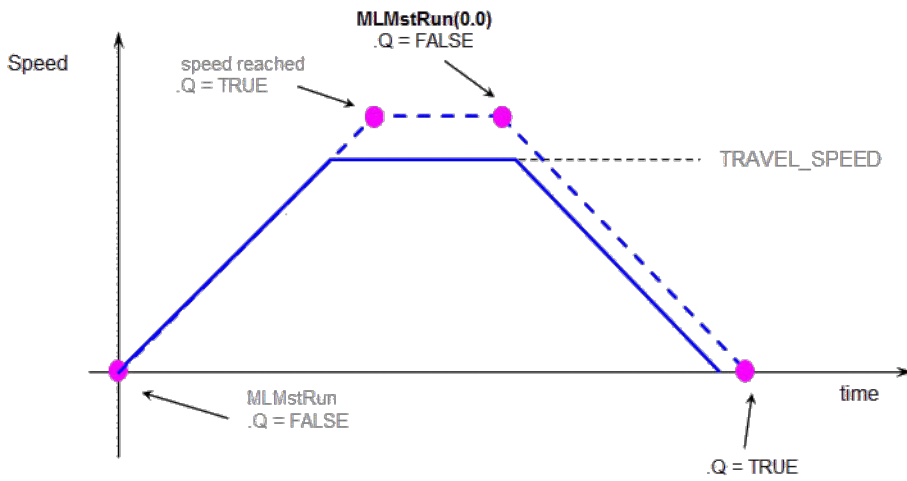


Figure 1-37: Master Functions Usage

**2.1.12.2 MLMstAbs** Pipe Network ✓

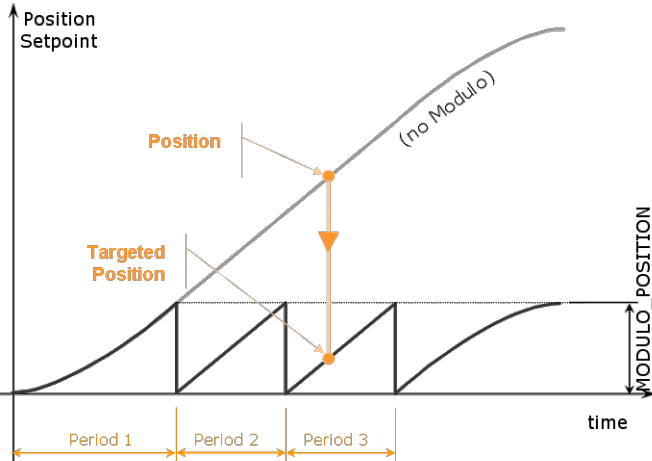
**2.1.12.3.1 Description**

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

**2.1.12.4.2 Arguments**

**2.1.12.5.3.1 Input**

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

<b>Position</b>	<p><b>Description</b></p> <p>Sets the value of the absolute destination position. 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"> <li>• If the Position input is between 0 and the Modulo Position, then the Master Block moves within the <b>current</b> period (no position rollover).</li> <li>• If the Position input is greater than the Modulo Position, then the Master Block moves during one of the <b>next</b> period (positive position rollover).</li> </ul>  <p>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 <b>previous</b> period (negative position rollover).</p>
	<p><b>Data type</b> LREAL</p>
	<p><b>Range</b> —</p>
	<p><b>Unit</b> User unit</p>
	<p><b>Default</b> —</p>

**2.1.12.6.4.2 Output**

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

**TIP**

To reduce the load on the CPU, call MLMstAbs only once for each move. This can be achieved by adding a control variable, which is illustrated below.

```

// Master: modulo is on and modulo range is 0 - 360
with rollover at 360
    
```



```

If Not MoveStarted Then
    Position := 500;
    MLMstAbs( PipeNetwork.MASTER, Position);
    MovesStarted := TRUE;
End_if;

```

#### NOTE

Perform one of the following to prevent undesired axis movement if modulo is turned on.

- Verify that the **Position** value is within the modulo range.
- If the **Position** value is outside of the modulo range, call MLMstAbs function only once. See the **TIP** above for an example of how to do this.

#### 2.1.12.7.5 Related Functions

"MLMstWriteSpeed" (p. 317)

"MLMstWriteDecel" (p. 313)

#### 2.1.12.8.6 Examples

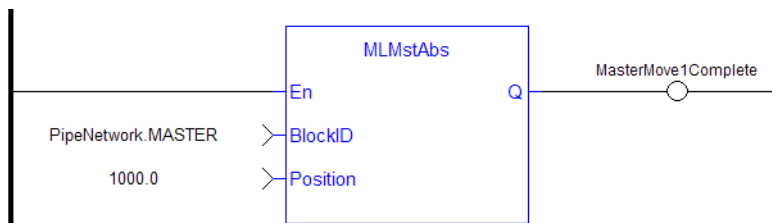
##### 2.1.12.9.7.1 Structured Text

```

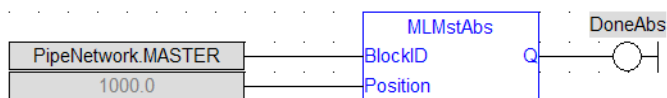
//Make an absolute positon move with a Master block called
"MASTER" to position 1000.0
MLMstAbs( PipeNetwork.MASTER, 1000.0 );

```

##### 2.1.12.10.8.2 Ladder Diagram



##### 2.1.12.11.9.3 Function Block Diagram



**2.1.12.12 MLMstAdd** **2.1.12.13.1 Description**

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

**2.1.12.14.2 Arguments****2.1.12.15.3.1 Input**

<b>EN</b>	<b>Description</b>	Enables FB to be executed
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Block ID</b>	<b>Description</b>	ID name of the Master Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>DeltaPos</b>	<b>Description</b>	Relative distance to move
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—

**2.1.12.16.4.2 Output**

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

**2.1.12.17.5 Related Functions**

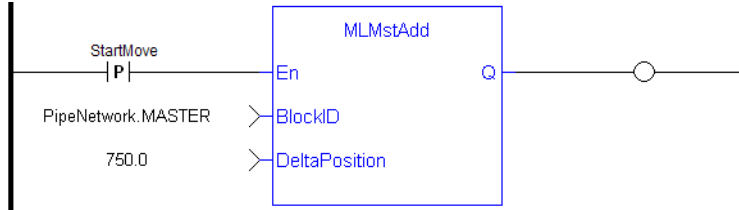
["MLMstWriteSpeed"](#) (p. 317)

["MLMstWriteDecel"](#) (p. 313)

**2.1.12.18.6 Example****2.1.12.19.7.1 Structured Text**

```
//At the endpoint of the previous move, with the Master pipe
block named "MASTER", make a move of 750.0
MLMstAdd( PipeNetwork.MASTER, 750.0 );
```

### 2.1.12.20.8.2 Ladder Diagram



#### NOTE

You must use a [pulse contact](#) to start the FB

### 2.1.12.21.9.3 Function Block Diagram



**2.1.12.22 MLMstForcePos** Pipe Network ✓**2.1.12.23.1 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.

**2.1.12.24.2 Arguments****2.1.12.25.3.1 Input**

<b>EN</b>	<b>Description</b>	Enables FB to be executed
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Block ID</b>	<b>Description</b>	ID name of the Master Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Position</b>	<b>Description</b>	Defines the Master starting position when the motion starts
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—

**2.1.12.26.4.2 Output**

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

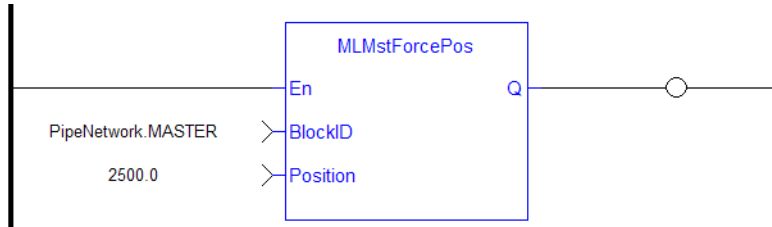
**2.1.12.27.5 Related Functions**

["MLMstReadInitPos" \(p. 301\)](#)

**2.1.12.28.6 Example****2.1.12.29.7.1 Structured Text**

```
//Reset the output position of the Master Pipe Block named
"Master" to 2500.0
MLMstForcePos(PipeNetwork.Master, 2500.0);
```

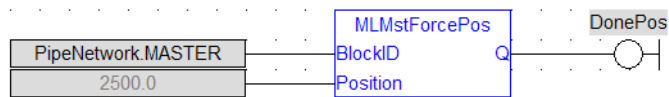
### 2.1.12.30.8.2 Ladder Diagram



#### NOTE

You must use a [pulse contact](#) to start the FB

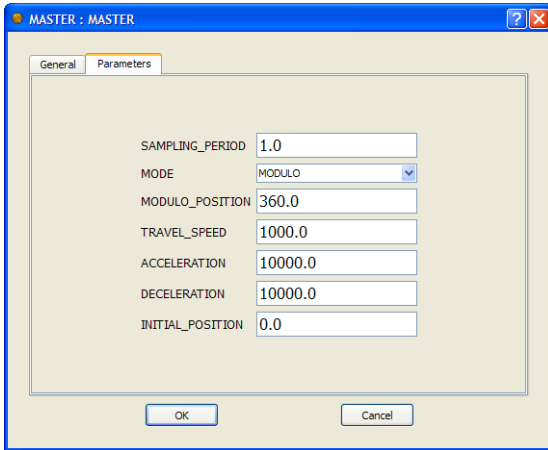
### 2.1.12.31.9.3 Function Block Diagram



**2.1.12.32 MLMstInit** Pipe Network ✓

**2.1.12.33.1 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 pipe block (see figure below), the Inputs for this function are initialized by default.



**Figure 1-38:** TMP Initialization

**2.1.12.34.2 Arguments**

**2.1.12.35.3.1 Input**

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

	<b>Default</b>	—
<b>Speed</b>	<b>Description</b>	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
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a> / sec
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	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
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a> /sec <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	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
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a> /sec <sup>2</sup>
	<b>Default</b>	—
<b>Initial Position</b>	<b>Description</b>	Initial position value expressed in user logical units. Used only at the pipe activation to initialize the position starting point
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>Modulo</b>	<b>Description</b>	The available modes are Modulo (True) or No modulo (False)
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.12.36.4.2 Output**

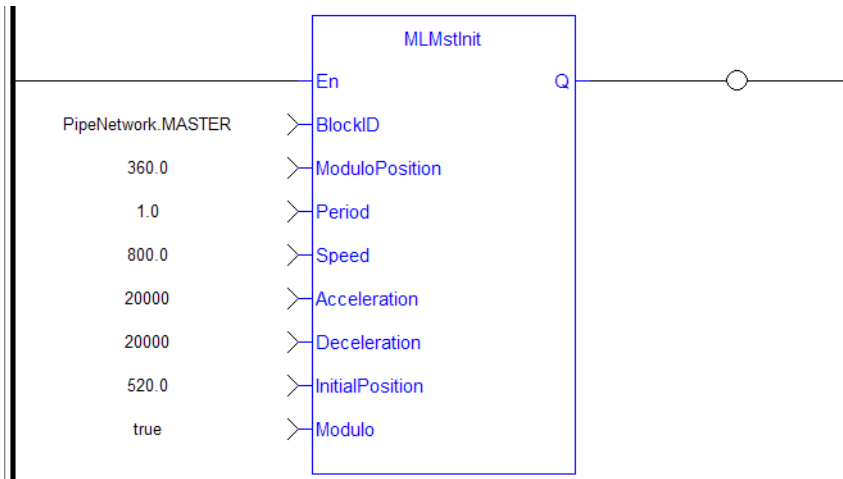
<b>Default (.Q)</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.12.37.5 Example**

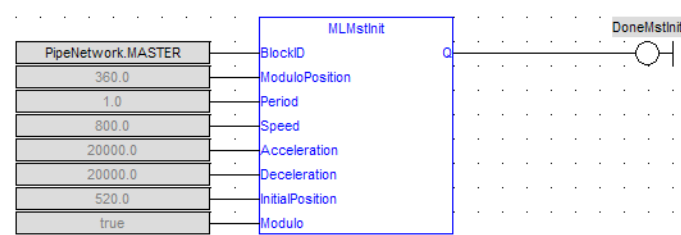
**2.1.12.38.6.1 Structured Text**

```
//Initialize a Master Pipe Block named "MASTER" to a Modulo of
360, motion generator sample period of 1, Speed of 1000.0, Accel
and Decel of 10000.0, Initial position of 0.0,
MLMstInit( PipeNetwork.MASTER, 360.0, 1.0, 1000.0, 10000.0,
10000.0, 0.0, true );
```

**2.1.12.39.7.2 Ladder Diagram**



**2.1.12.40.8.3 Function Block Diagram**





**2.1.12.41 MLMstReadAccel** Pipe Network ✓**2.1.12.42.1 Description**

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

**2.1.12.43.2 Arguments****2.1.12.44.3.1 Input**

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

**2.1.12.45.4.2 Output**

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

**2.1.12.46.5 Related Functions**

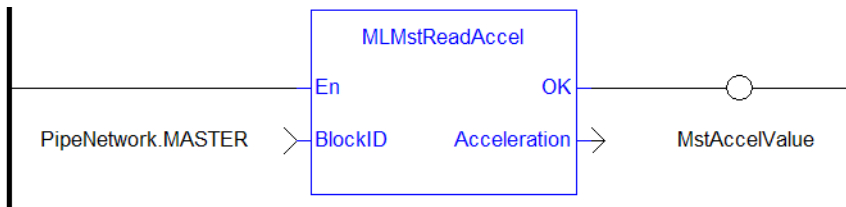
["MLMstReadSpeed"](#) (p. 303)

["MLMstReadDecel"](#) (p. 299)

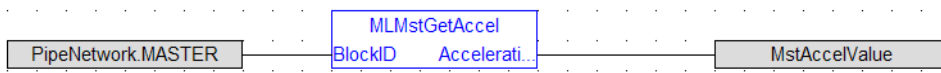
**2.1.12.47.6 Example****2.1.12.48.7.1 Structured Text**

```
// Read the present acceleration of a Pipe Block named
"MASTER"
MLMstReadAccel( PipeNetwork.MASTER );
```

**2.1.12.49.8.2 Ladder Diagram**



### 2.1.12.50 Function Block Diagram



**2.1.12.51 MLMstReadDecel** Pipe Network ✓**2.1.12.52.1 Description**

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

**2.1.12.53.2 Arguments****2.1.12.54.3.1 Input**

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

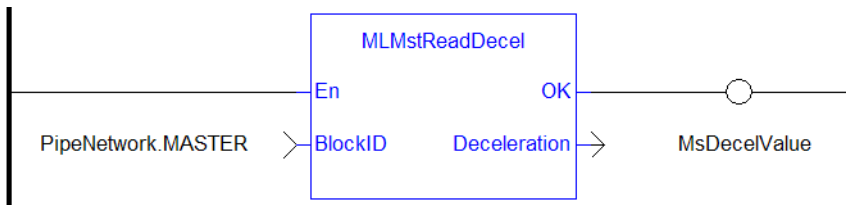
**2.1.12.55.4.2 Output**

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

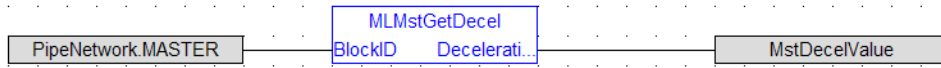
**2.1.12.56.5 Example****2.1.12.57.6.1 Structured Text**

```
// Read the present deceleration of a Pipe Block named
"MASTER"
MLMstReadDecel( PipeNetwork.MASTER );
```

**2.1.12.58.7.2 Ladder Diagram**



### 2.1.12.59.8.3 Function Block Diagram



**2.1.12.60 MLMstReadInitPos** **2.1.12.61.1 Description**

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

**2.1.12.62.2 Arguments****2.1.12.63.3.1 Input**

<b>EN</b>	<b>Description</b>	Enables FB to be executed
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Block ID</b>	<b>Description</b>	PipeNetwork Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—

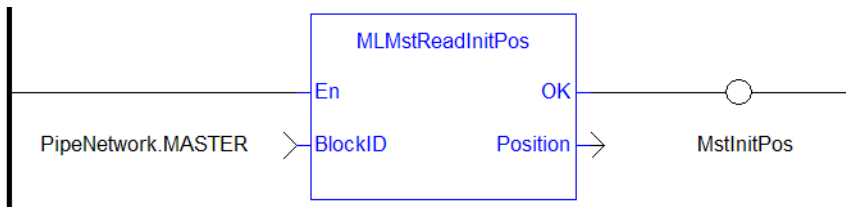
**2.1.12.64.4.2 Output**

<b>OK</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Position</b>	<b>Description</b>	Returns Initial Position
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit

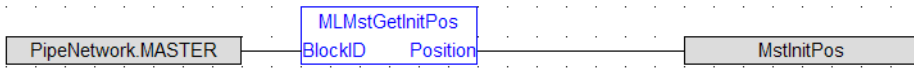
**2.1.12.65.5 Example****2.1.12.66.6.1 Structured Text**

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

**2.1.12.67.7.2 Ladder Diagram**



### 2.1.12.68.8.3 Function Block Diagram



**2.1.12.69 MLMstReadSpeed** **2.1.12.70.1 Description**

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

**2.1.12.71.2 Arguments****2.1.12.72.3.1 Input**

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

**2.1.12.73.4.2 Output**

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

**2.1.12.74.5 Related Functions**

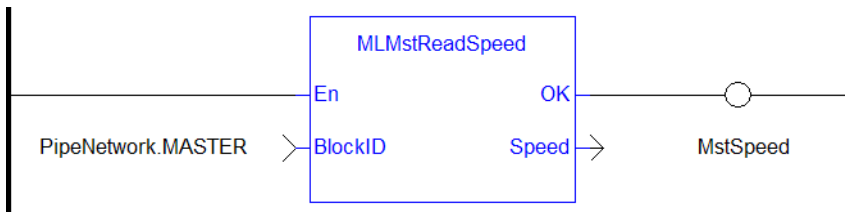
["MLMstReadAccel" \(p. 297\)](#)

["MLMstReadDecel" \(p. 299\)](#)

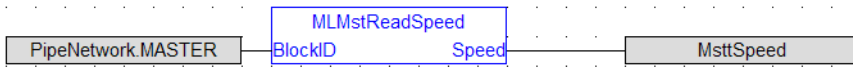
**2.1.12.75.6 Example****2.1.12.76.7.1 Structured Text**

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

**2.1.12.77.8.2 Ladder Diagram**



### 2.1.12.78.9.3 Function Block Diagram





**2.1.12.79 MLMstRel** **2.1.12.80.1 Description**

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

**2.1.12.81.2 Arguments****2.1.12.82.3.1 Input**

<b>EN</b>	<b>Description</b>	Enables FB to be executed
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Block ID</b>	<b>Description</b>	ID name of the Master Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>DeltaPos</b>	<b>Description</b>	Relative distance to move
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—

**2.1.12.83.4.2 Output**

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

**2.1.12.84.5 Related Functions**

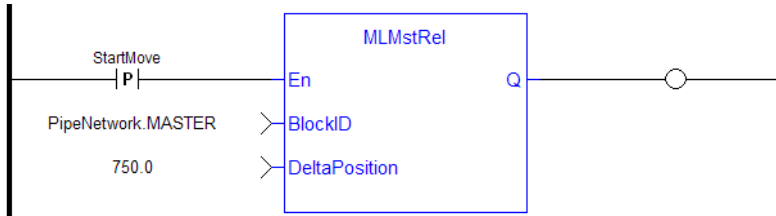
["MLMstWriteSpeed" \(p. 317\)](#)

["MLMstWriteDecel" \(p. 313\)](#)

**2.1.12.85.6 Example****2.1.12.86.7.1 Structured Text**

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

### 2.1.12.87.8.2 Ladder Diagram



#### NOTE

You must use a [pulse contact](#) to start the FB

### 2.1.12.88.9.3 Function Block Diagram



**2.1.12.89 MLMstRun** Pipe Network ✓**2.1.12.90.1 Description**

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

**2.1.12.91.2 Arguments****2.1.12.92.3.1 Input**

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

**2.1.12.93.4.2 Output**

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

**2.1.12.94.5 Related Functions**

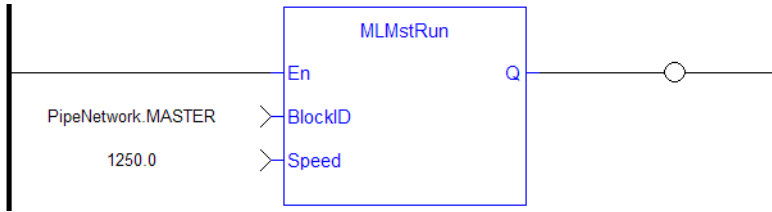
["MLMstWriteSpeed"](#) (p. 317)

["MLMstWriteDecel"](#) (p. 313)

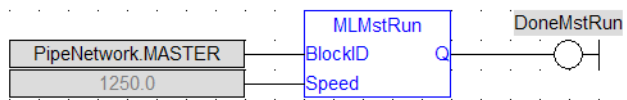
**2.1.12.95.6 Example****2.1.12.96.7.1 Structured Text**

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

### 2.1.12.97.8.2 Ladder Diagram



### 2.1.12.98.9.3 Function Block Diagram



**2.1.12.99 MLMstStatus** **2.1.12.100.1 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 jog move (MLMstRun) is commanded and the current speed is not at the commanded speed.
1	(Stable state Running or Stopped) is entered when a jog move (MLMstRun) is commanded and the current speed is at the commanded speed.  (Stable state Running or Stopped) is entered when a position move is programmed and motion is completed.
2	(Speed change) is entered when the current speed is greater than the commanded speed.
3	(Speed reversal while positioning) is entered when a position move is programmed and the distance to go requires a speed reversal.
4	(Acceleration while positioning) current speed is below the travel speed
5	(Constant Speed while positioning) is entered when a positioning move is commanded and the current speed is at the commanded speed.
6	(Deceleration while positioning) is entered when a positioning move is commanded and the 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.

**2.1.12.101.2 Arguments****2.1.12.102.3.1 Input**

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

**2.1.12.103.4.2 Output**

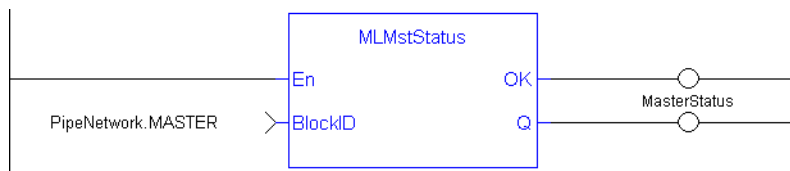
<b>OK</b>	<b>Description</b>	Returns true when function successfully executes
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Default (.Q)</b>	<b>Description</b>	Returns the status of the generator
	<b>Data type</b>	DINT
	<b>Unit</b>	n/a

**2.1.12.104.5 Example**

**2.1.12.105.6.1 Structured Text**

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

**2.1.12.106.7.2 Ladder Diagram**



**2.1.12.107.8.3 Function Block Diagram**



**2.1.12.108 MLMstWriteAccel** Pipe Network ✓**2.1.12.109.1 Description**

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

**2.1.12.110.2 Arguments****2.1.12.111.3.1 Input**

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

**2.1.12.112.4.2 Output**

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

**2.1.12.113.5 Related Functions**

["MLMstAbs"](#) (p. 287)

["MLMstRel"](#) (p. 305)

["MLMstWriteSpeed"](#) (p. 317)

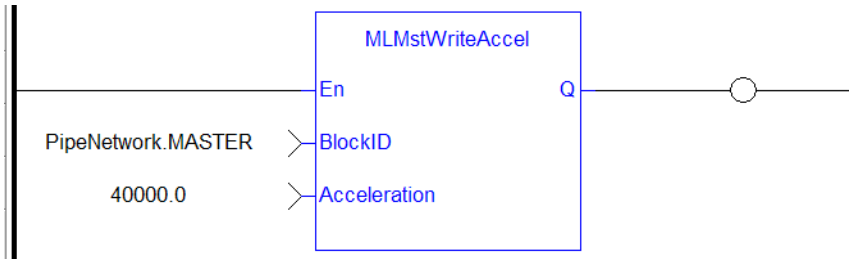
["MLMstWriteDecel"](#) (p. 313)

**2.1.12.114.6 Example**

### 2.1.12.115.7.1 Structured Text

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

### 2.1.12.116.8.2 Ladder Diagram



### 2.1.12.117.9.3 Function Block Diagram





**2.1.12.118 MLMstWriteDecel** Pipe Network ✓**2.1.12.119.1 Description**

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

**2.1.12.120.2 Arguments****2.1.12.121.3.1 Input**

<b>EN</b>	<b>Description</b>	Enables FB to be executed
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Block ID</b>	<b>Description</b>	ID name of the Master Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Deceleration value
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a> /sec <sup>2</sup>
	<b>Default</b>	—

**2.1.12.122.4.2 Output**

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

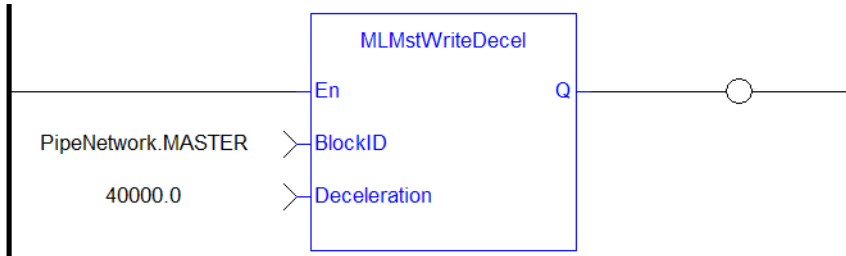
**2.1.12.123.5 Related Functions**

["MLMstWriteSpeed"](#) (p. 317)

**2.1.12.124.6 Example****2.1.12.125.7.1 Structured Text**

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

### 2.1.12.126.8.2 Ladder Diagram



### 2.1.12.127.9.3 Function Block Diagram



**2.1.12.128 MLMstWriteInitPos** **2.1.12.129.1 Description**

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

**2.1.12.130.2 Arguments****2.1.12.131.3.1 Input**

<b>EN</b>	<b>Description</b>	Enables FB to be executed
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Block ID</b>	<b>Description</b>	ID name of the Master Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Position</b>	<b>Description</b>	Initial position
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—

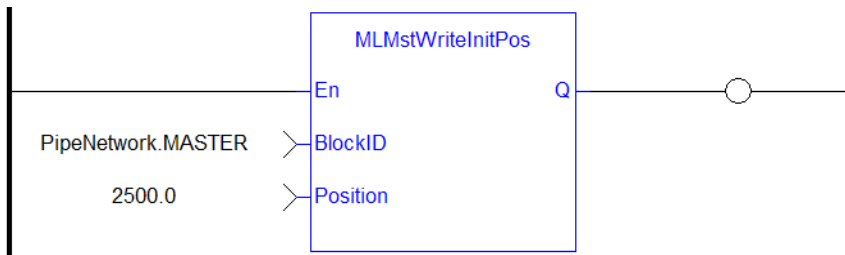
**2.1.12.132.4.2 Output**

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

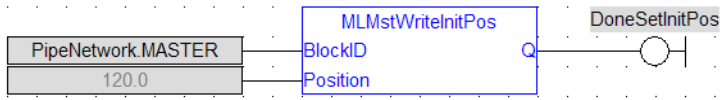
**2.1.12.133.5 Example****2.1.12.134.6.1 Structured Text**

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

**2.1.12.135.7.2 Ladder Diagram**



### 2.1.12.136.8.3 Function Block Diagram



**2.1.12.137 MLMstWriteSpeed** **2.1.12.138.1 Description**

Set the speed of motion for the "MLMstAbs" (p. 287) and "MLMstRel" (p. 305) blocks. Returns TRUE if the function succeeded. This function does not generate any motion.

**2.1.12.139.2 Arguments****2.1.12.140.3.1 Input**

<b>EN</b>	<b>Description</b>	Enables FB to be executed
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Block ID</b>	<b>Description</b>	ID name of the Master Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Speed</b>	<b>Description</b>	Speed of the motion
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a> /sec
	<b>Default</b>	—

**2.1.12.141.4.2 Output**

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

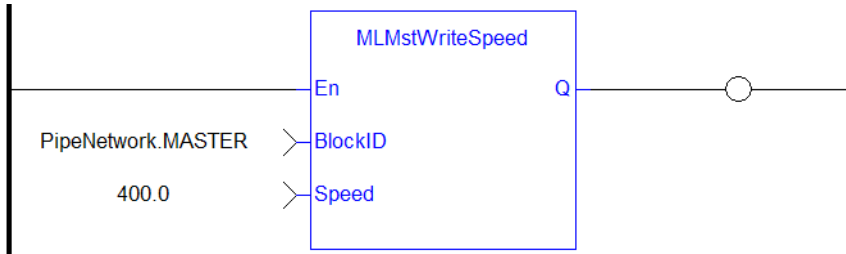
**2.1.12.142.5 Related Functions**

"MLMstWriteDecel" (p. 313)

**2.1.12.143.6 Example****2.1.12.144.7.1 Structured Text**

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

### 2.1.12.145.8.2 Ladder Diagram



### 2.1.12.146.9.3 Function Block Diagram



## 2.1.13 Motion Library - Phaser

### TIP

- For an example of Phaser functions, see ["Usage example of Phaser Functions"](#) (p. 319)

Names	Description	Return type
<a href="#">"MLPhaInit"</a> (p. 320)	Initializes a phaser Pipe Block	BOOL
<a href="#">"MLPhaReadPhase"</a> (p. 324)	Gets the phase value of a phaser block	None
<a href="#">"MLPhaReadSlope"</a> (p. 325)	Gets the phase slope value of a phaser block	None
<a href="#">"MLPhaWritePhase"</a> (p. 326)	Sets the phase value of a phaser block	BOOL
<a href="#">"MLPhaWriteSlope"</a> (p. 328)	Sets the phase slope value of a phaser block	BOOL
<a href="#">"MLPhaReadActPhase"</a> (p. 323)	Get the actual phase value of a phaser block.	LREAL

### TIP

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

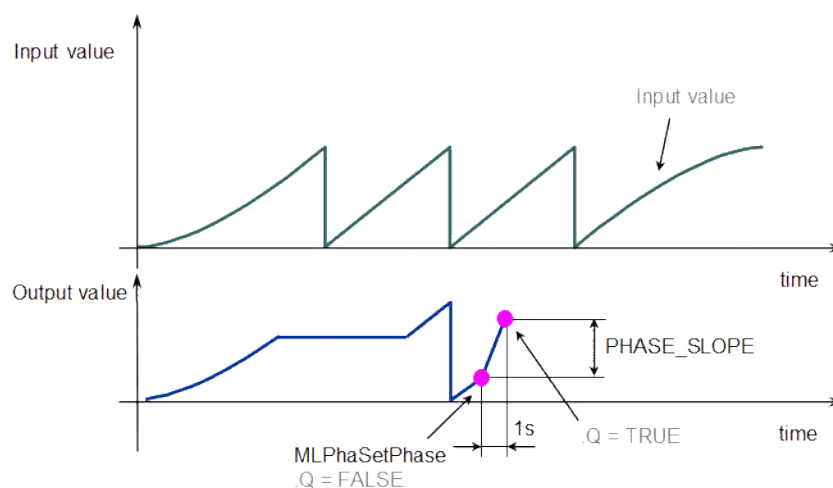
When executing, the phaser block is in one of three states: **Standby**, **Changing phase**, or **Applying phase**.

<b>Standby</b>	Entered when the Block is initialized. Exits to the State "Changing Phase" when the Phase value is changed via the <a href="#">"MLPhaWritePhase"</a> (p. 326) command
<b>Changing Phase</b>	Entered when a new value is programmed, and exits to "Applying phase" state when the programmed phase offset is reached. The current Phase offset value is slewed to the new phase offset by the amount of the slew value.
<b>Applying Phase</b>	Entered when the programmed Phase value is reached. Exits to the Changing phase state whenever a new value is programmed via the <a href="#">"MLPhaWritePhase"</a> (p. 326) function changes the Phase Offset target.

### 2.1.13.1 Usage example of Phaser Functions

You can call ["MLPhaWritePhase"](#) (p. 326) function to modify the Phase value..

You can call ["MLPhaWriteSlope"](#) (p. 328) to modify the rate of change of phase, or slope, applied when the Phase value is changed.



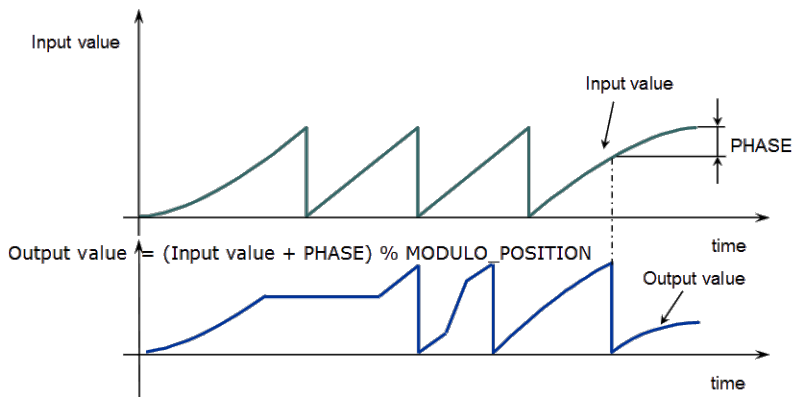


Figure 1-39: Phaser Functions Usage

**NOTE**

**% MODULO\_POSITION** is in the equation to take into account the modulo (periodicity) of the value.

**2.1.13.2 MLPalnit** Pipe Network ✓

**2.1.13.3.1 Description**

Initializes a phaser Pipe Block. Returns TRUE if the function succeeded.

This function block is automatically called by the Function PipeNetwork(MLPN\_CREATE\_OBJECTS) if a Phaser Block is added to the Pipe Network, with user-defined settings entered in the Pipe Blocks Properties screen.

The Phaser Pipe Block is assigned a Name, OUTPUT\_PERIOD, PHASE, PHASE\_SLOPE\_TYPE, and STANDBY\_VALUE.

**2.1.13.4.2 Arguments**

**2.1.13.5.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID Name of a Phaser function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ModuloPosition</b>	<b>Description</b>	Rollover Position of the Phaser block
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	360.0
<b>Phase</b>	<b>Description</b>	Amount of Phase adjustment
	<b>Data type</b>	LREAL



	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	0.0
<b>UseUserSlope</b>	<b>Description</b>	Setting determines if Max Slope or user-defined slope is used
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	Max Slope
<b>PhaseSlope</b>	<b>Description</b>	User-defined slope for making the phase adjustment
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	0.0
<b>StandbyValue</b>	<b>Description</b>	This value is output from the Phaser Block, when the pipe is active, until the " <a href="#">MLPhaWritePhase</a> " (p. 326) function is executed.
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	0.0

#### 2.1.13.6.4.2 Output

<b>Default (.Q)</b>	<b>Description</b>	Returns True if the function block is successfully executing
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

#### 2.1.13.7.5 Related Functions

["MLPhaReadPhase"](#) (p. 324)

["MLPhaReadSlope"](#) (p. 325)

MLPhalnit

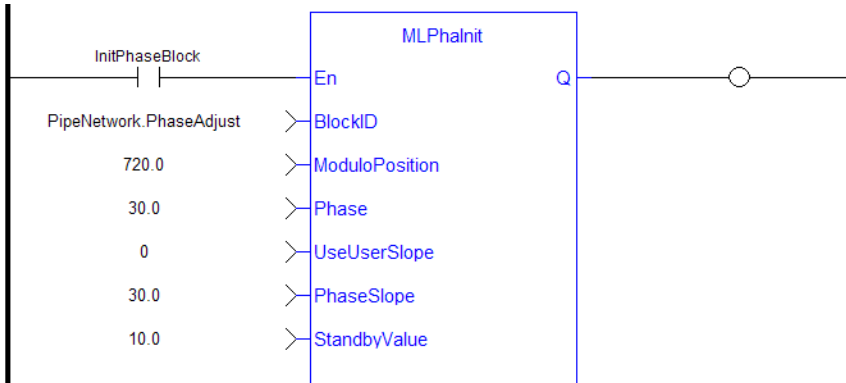
["MLPhaWritePhase"](#) (p. 326)

#### 2.1.13.8.6 Example

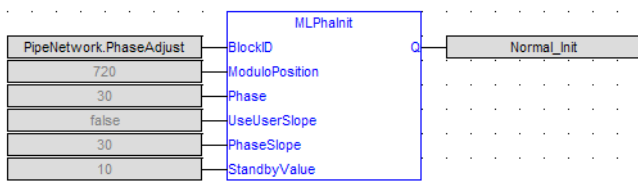
#### 2.1.13.9.7.1 Structured Text

```
//Initialize a Phaser Pipe Block named "PhaseAdjust" to a
Modulo of 720, phase offset value of 30, use the Max Slope
MLPhaInit( PipeNetwork.PhaseAdjust , 720, 30, false, 30 , 10 );
```

### 2.1.13.10.8.2 Ladder Diagram



### 2.1.13.11.9.3 Function Block Diagram



**2.1.13.12 MLPhaReadActPhase** Pipe Network ✓**2.1.13.13.1 Description**

Get the actual phase value of a phaser block.

If a "PHASE\_SLOPE\_USER" (refer to "[MLPhaReadSlope](#)" (p. 325) and "[MLPhaWriteSlope](#)" (p. 328)) value is being used, the new phase (refer to "[MLPhaWritePhase](#)" (p. 326)) isn't set immediately; the phase will be ramped with the slope value from the old phase value to the new phase value. MLPhaReadActPhase returns this ramping value.

["MLPhaReadPhase"](#) (p. 324) returns the new value and this also when the phaser is still ramping. If using max slope means no ramping MLPhaReadActPhase and MLPhaReadPhase return always the same value.

**2.1.13.14.2 Arguments****2.1.13.15.3.1 Input**

<b>Enable</b>	<b>Description</b>	
	<b>Data type</b>	
	<b>Range</b>	
	<b>Unit</b>	
	<b>Default</b>	
<b>BlockID</b>	<b>Description</b>	ID Name of a Phaser function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.13.16.4.2 Output**

<b>OK</b>	<b>Description</b>	
	<b>Data type</b>	
	<b>Unit</b>	
<b>Phase</b>	<b>Description</b>	
	<b>Data type</b>	LREAL
	<b>Unit</b>	

**2.1.13.17.5 Related Functions**

MLPhaReadPhase, MLPhaWritePhase, MLPhaReadSlope, MLPhaWriteSlope

**2.1.13.18 MLPhaReadPhase** Pipe Network ✓

**2.1.13.19.1 Description**

Get the phase value of a phaser block.

**2.1.13.20.2 Arguments**

**2.1.13.21.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID Name of a Phaser function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.13.22.4.2 Output**

<b>Phase</b>	<b>Data type</b>	LREAL
--------------	------------------	-------

**2.1.13.23.5 Example**

**2.1.13.24.6.1 Structured Text**

```
PresentPhase := MLPhaReadPhase ( PipeNetwork.PhaseAdjust );
```

**2.1.13.25.7.2 Ladder Diagram**



**2.1.13.26.8.3 Function Block Diagram**



**2.1.13.27 MLPhaReadSlope** Pipe Network ✓

**2.1.13.28.1 Description**

Get the phase slope value of a phaser block.

**2.1.13.29.2 Arguments**

**2.1.13.30.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID Name of a Phaser function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.13.31.4.2 Output**

<b>Slope</b>	<b>Description</b>	Present Slope value
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit/sec
	<b>Default</b>	Value defined in the setup of a Phaser Block within a Pipe Network. Depending on the phase slope type setting, it is the VALUE in "PHASE_SLOPE_USER", "PHASE" or the max slope.

**2.1.13.32.5 Related Functions**

[MLPhaReadSlope](#)

**2.1.13.33.6 Example**

**2.1.13.34.7.1 Structured Text**

```
PresentSlope :=MLPhaReadSlope( PipeNetwork.PhaseAdjust );
```

**2.1.13.35.8.2 Ladder Diagram**



**2.1.13.36.9.3 Function Block Diagram**



**2.1.13.37 MLPhaWritePhase** Pipe Network ✓**2.1.13.38.1 Description**

Set the phase value of a phaser block.

**2.1.13.39.2 Arguments****2.1.13.40.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID Name of a Phaser function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Phase</b>	<b>Description</b>	Phase value
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	Value defined in the setup of a phaser Block within a Pipe Network. It is in the "PHASE" field in the parameter tab

**2.1.13.41.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns True if the function block is successfully executing
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

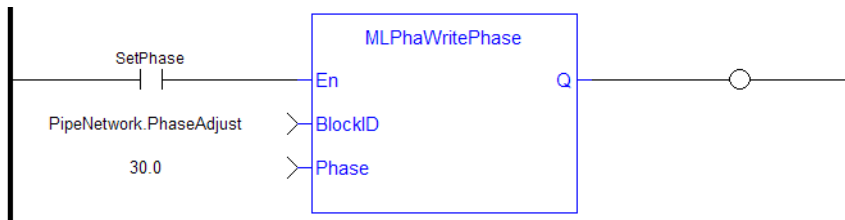
**2.1.13.42.5 Related Functions**

["MLPhaReadPhase" \(p. 324\)](#)

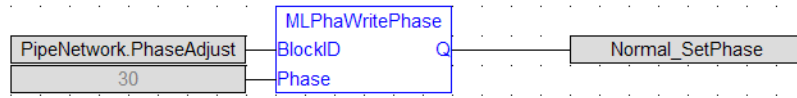
**2.1.13.43.6 Example****2.1.13.44.7.1 Structured Text**

```
MLPhaWritePhase( PipeNetwork.PhaseAdjust , 30 );
```

**2.1.13.45.8.2 Ladder Diagram**



**2.1.13.46.9.3 Function Block Diagram**



**2.1.13.47 MLPhaWriteSlope** Pipe Network ✓**2.1.13.48.1 Description**

Set the phase value of a phaser block. Returns TRUE if the function succeeded.

**2.1.13.49.2 Arguments****2.1.13.50.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID Name of a Phaser function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
	<b>Description</b>	Set slope of phase adjust
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	Value defined in the setup of a Phaser Block within a Pipe Network. Depending on the phase slope type setting, it is the VALUE in "PHASE_SLOPE_USER", "PHASE" or the max slope.

**2.1.13.51.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns True if the function block is successfully executing
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.13.52.5 Related Functions**

["MLPhaReadSlope" \(p. 325\)](#)

**2.1.13.53.6 Example****2.1.13.54.7.1 Structured Text**

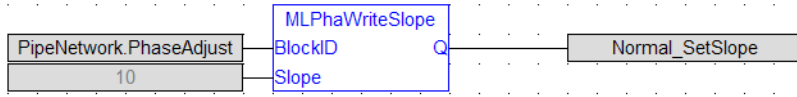
```
MLPhaWriteSlope ( PipeNetwork.PhaseAdjust , 10 );
```

**2.1.13.55.8.2 Ladder Diagram**





**2.1.13.56.9.3 Function Block Diagram**



## 2.1.14 Motion Library - PMP

Name	Description	Return type
"MLPmpAbs" (p. 330)	Moves to an Absolute Position	BOOL
"MLPmpForcePos" (p. 332)	Forces the specified position. Possible only when the block is <b>not</b> moving.	BOOL
"MLPmpInit" (p. 334)	Initializes a PMP object (Parabolic Motion Profile generator) with user-defined settings	BOOL
"MLPmpReadAccel" (p. 338)	Gets the Acceleration parameter of a PMP block	None
"MLPmpReadFstSpd" (p. 339)	Gets the FirstTravelSpeed parameter of a PMP block	None
"MLPmpReadInitPos" (p. 341)	Gets the InitialPosition parameter of a PMP block	None
"MLPmpReadJerk" (p. 342)	Gets the Jerk parameter of a PMP block	None
"MLPmpReadLstSpd" (p. 343)	Gets the LastTravelSpeed parameter of a PMP block	None
"MLPmpRel" (p. 345)	Does two subsequent relative moves	BOOL
"MLPmpRun" (p. 347)	Jog the generator at the specified speed	BOOL
"MLPmpStatus" (p. 349)	Returns the status of the PMP block generator	None
"MLPmpWriteAccel" (p. 351)	Sets the acceleration parameter of a PMP block	BOOL
"MLPmpWriteFstSpd" (p. 353)	Sets the FirstTravelSpeed parameter of a PMP block	BOOL
"MLPmpWriteJerk" (p. 355)	Sets the jerk parameter of a PMP block	BOOL
"MLPmpWriteLstSpd" (p. 357)	Sets the LastTravelSpeed parameter of a PMP block	BOOL

### 2.1.14.1 MLPmpAbs

#### 2.1.14.2.1 Description

Move to an Absolute Position using a parabolic acceleration profile. The FIRST\_TRAVEL\_SPEED is used as the velocity for the motion. JERK determines the level of parabolic acceleration. Returns TRUE if the function succeeded.

#### 2.1.14.3.2 Arguments

##### 2.1.14.4.3.1 Input

<b>BlockID</b>	<b>Description</b>	ID Name of the PMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Position</b>	<b>Description</b>	Absolute Position of motor/load to be at after this FB is complete
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—

**2.1.14.5.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns True if the function block is successfully executing
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.14.6.5 Related Functions**

"MLPmpWriteAccel" (p. 351)

"MLPmpWriteJerk" (p. 355)

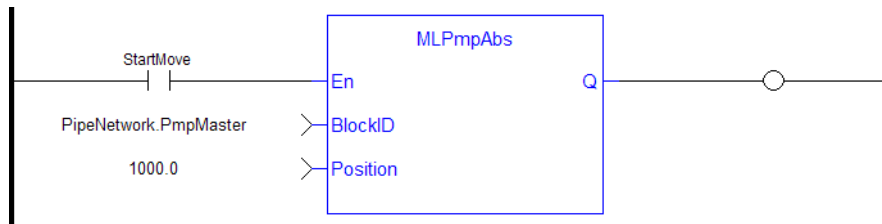
"MLPmpWriteFstSpd" (p. 353)

**2.1.14.7.6 Example**

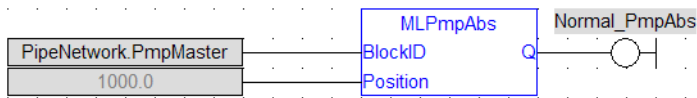
**2.1.14.8.7.1 Structured Text**

```
MLPmpAbs ( PipeNetwork.PmpMaster, 1000.0 ) ;
```

**2.1.14.9.8.2 Ladder Diagram**



**2.1.14.10.9.3 Function Block Diagram**



**2.1.14.11 MLPmpForcePos** **2.1.14.12.1 Description**

Forces the position of a PMP Block to a specified position. This block can only be executed when motion is not occurring. It can be used to force the PMP starting position to the desired values from which to start motion.

**2.1.14.13.2 Arguments****2.1.14.14.3.1 Input**

<b>EN</b>	<b>Description</b>	Enables FB to be executed
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Block ID</b>	<b>Description</b>	ID name of the PMP Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Position</b>	<b>Description</b>	Defines the PMP starting position when the motion starts
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—

**2.1.14.15.4.2 Output**

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

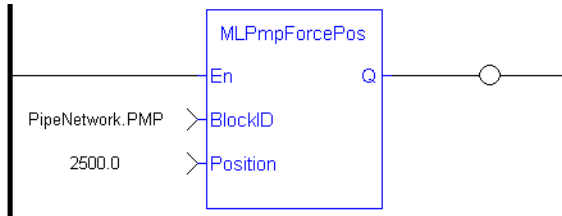
**2.1.14.16.5 Related Functions**

["MLPmpReadInitPos" \(p. 341\)](#)

**2.1.14.17.6 Example****2.1.14.18.7.1 Structured Text**

```
MLPmpForcePos ( PipeNetwork.PMP, 2500.0 );
```

### 2.1.14.19.8.2 Ladder Diagram



#### NOTE

You must use a [pulse contact](#) to start the FB

### 2.1.14.20.9.3 Function Block Diagram



**2.1.14.21 MLPmplnit** **2.1.14.22.1 Description**

Initializes a Pmp Block for use in a PLC Program. This function block is automatically called by the Function PipeNetwork(MLPN\_CREATE\_OBJECTS) if a Pmp Block is added to the Pipe Network, with user-defined settings entered in the Pipe Blocks Properties screen.

The Pmp Pipe Block is assigned a Name, SAMPLING\_PERIOD, MODULO\_POSITION, FIRST\_TRAVEL\_SPEED, LAST\_TRAVEL\_SPEED, ACCELERATION, JERK, and INITIAL Position. Some of these parameters can be changes in an application program using other MLPmp function blocks

A MLPmpRel function block is used to make a bi directional motion. First movement in one direction, then a return motion back to the initial position. A MLPmpAbs function block is use to move one direction to an absolute position.

**NOTE**

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

**2.1.14.23.2 Arguments****2.1.14.24.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID Name of a PMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ModuloPosition</b>	<b>Description</b>	Modulo Position for cyclic motion systems expressed in user logical units (Position Rollover Value)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	360.0
<b>Period</b>	<b>Description</b>	Sampling period of the generator expressed according to the update cycle (e.g. 2.0 means the sampling is done once every 2 cycles)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	1.0
<b>FirstTravelSpeed</b>	<b>Description</b>	First Travel Speed of the motion

	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	100.0
<b>LastTravelSpeed</b>	<b>Description</b>	Last Travel Speed of the motion
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	100.0
<b>Acceleration</b>	<b>Description</b>	Acceleration of the Pmp block motion
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	1000.0
<b>Jerk</b>	<b>Description</b>	Jerk
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	0
<b>InitialPosition</b>	<b>Description</b>	Initial Position of the Pmp block when the Pipe Network is start up
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	0
<b>Modulo</b>	<b>Description</b>	The available modes are Modulo (True) or No modulo (False)
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 2.1.14.25.4.2 Output

<b>Default (.Q)</b>	<b>Description</b>	Returns True if the function block is successfully executing
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

### 2.1.14.26.5 Related Functions

- "MLPmpReadAccel" (p. 338)
- "MLPmpReadFstSpd" (p. 339)
- "MLPmpReadInitPos" (p. 341)
- "MLPmpReadJerk" (p. 342)
- "MLPmpReadLstSpd" (p. 343)

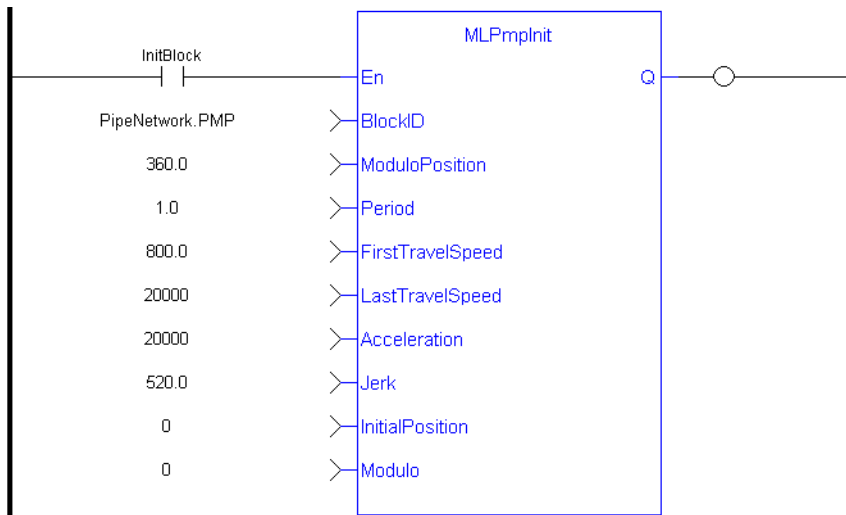
### 2.1.14.27.6 Example

#### 2.1.14.28.7.1 Structured Text

```

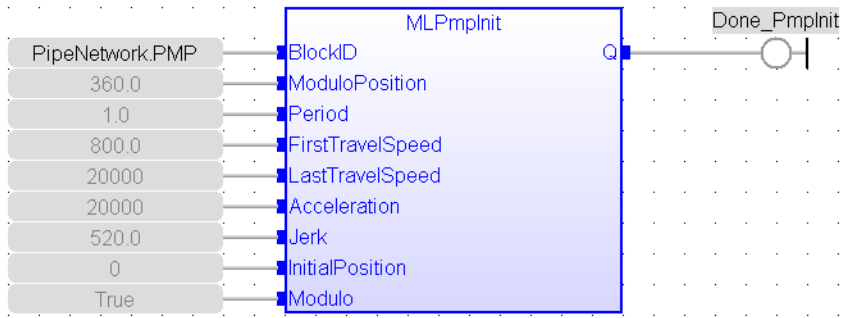
//Initialize a PMP Pipe Block named "PMP" to a modulo roll over
of 360, motion generator sample period of 1, First Travel Speed
of 800.0, Second Travel Speed of 20000.0, Accel of 20000.0, Jerk
of 520.0, Initial position of 0.0
MLPmpInit( PipeNetwork.Pmp , 360.0, 1.0, 800.0, 20000.0,
20000.0, 520.0, 0, true ) ;
    
```

### 2.1.14.29.8.2 Ladder Diagram



### 2.1.14.30.9.3 Function Block Diagram





**2.1.14.31 MLPmpReadAccel** Pipe Network ✓

**2.1.14.32.1 Description**

Get the Acceleration parameter of a PMP block used in both the MLPmpAbs and MLPmpRel function block.

**2.1.14.33.2 Arguments**

**2.1.14.34.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID Name of the PMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.14.35.4.2 Output**

<b>Acceleration</b>	<b>Description</b>	Present Acceleration of the PMP PipeNetwork Function Block
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	Value defined PMP Block when creating a Pipe Network. It is in the "ACCELERATION" field in the parameter tab.

**2.1.14.36.5 Related Functions**

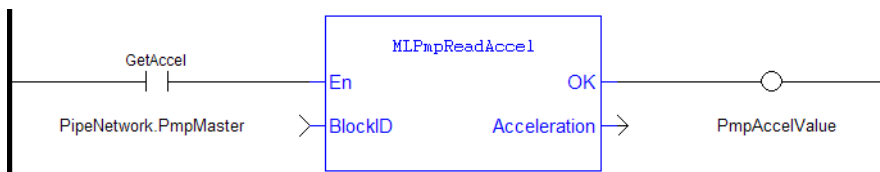
- ["MLPmpReadFstSpd" \(p. 339\)](#)
- ["MLPmpReadInitPos" \(p. 341\)](#)
- ["MLPmpReadJerk" \(p. 342\)](#)
- ["MLPmpReadLstSpd" \(p. 343\)](#)

**2.1.14.37.6 Example**

**2.1.14.38.7.1 Structured Text**

```
PmpAccelValue := MLPmpReadAccel ( PipeNetwork.PmpMaster ) ;
```

**2.1.14.39.8.2 Ladder Diagram**



**2.1.14.40.9.3 Function Block Diagram**



**2.1.14.41 MLPmpReadFstSpd** Pipe Network ✓**2.1.14.42.1 Descriptions**

Get the FirstTravelSpeed parameter of a PMP block. This parameter is used as the first of 2 speeds in an MLPmpRel Motion Block. It is also used as the speed in an MLPmpAbs Motion Block.

**2.1.14.43.2 Arguments****2.1.14.44.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID Name of a PMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.14.45.4.2 Output**

<b>FirstTravelSpeed</b>	<b>Description</b>	Present first travel velocity of the PMP PipeNetwork Function Block
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit/sec
	<b>Default</b>	Value defined in the setup of a PMP Block within a Pipe Network. It is in the "FIRST_TRAVEL_SPEED" field in the parameter tab

**2.1.14.46.5 Related Functions**

["MLPmpReadAccel" \(p. 338\)](#)

[MLPmpReadFstSpd](#)

["MLPmpReadInitPos" \(p. 341\)](#)

["MLPmpReadJerk" \(p. 342\)](#)

["MLPmpReadLstSpd" \(p. 343\)](#)

["MLPmpWriteLstSpd" \(p. 357\)](#)

**2.1.14.47.6 Example****2.1.14.48.7.1 Structured Text**

```
FirstSpeedValue := MLPmpReadFstSpd( PipeNetwork.PmpMaster ) ;
```

**2.1.14.49.8.2 Ladder Diagram**



### 2.1.14.50.9.3 Function Block Diagram



**2.1.14.51 MLPmpReadInitPos** Pipe Network ✓

**2.1.14.52.1 Description**

Get the Initial Position parameter of a PMP block. This value is the position the Pmpblock starts at when the Pipe Network is enabled. This position can be set when adding a Pmp Block to a Pipe Network and defining the parameters for that block.

**2.1.14.53.2 Arguments**

**2.1.14.54.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID Name of a PMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.14.55.4.2 Output**

<b>InitialPosition</b>	<b>Description</b>	Present Initial Position of the PMP PipeNetwork Function Block
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit
	<b>Default</b>	Value defined in the setup of a PMP Block within a Pipe Network. It is in the "INITIAL_POSITION" field in the parameter tab.

**2.1.14.56.5 Related Functions**

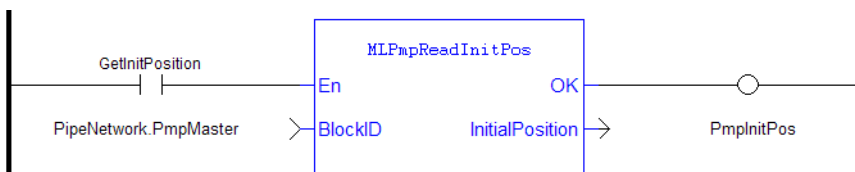
"MLPmpInit" (p. 334)

**2.1.14.57.6 Example**

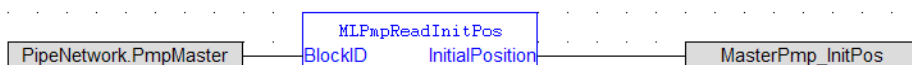
**2.1.14.58.7.1 Structured Text**

```
PmpInitPos := MLPmpReadInitPos ( PipeNetwork.PmpMaster ) ;
```

**2.1.14.59.8.2 Ladder Diagram**



**2.1.14.60.9.3 Function Block Diagram**



**2.1.14.61 MLPmpReadJerk** Pipe Network ✓

**2.1.14.62.1 Description**

Get the Jerk parameter of a PMP block used in both the MLPmpAbs and MLPmpRel function block.

**2.1.14.63.2 Arguments**

**2.1.14.64.3.1 Input**

BlockID	Description	ID Name of a PMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

**2.1.14.65.4.2 Output**

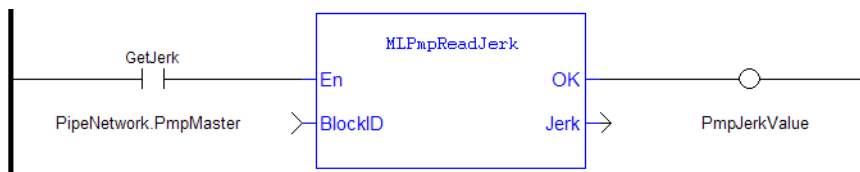
Jerk	Description	Jerk of the PMP PipeNetwork Function Block
	Data type	LREAL
	Unit	User unit/sec <sup>3</sup>
	Default	Value defined in the setup of a PMP Block within a Pipe Network. It is in the “JERK” field in the parameter tab.

**2.1.14.66.5 Example**

**2.1.14.67.6.1 Structured Text**

```
PmpJerkValue := MLPmpReadJerk( PipeNetwork.PmpMaster ) ;
```

**2.1.14.68.7.2 Ladder Diagram**



**2.1.14.69.8.3 Function Block Diagram**



**2.1.14.70 MLPmpReadLstSpd** Pipe Network ✓**2.1.14.71.1 Description**

Get the LastTravelSpeed parameter of a PMP block used in the MLPmpRel function block.

**2.1.14.72.2 Arguments****2.1.14.73.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID Name of a PMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.14.74.4.2 Output**

<b>LastTravelSpeed</b>	<b>Description</b>	Last Travel Speed of the PMP PipeNetwork Function Block
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit/sec
	<b>Default</b>	Value defined in the setup of a PMP Block within a Pipe Network. It is in the "LAST_TRAVEL_SPEED" field in the parameter tab.

**2.1.14.75.5 Related Functions**

"MLPmpReadAccel" (p. 338)

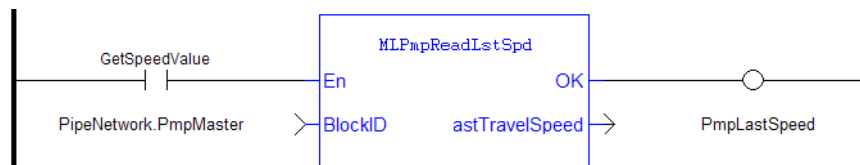
"MLPmpReadFstSpd" (p. 339)

"MLPmpReadInitPos" (p. 341)

"MLPmpReadJerk" (p. 342)

**2.1.14.76.6 Example****2.1.14.77.7.1 Structured Text**

```
PmpLastSpeed := MLPmpReadLstSpd( PipeNetwork.PmpMaster ) ;
```

**2.1.14.78.8.2 Ladder Diagram****2.1.14.79.9.3 Function Block Diagram**





## 2.1.14.80 MLPmpRel Pipe Network ✓

### 2.1.14.81.1 Description

This function is used to perform two subsequent relative moves. Using the MLPmpRel function block, the PMP Generator is capable of producing forward-backward motions with a non-stop, jerk-free transition through zero speed (see Figure below). This feature is frequently useful for linear axes which must move back and forward without any pause at one end.

This function can also be used to do a single relative move, ending in zero speed, by setting the **DeltaSecond** argument to zero (0.0). If it is done, for the controlling speed to be the first move, the “Last\_Travel\_Speed” parameter has to be set equal to or greater than the “First\_Travel\_Speed” parameter.

In general, the slower of the two “Speeds” is utilized to optimize the S-curve behavior for the move whether it is a 2 or 1 delta move.

If the DeltaSecond argument is non-zero, it must have the opposite sign than the sign of the DeltaFirst argument.

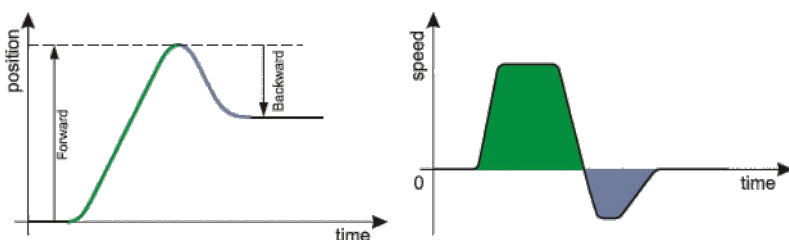


Figure 1-40: PMP Generator Forward & Backward Motion Profile

### 2.1.14.82.2 Arguments

#### 2.1.14.83.3.1 Input

<b>BlockID</b>	<b>Description</b>	ID Name of the PMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>DeltaFirst</b>	<b>Description</b>	Length of first Move
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	Value defined in the setup of a PMP Block within a Pipe Network. It is the “FIRST_TRAVEL_SPEED” field in the parameter tab.
<b>DeltaSecond</b>	<b>Description</b>	Length of second (return) Move
	<b>Data type</b>	LREAL
	<b>Range</b>	—

<b>Unit</b>	User unit
<b>Default</b>	Value defined in the setup of a PMP Block within a Pipe Network. It is the "LAST_TRAVEL_SPEED" field in the parameter tab.

**2.1.14.84.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns True if the MLPmRel successfully completed
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.14.85.5 Related Functions**

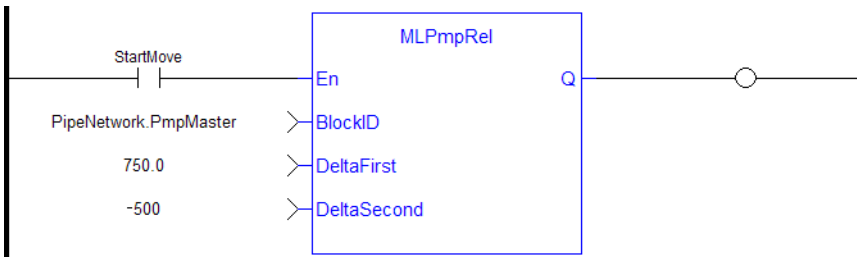
- ["MLPmpWriteAccel" \(p. 351\)](#)
- ["MLPmpWriteFstSpd" \(p. 353\)](#)
- ["MLPmpWriteJerk" \(p. 355\)](#)
- ["MLPmpWriteLstSpd" \(p. 357\)](#)

**2.1.14.86.6 Example**

**2.1.14.87.7.1 Structured Text**

```
//Execute a Relative move on a PMP Block name "PmpMaster" with a
First Travel Speed of 750.0, Second Travel Speed of -500,
MLPmpRel( PipeNetwork.PmpMaster, 750 , -500);
```

**2.1.14.88.8.2 Ladder Diagram**



**2.1.14.89.9.3 Function Block Diagram**



### 2.1.14.90 MLPmpRun Pipe Network ✓

#### 2.1.14.91.1 Description

Jog the generator at the requested speed.

#### 2.1.14.92.2 Arguments

##### 2.1.14.93.3.1 Input

<b>EN</b>	<b>Description</b>	Enables FB to be executed. Is only recognized if the PMP generator is Idle or at constant velocity as determined from the " <a href="#">MLPmpStatus</a> " (p. 349) function.
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>BlockID</b>	<b>Description</b>	ID Name of the PMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Speed</b>	<b>Description</b>	The desired rate at which to Jog. If the speed is 0.0 User Units / second the PMP block decelerates to zero speed and switches to the Idle state (0).
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	<a href="#">User unit</a> /sec
	<b>Default</b>	—

##### 2.1.14.94.4.2 Output

<b>Default (.Q)</b>	<b>Description</b>	Returns True if the MLPmpRun successfully completed.
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

#### 2.1.14.95.5 Related Functions

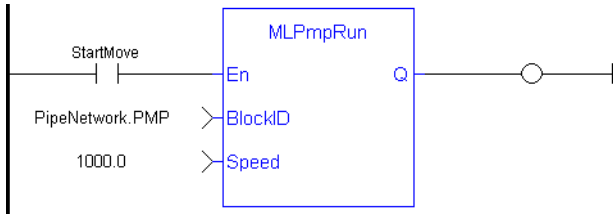
["MLPmpStatus"](#) (p. 349)

#### 2.1.14.96.6 Example

#### 2.1.14.97.7.1 Structured Text

```
//Execute a Relative move on a PMP Block name "PmpMaster" with a
Jog speed of 1000.0
MLPmpRun( PipeNetwork.PmpMaster, 1000.0 ) ;
```

### 2.1.14.98.8.2 Ladder Diagram



### 2.1.14.99.9.3 Function Block Diagram



**2.1.14.100 MLPmpStatus** Pipe Network ✓**2.1.14.101.1 Description**

Returns the status of the PMP block generator.

**2.1.14.102.2 Arguments****2.1.14.103.3.1 Input**

<b>EN</b>	<b>Description</b>	Enables FB to be executed.
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>BlockID</b>	<b>Description</b>	ID Name of the PMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.14.104.4.2 Output**

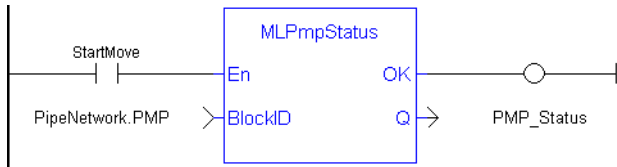
<b>OK</b>	<b>Description</b>	Returns true when function successfully executes								
	<b>Data type</b>	BOOL								
	<b>Unit</b>	n/a								
<b>Default (.Q)</b>	<b>Description</b>	Returns the status of the PMP block generator								
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Indicates that the PMP block is idle. No command is currently running in the generator. It can be used to determine that a previous move is complete.</td> </tr> <tr> <td>1</td> <td>Indicates that the PMP block is either accelerating to a position or speed, or is decelerating to a position or speed.</td> </tr> <tr> <td>2</td> <td>Indicates that the PMP block is running at a constant speed.</td> </tr> </tbody> </table>	Value	Description	0	Indicates that the PMP block is idle. No command is currently running in the generator. It can be used to determine that a previous move is complete.	1	Indicates that the PMP block is either accelerating to a position or speed, or is decelerating to a position or speed.	2	Indicates that the PMP block is running at a constant speed.
Value	Description									
0	Indicates that the PMP block is idle. No command is currently running in the generator. It can be used to determine that a previous move is complete.									
1	Indicates that the PMP block is either accelerating to a position or speed, or is decelerating to a position or speed.									
2	Indicates that the PMP block is running at a constant speed.									
	<b>Data type</b>	DINT								
	<b>Unit</b>	n/a								

**2.1.14.105.5 Example**

### 2.1.14.106.6.1 Structured Text

```
PMP_Status := MLPmpStatus ( PipeNetwork.PmpMaster ) ;
Done :=TRUE;
```

### 2.1.14.107.7.2 Ladder Diagram



### 2.1.14.108.8.3 Function Block Diagram



**2.1.14.109 MLPmpWriteAccel** Pipe Network ✓**2.1.14.110.1 Description**

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

Acceleration can also be set when adding a Pmp Block to a Pipe Network and defining the parameters for that block.

**2.1.14.111.2 Arguments****2.1.14.112.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID Name of the PMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Acceleration Value
	<b>Data type</b>	LREAL
	<b>Range</b>	> 0
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	Value defined in the setup of a PMP Block within a Pipe Network. It is the "ACCELERATION" field the parameter tab.

**2.1.14.113.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns True if the function block is successfully executing
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.14.114.5 Related Functions**

["MLPmpAbs" \(p. 330\)](#)

["MLPmpRel" \(p. 345\)](#)

["MLPmpWriteFstSpd" \(p. 353\)](#)

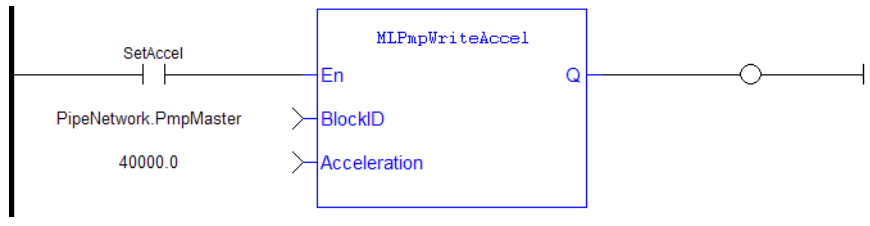
["MLPmpWriteJerk" \(p. 355\)](#)

["MLPmpWriteLstSpd" \(p. 357\)](#)

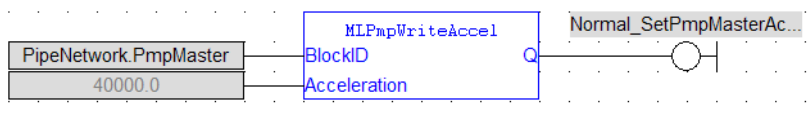
**2.1.14.115.6 Example****2.1.14.116.7.1 Structured Text**

```
MLPmpWriteAccel ( PipeNetwork.PmpMaster, 40000.0 ) ;
```

### 2.1.14.117.8.2 Ladder Diagram



### 2.1.14.118.9.3 Function Block Diagram





**2.1.14.119 MLPmpWriteFstSpd** **2.1.14.120.1 Description**

Set the FirstTravelSpeed parameter of a PMP block. Returns TRUE if the function succeeded. FirstTravelSpeed can also be set when adding a Pmp Block to a Pipe Network and defining the parameters for that block.

**2.1.14.121.2 Arguments****2.1.14.122.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID Name of the PMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>First Travel Speed</b>	<b>Description</b>	First Travel Speed Value
	<b>Data type</b>	LREAL
	<b>Range</b>	> 0
	<b>Unit</b>	User unit/sec
	<b>Default</b>	Value defined in the setup of a PMP Block within a Pipe Network. It is the "FIRST_TRAVEL_SPEED" field in the parameter tab.

**2.1.14.123.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns True if the function block is successfully executing
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.14.124.5 Related Functions**

["MLPmpAbs" \(p. 330\)](#)

["MLPmpRel" \(p. 345\)](#)

["MLPmpWriteAccel" \(p. 351\)](#)

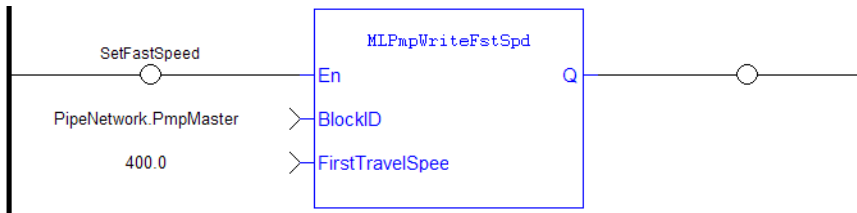
["MLPmpWriteJerk" \(p. 355\)](#)

["MLPmpWriteLstSpd" \(p. 357\)](#)

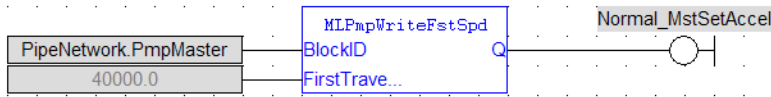
**2.1.14.125.6 Example****2.1.14.126.7.1 Structured Text**

```
MLPmpWriteFstSpd( PipeNetwork.PmpMaster, 300.0 ) ;
```

### 2.1.14.127.8.2 Ladder Diagram



### 2.1.14.128.9.3 Function Block Diagram



**2.1.14.129 MLPmpWriteJerk** **2.1.14.130.1 Description**

Set the jerk parameter of a PMP block. Returns TRUE if the function succeeded. Jerk can also be set when adding a Pmp Block to a Pipe Network and defining the parameters for that block.

**2.1.14.131.2 Arguments****2.1.14.132.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID Name of the PMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Jerk Value
	<b>Data type</b>	LREAL
	<b>Range</b>	> 0
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	Value defined in the setup of a PMP Block within a Pipe Network. It is the "JERK" field in the parameter tab.

**2.1.14.133.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns True if the function block is successfully executing
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.14.134.5 Related Functions**

["MLPmpAbs" \(p. 330\)](#)

["MLPmpReadJerk" \(p. 342\)](#)

["MLPmpRel" \(p. 345\)](#)

["MLPmpWriteAccel" \(p. 351\)](#)

["MLPmpWriteFstSpd" \(p. 353\)](#)

["MLPmpWriteLstSpd" \(p. 357\)](#)

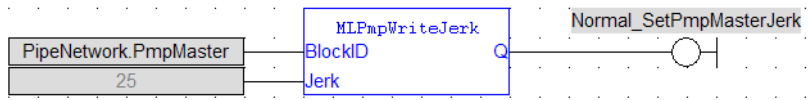
**2.1.14.135.6 Example****2.1.14.136.7.1 Structured Text**

```
MLPmpWriteJerk( PipeNetwork.PmpMaster, 15.0 ) ;
```

### 2.1.14.137.8.2 Ladder Diagram



### 2.1.14.138.9.3 Function Block Diagram



**2.1.14.139 MLPmpWriteLstSpd** **2.1.14.140.1 Description**

Set the LastTravelSpeed parameter of a PMP block. Returns TRUE if the function succeeded. Last Travel Speed can also be set when adding a Pmp Block to a Pipe Network and defining the parameters for that block.

**2.1.14.141.2 Arguments****2.1.14.142.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID Name of the PMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Last Speed</b>	<b>Description</b>	Last Travel Speed Value
	<b>Data type</b>	LREAL
	<b>Range</b>	> 0
	<b>Unit</b>	User unit/sec
	<b>Default</b>	Value defined in the setup of a PMP Block within a Pipe Network. It is in the "LAST_TRAVEL_SPEED" field in the parameter tab.

**2.1.14.143.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns True if the function block is successfully executing
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.14.144.5 Related Functions**

["MLPmpAbs" \(p. 330\)](#)

["MLPmpReadLstSpd" \(p. 343\)](#)

["MLPmpRel" \(p. 345\)](#)

["MLPmpWriteAccel" \(p. 351\)](#)

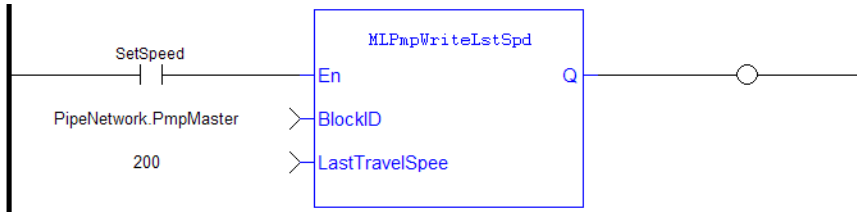
["MLPmpWriteFstSpd" \(p. 353\)](#)

["MLPmpWriteJerk" \(p. 355\)](#)

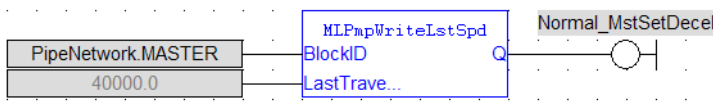
**2.1.14.145.6 Example****2.1.14.146.7.1 Structured Text**

```
MLPmpWriteLstSpd( PipeNetwork.PmpMaster, 650 ) ;
```

### 2.1.14.147.8.2 Ladder Diagram



### 2.1.14.148.9.3 Function Block Diagram



## 2.1.15 Motion Library - Sampler

Name	Description	Return type
"MLSmpConnect" (p. 359)	Connects a sampler block to a pipe network axis or pipe block	BOOL
"MLSmpConECAT" (p. 361)	Connects a sampler block to the specified CoE object in a PDO	BOOL
"MLSmpConPLCAxis" (p. 366)	Connects a sampler block to a PLCopen axis variable	BOOL
"MLSmpConPNAxis" (p. 368)	Connects a sampler block to a Pipe Network axis variable	BOOL
"MLSmpInit" (p. 363)	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.

### 2.1.15.1 MLSmpConnect

#### 2.1.15.2.1 Description

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

#### 2.1.15.3.2.1 Related Function Blocks

"MLSmpConECAT" (p. 361), "MLSmpInit" (p. 363), "MLSmpConPNAxis" (p. 368), "MLSmpConPLCAxis" (p. 366)

#### 2.1.15.4.3 Arguments

##### 2.1.15.5.4.1 Input

<b>BlockID</b>	<b>Description</b>	ID Name of the SMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>PipeBlockID</b>	<b>Description</b>	ID Name of the Pipe Block the sampler is connected to
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—

##### 2.1.15.6.5.2 Output

<b>Default (.Q)</b>	<b>Description</b>	Returns True if the Sampler is connected.
---------------------	--------------------	---

<b>Data type</b>	BOOL
<b>Unit</b>	n/a

**2.1.15.7.6.3 Return Type**

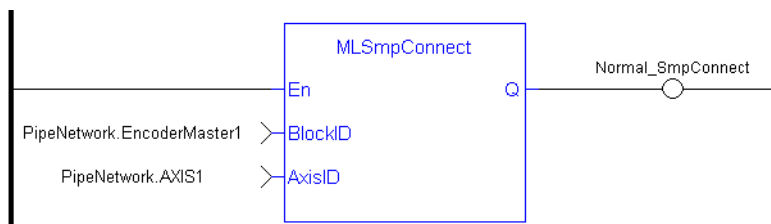
BOOL

**2.1.15.8.7 Example**

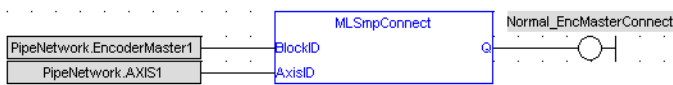
**2.1.15.9.8.1 Structured Text**

```
//Connect a Sampler pipe block named "EncoderMaster1" to a
PipeNetwork Axis block named AXIS1
MLSmpConnect( PipeNetwork.EncoderMaster1, PipeNetwork.AXIS1 ) ;
```

**2.1.15.10.9.2 Ladder Diagram**



**2.1.15.11.10.3 Function Block Diagram**





### 2.1.15.12 MLSmpConECAT Pipe Network ✓

#### 2.1.15.13.1 Description

Connects a sampler block to the specified CoE object. The CoE object must be included in a PDO for the specified EtherCAT device.

Using this function, you can use any EtherCAT data source as input for the specified sampler block.

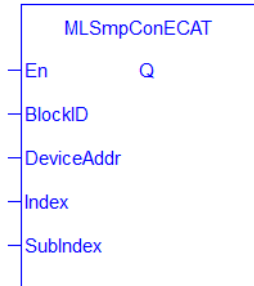


Figure 1-41: MLSmpConECAT function

#### 2.1.15.14.2.1 Related Function Blocks

"MLSmpConnect" (p. 359), "MLSmplnit" (p. 363)

#### 2.1.15.15.3 Arguments

##### 2.1.15.16.4.1 Input

<b>BlockID</b>	<b>Description</b>	ID Name of the Sampler function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>DeviceAddr</b>	<b>Description</b>	The EtherCAT address of the slave device. The first node usually has the value '1001'. Alternately, you can use the members of the EtherCATCode structure to specify a device's address.
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Index</b>	<b>Description</b>	The CoE index of the object to be connected with the Sampler block.
	<b>Data Type</b>	UINT
	<b>Range</b>	—
	<b>Unit</b>	n/a

	<b>Default</b>	—
<b>SubIndex</b>	<b>Description</b>	The CoE sub-index of the object to be connected with the Sampler block.
	<b>Data Type</b>	UINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.15.17.5.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Function block is operational
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.15.18.6.3 Return Type**

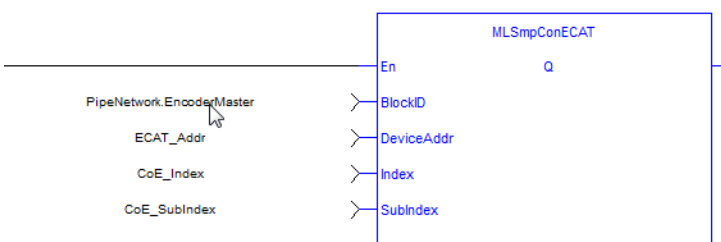
BOOL

**2.1.15.19.7 Example**

**2.1.15.20.8.1 Structured Text**

```
//Connect a Sampler pipe block named "EncoderMaster" to an ECAT
Index Object defined by variable "CoE_SubIndex" with the
SubIndex defined by variable "CoE_SubIndex", from a device with
Ethercat Address defined by "ECAT_Addr"
MLSmpConECAT(PipeNetwork.EncoderMaster, ECAT_Addr, CoE_Index,
CoE_SubIndex );
```

**2.1.15.21.9.2 Ladder Diagram**



**2.1.15.22.10.3 Function Block Diagram**



**2.1.15.23 MLSmpConnectEx** **2.1.15.24.1 Description**

Connect a sampler to the specified data source.

**NOTE**

This function was deprecated in KAS v2.9. Please use either "[MLSmpConECAT](#)" (p. 361), "[MLSmpConPLCAxis](#)" (p. 366), or "[MLSmpConPNAxis](#)" (p. 368) instead.

**2.1.15.25 MLSmplnit** **2.1.15.26.1 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.

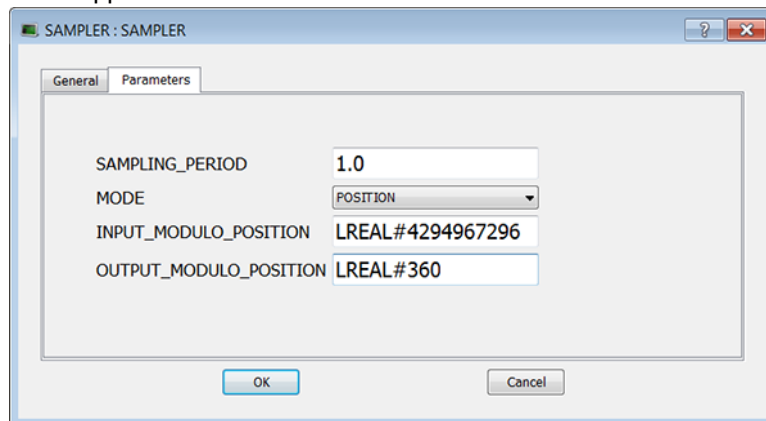
This function block is automatically called by the Function PipeNetwork(MLPN\_CREATE\_OBJECTS) if a Smp Block is added to the Pipe Network, with user-defined settings entered in the Pipe Blocks Properties screen.

The Smp Pipe Block is assigned a Name, SAMPLING\_PERIOD, MODE, INPUT\_VALUE\_PERIOD and OUTPUT\_VALUE\_PERIOD.

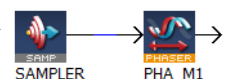
**Using AKD Secondary Feedback**

The Sampler can be connected to the secondary feedback on the AKD using "[MLSmpConPNAxis](#)" (p. 368). The scaling for the AKD Secondary Feedback is setup using AKD Parameters: [DRV.HANDWHEEL](#) and [FB2.ENCRES](#). The feedback signal comes through Ethercat in object 0x2050.

The scaling for this position signal is 0 to 4294967296 = 0 to FB2.ENCRES. Object 0x2050 rollovers to 0 when reaching 4294967296. One way to handle this rollover is in the Sampler block setup parameters set up INPUT\_MODULO\_POSITION for LREAL#4294967296 and OUTPUT\_MODULO\_POSITION to the modulo of the application.



Place a Phaser Block (and write MLPhaWritePhase in the application code) or Gear Block (and write MLGearWriteOff ) after the Sampler Block to offset the Sampler Block Output Position in the pipe Network.

**2.1.15.27.2.1 Related Function Blocks**

"[MLSmpConnect](#)" (p. 359), "[MLSmpConECAT](#)" (p. 361), "[MLSmpConPLCAxis](#)" (p. 366), "[MLSmpConPNAxis](#)" (p. 368)

**2.1.15.28.3 Arguments**

## 2.1.15.29.4.1 Input

<b>BlockID</b>	<b>Description</b>	ID Name of the SMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>SamplingPeriod</b>	<b>Description</b>	period that the device is sampled
	<b>Data type</b>	LREAL
	<b>Range</b>	0.25 to ?
	<b>Unit</b>	millisecond
	<b>Default</b>	1.0
<b>Mode</b>	<b>Description</b>	Sampled output can be either position or velocity
	<b>Data type</b>	DINT
	<b>Range</b>	[1 , 2] Position or Speed
	<b>Unit</b>	n/a
	<b>Default</b>	position
<b>InputModuloPosition</b>	<b>Description</b>	Period of the input signal. This should be set equal to $2^{32}$ (4294967296).
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	360.0
<b>OutputModuloPosition</b>	<b>Description</b>	Period of the output signal
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	360.0

## 2.1.15.30.5.2 Output

<b>Default (.Q)</b>	<b>Description</b>	Smp Block successfully initiated
	<b>Data type</b>	BOOL

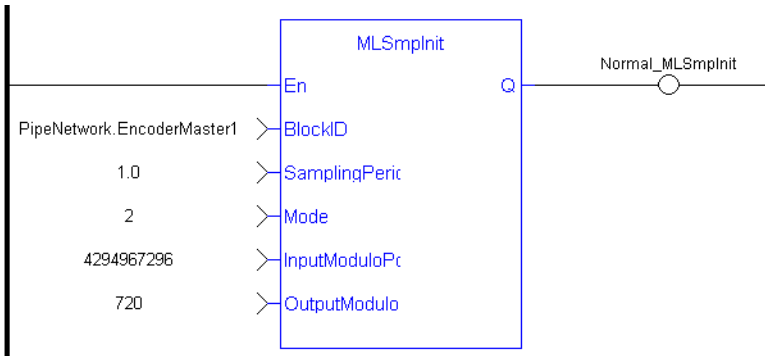
<b>Unit</b>	n/a
-------------	-----

**2.1.15.31.6 Example**

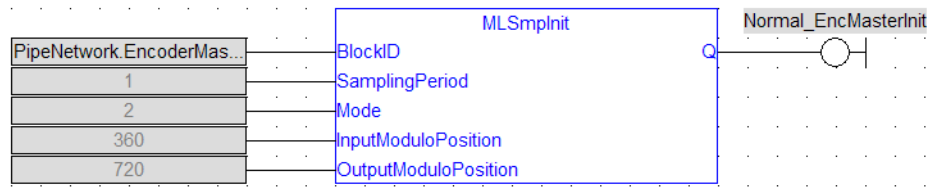
**2.1.15.32.7.1 Structured Text**

```
//Initialize a Sampler Pipe Block named "EncoderMaster1" to a
Sample Period of 1 millisecc, Mode of Operation to 2(Velocity),
Input Modulo of 4294967296, and Output Modulo of 720
MLSmpInit( PipeNetwork.EncoderMaster1, 1.0,2,4294967296,720);
```

**2.1.15.33.8.2 Ladder Diagram**



**2.1.15.34.9.3 Function Block Diagram**



**2.1.15.35 MLSmpConPLCAxis** Pipe Network ✓

**2.1.15.36.1 Description**

This function connects a sampler block to a specific variable from a PLCOpen Axis.



**Figure 1-42:** MLSmpConPLCAxis function

**2.1.15.37.2 Arguments**

**2.1.15.38.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID Name of the SMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxisID</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function (for more details, <a href="#">click here...</a> )
	<b>Data Type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Variable</b>	<b>Description</b>	Variable to be connected to. You need to use one of the following <a href="#">Internal Defines</a> : <ul style="list-style-type: none"> <li>• MC_ACTUAL_POSITION</li> <li>• MC_COMMAND_POSITION</li> <li>• MC_NORMAL_COMMAND_POSITION</li> <li>• MC_SUPERIMPOSED_COMMAND_POSITION</li> <li>• MC_PHASE_COMMAND_POSITION</li> </ul>
	<b>Data Type</b>	UINT
	<b>Range</b>	n/a (use available macros)
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.1.15.39.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Function block is operational
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.15.40.5.3 Return Type**

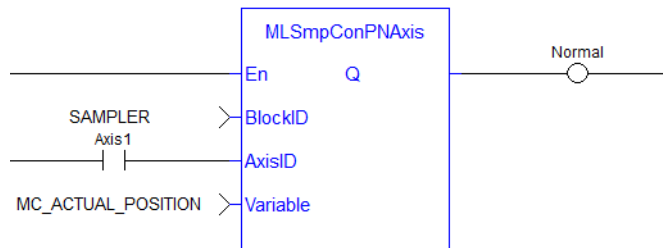
BOOL

**2.1.15.41.6 Example**

**2.1.15.42.7.1 Structured Text**

```
//Connect a Sampler pipe block named "SAMPLER" to a variable
named "MC_ACTUAL_POSITION" from a PLCOpen Axis named Axis1.
MLSmpConPLCAxis( PipeNetwork.SAMPLER, Axis1, MC_ACTUAL_
POSITION) ;
```

**2.1.15.43.8.2 Ladder Diagram**



**2.1.15.44.9.3 Function Block Diagram**



**2.1.15.45 MLsmpConPNAxis** Pipe Network ✓

**2.1.15.46.1 Description**

This function connects a sampler block to a specific variable from a PipeNetwork Axis.



**Figure 1-43:** MLsmpConPNAxis function

**2.1.15.47.2 Arguments**

**2.1.15.48.3.1 Input**

<b>BlockID</b>	<b>Description</b>	ID Name of the SMP function block in the Pipe Network
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxisID</b>	<b>Description</b>	ID Name of the Axis the sampler is connected to
	<b>Data Type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Variable</b>	<b>Description</b>	Variable to be connected to. You need to use one of the following <a href="#">Internal Defines</a> : <ul style="list-style-type: none"> <li>• ML_PIPE_POSITION</li> <li>• ML_REFERENCE_POSITION</li> <li>• ML_GENERATOR_POSITION</li> <li>• ML_ACTUAL_POSITION</li> <li>• ML_FEEDBACK_POSITION</li> <li>• ML_SECOND_FEEDBACK_POSITION</li> <li>• ML_ACTUAL_VELOCITY</li> <li>• ML_ACTUAL_TORQUE</li> <li>• ML_FOLLOWING_ERROR</li> <li>• ML_CURRENT_POSITION</li> </ul>
	<b>Data Type</b>	UINT
	<b>Range</b>	n/a (use available macros)
	<b>Unit</b>	n/a



<b>Default</b>	—
----------------	---

**2.1.15.49.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Function block is operational
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

**2.1.15.50.5.3 Return Type**

BOOL

**2.1.15.51.6 Example**

**2.1.15.52.7.1 Structured Text**

```
//Connect a sampler block named "SAMPLER" to a variable named
ML_PIPE_POSITION from a Pipe Network Axis named
PipeNetwork.AXIS1
MLSmpConPNAxis( PipeNetwork.SAMPLER , PipeNetwork.AXIS1, ML_
PIPE_POSITION );
```

**2.1.15.53.8.2 Ladder Diagram**



**2.1.15.54.9.3 Function Block Diagram**



## 2.1.16 Motion Library - Synchronizer

### TIP

- For a Synchronizer example, see "[Usage example of Synchronizer Functions](#)" (p. 380)

Name	Description	Return type
<a href="#">MLSyncInit</a>	Initializes a synchronizer Pipe Block	BOOL
<a href="#">MLSyncReadDeltaS</a>	Gets the output phasing value of a synchronizer block	None
<a href="#">MLSyncStart</a>	Starts a synchronization of a synchronizer Pipe Block	BOOL
<a href="#">MLSyncStop</a>	De-synchronizes a synchronizer Pipe Block	BOOL
<a href="#">MLSyncWriteDeltaS</a>	Sets the output phasing value of a synchronizer block	BOOL

### 2.1.16.1 MLSyncInit

#### 2.1.16.2.1 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);
```

#### 2.1.16.3.2 Arguments

##### 2.1.16.4.3.1 Input

<b>BLockID</b>	<b>Description</b>	Name of the Pipe Network Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ModuloPosition</b>	<b>Description</b>	The modulo distance
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>CurveType</b>	<b>Description</b>	The curve type to the motion when starting and stopping synchronization. Option are Parabolic or Polynomial
	<b>Data type</b>	DINT
	<b>Range</b>	[1, 2] (1 = Parabolic, 2 = Polynomial)
	<b>Unit</b>	n/a

	<b>Default</b>	—
<b>DeltaS</b>	<b>Description</b>	The Distance to get in or out of synchronization. This parameter is used in the MLSyncStart and MLSyncStop FunctionBlocks
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—

**2.1.16.5.4.2 Output**

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

**2.1.16.6.5 Related Functions**

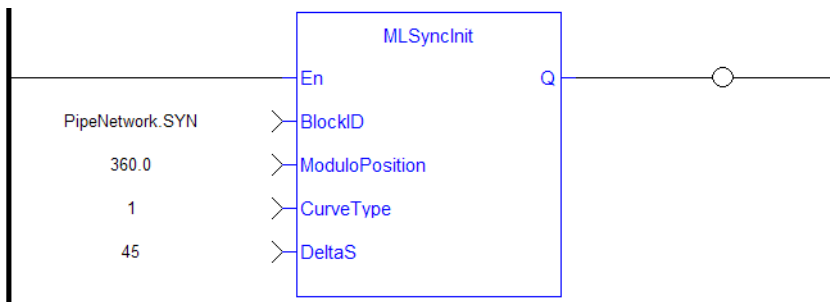
[MLSyncWriteDeltaS](#)

**2.1.16.7.6 Example**

**2.1.16.8.7.1 Structured Text**

```
//Initialize a synchronizer Pipe Block named " SYN" with a
modulo of 360, Curve Type of Parabolic, and a distance (DeltaS)
of 30 to get in and out of synchronization
MLSyncInit( PipeNetwork.SYN, 360, 1, 30 );
```

**2.1.16.9.8.2 Ladder Diagram**



**2.1.16.10.9.3 Function Block Diagram**



2.1.16.11 MLSyncReadDeltaS Pipe Network ✓

2.1.16.12.1 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 "Get Output Phasing after MLSyncStart" (p. 372)). 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 "Get Output Phasing after MLSyncStop" (p. 372)).

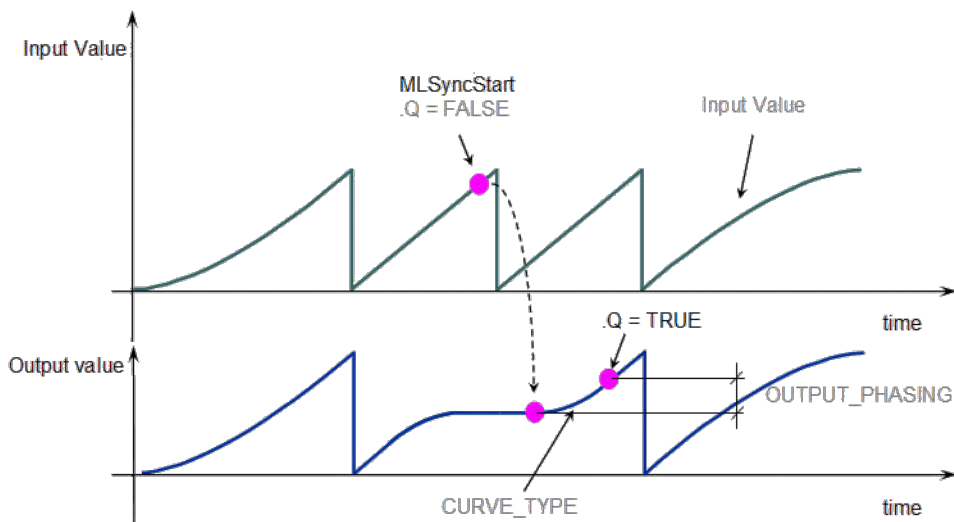


Figure 1-44: Get Output Phasing after MLSyncStart

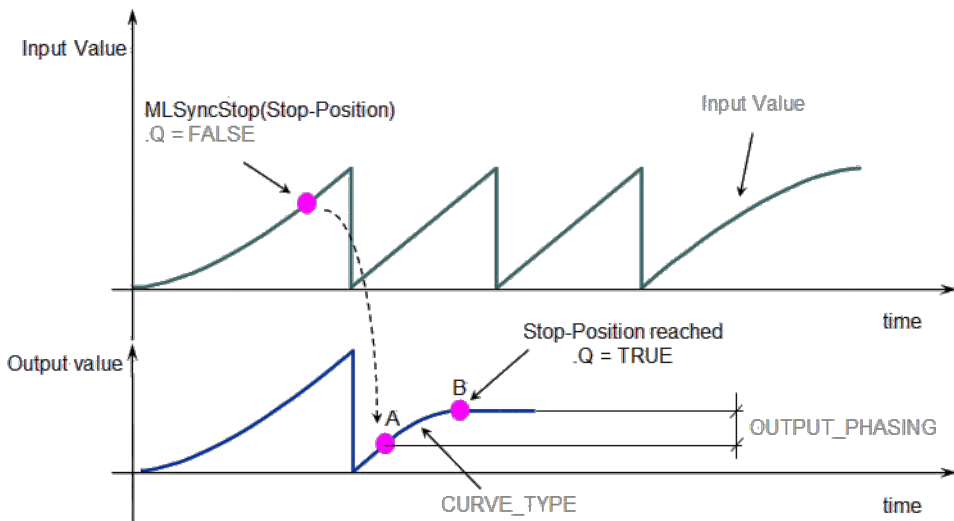


Figure 1-45: Get Output Phasing after MLSyncStop

2.1.16.13.2 Arguments

2.1.16.14.3.1 Input

BlockID	Description	Name of the Pipe Network Block
	Data type	DINT
	Range	[-2147483648, 2147483648]

<b>Unit</b>	n/a
<b>Default</b>	—

**2.1.16.15.4.2 Output**

<b>DeltaS</b>	<b>Description</b>	Present Delta Slope value
	<b>Data type</b>	LREAL
	<b>Unit</b>	User unit

**2.1.16.16.5 Related Functions**

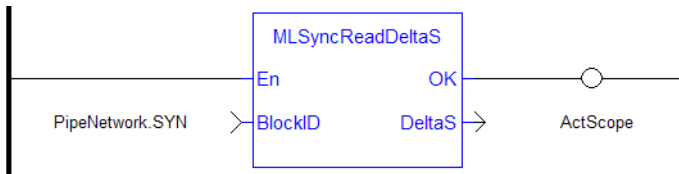
[MLSyncWriteDeltaS](#)

**2.1.16.17.6 Example**

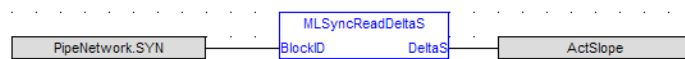
**2.1.16.18.7.1 Structured Text**

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

**2.1.16.19.8.2 Ladder Diagram**



**2.1.16.20.9.3 Function Block Diagram**



**2.1.16.21 MLSyncStart** Pipe Network ✓

**2.1.16.22.1 Description**

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

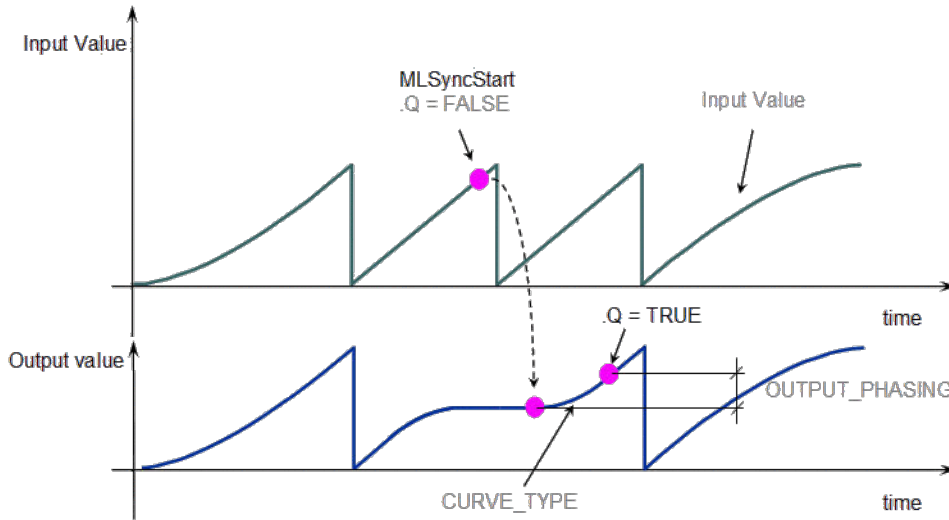


Figure 1-46: MLSyncStart

**2.1.16.23.2 Arguments**

**2.1.16.24.3.1 Input**

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

**2.1.16.25.4.2 Output**

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

**2.1.16.26.5 Example**

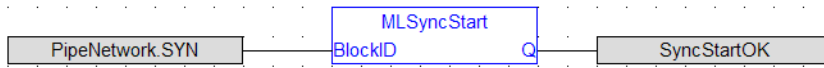
**2.1.16.27.6.1 Structured Text**

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

**2.1.16.28.7.2 Ladder Diagram**



### 2.1.16.29.8.3 Function Block Diagram



**2.1.16.30 MLSyncStop** Pipe Network ✓

**2.1.16.31.1 Description**

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

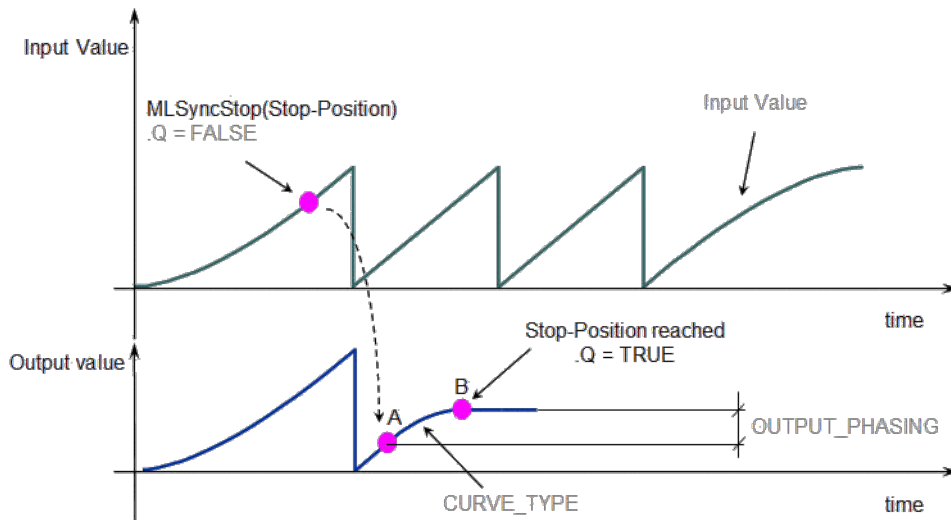


Figure 1-47: MLSyncStop

**2.1.16.32.2 Arguments**

**2.1.16.33.3.1 Input**

<b>Position</b>	<b>Description</b>	Motion Stop Position
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—

**2.1.16.34.4.2 Output**

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

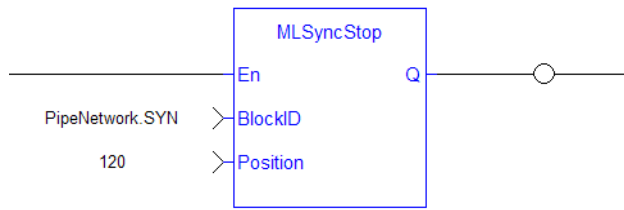
**2.1.16.35.5 Example**

**2.1.16.36.6.1 Structured Text**

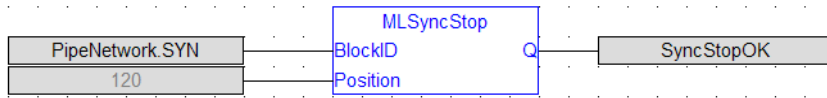
```
MLSyncStop( PipeNetwork.SYN , 120 );
```

**2.1.16.37.7.2 Ladder Diagram**





### 2.1.16.38.8.3 Function Block Diagram



### 2.1.16.39 MLSyncWriteDeltaS

#### 2.1.16.40.1 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.

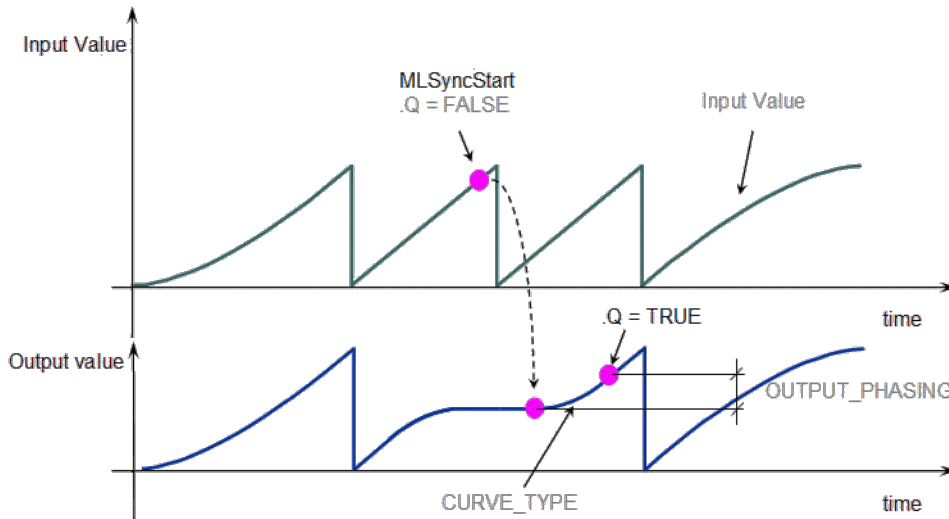


Figure 1-48: Set output phasing after MLSyncStart

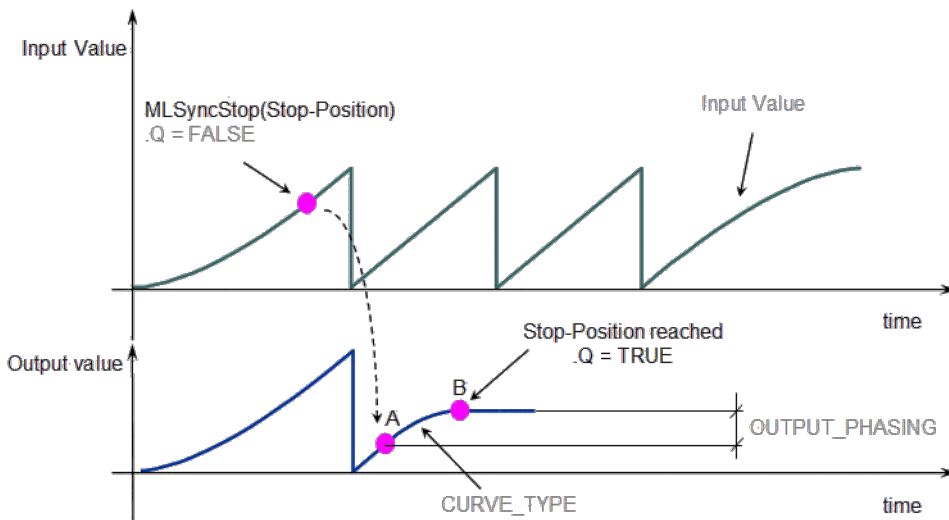


Figure 1-49: Set output phasing after MLSyncStop

#### 2.1.16.41.2 Arguments

##### 2.1.16.42.3.1 Input

<b>BlockID</b>	<b>Description</b>	Name of the Pipe Network Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]

	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>DeltaS</b>	<b>Description</b>	Slope to be used during Start and stop of Synchronization
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—

#### 2.1.16.43.4.2 Output

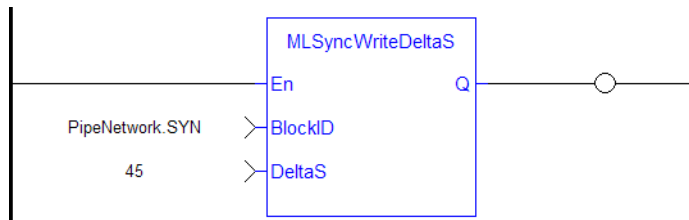
<b>Default (.Q)</b>	<b>Description</b>	Function Block Execute Successfully
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

#### 2.1.16.44.5 Example

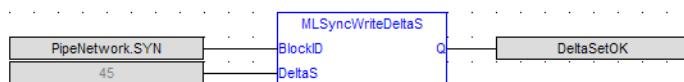
##### 2.1.16.45.6.1 Structured Text

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

#### 2.1.16.46.7.2 Ladder Diagram



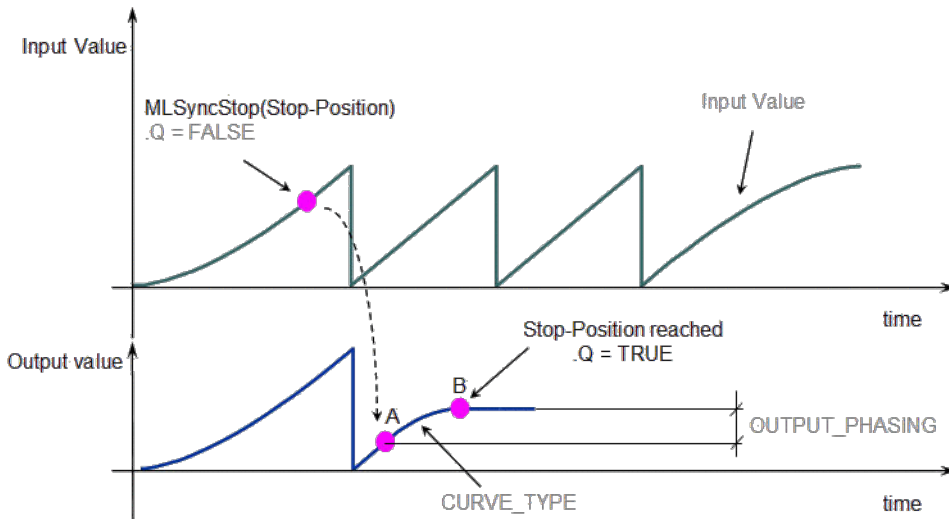
#### 2.1.16.47.8.3 Function Block Diagram



### 2.1.16.48 Usage example of Synchronizer Functions

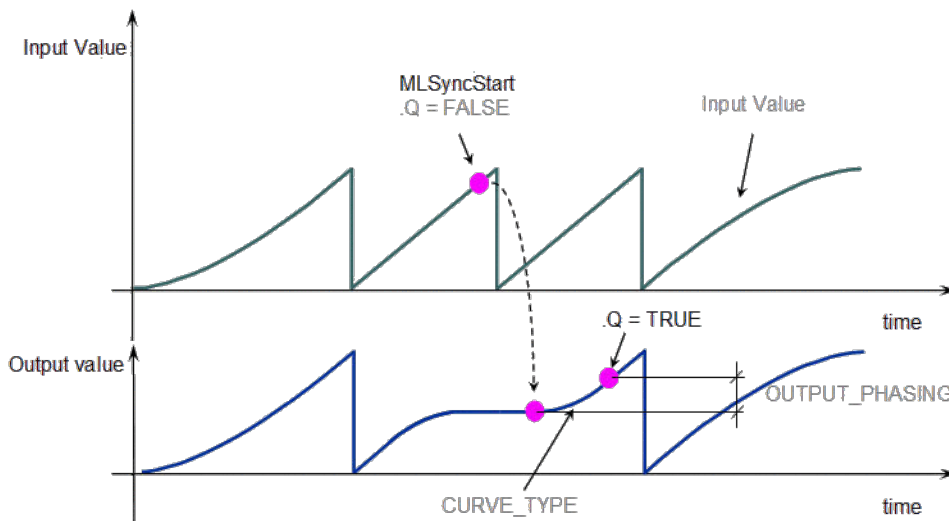
When you call the **MLSyncStop** function, the output value is adapted according to the specified Stop-Position (point B).

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 1-50:** Synchronizer Functions Usage

## 2.1.17 Motion Library - Trigger

### TIP

- For an example of Trigger functions, see "[Usage example of Trigger Functions](#)" (p. 396)

Name	Description	Return type
<a href="#">MLTrigClearFlag</a>	Clears the flag of an initiated Trigger block	BOOL
<a href="#">MLTrigInit</a>	Initializes a Trigger object	BOOL
<a href="#">MLTrigsTriggered</a>	Checks if the selected block has been triggered	BOOL
<a href="#">MLTrigReadDelay</a>	Returns the time that the trigger block uses to compensate the delay of the sensor that captures the triggering signal	None
<a href="#">MLTrigReadPos</a>	Returns the position of the block at the moment when it was triggered	None
<a href="#">MLTrigReadTime</a>	Returns the time of the moment where the block was triggered in milliseconds	None
" <a href="#">MLTrigSetEdge</a> " (p. 393)	Sets the edge configuration for a Trigger object	BOOL
<a href="#">MLTrigWriteDelay</a>	Sets the time that the trigger block uses to compensate for the delay introduced by the sensor that captures the triggering signal	BOOL

### 2.1.17.1 MLTrigClearFlag

#### 2.1.17.2.1 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`

#### 2.1.17.3.2 Arguments

##### 2.1.17.4.3.1 Input

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

##### 2.1.17.5.4.2 Output

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

##### 2.1.17.6.5.3 Return Type

BOOL

### 2.1.17.7.6 Related Functions

[MLAxisRstFastIn](#)

[MLTrigsTriggered](#)

[MLTrigReadPos](#)

[MLTrigReadTime](#)

### 2.1.17.8 See Also

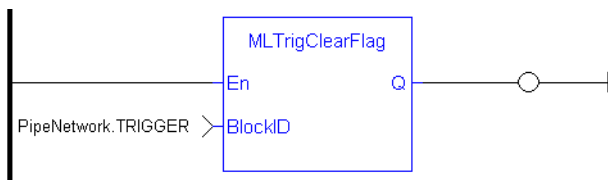
- [Fast Inputs with Pipe Network Motion](#)
- [Fast Inputs with PLCOpen](#)
- [Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block](#)
- [Fast Homing Example with the PLCopen Motion Engine](#)
- [Pipe Network Registration and Fast Homing](#)
- [Registration Position Capture Example with Pipe Network Trigger Block](#)

### 2.1.17.9.1 Example

#### 2.1.17.10.2.1 Structured Text

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

### 2.1.17.11.3.2 Ladder Diagram



### 2.1.17.12.4.3 Function Block Diagram



### 2.1.17.13 MLTrigInit

#### 2.1.17.14.1 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.

#### 2.1.17.15.2 Arguments

##### 2.1.17.16.3.1 Input

<b>BlockID</b>	<b>Description</b>	ID number of a created Pipe Block
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Input_Axis</b>	<b>Description</b>	Name of the axis where the Fast Input is located
	<b>Data type</b>	STRING
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>InputID</b>	<b>Description</b>	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 <a href="#">AKD Capture Engine Configuration</a> .
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]

	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>EdgeID</b>	<b>Description</b>	Trigger at rising or falling edge of Fast Input. Enter 1 for rising edge, 2 for falling edge, and 0 disables the Fast Input
	<b>Data type</b>	DINT
	<b>Range</b>	[0, 2]
	<b>Unit</b>	n/a
	<b>Default</b>	1 (Rising edge)

#### 2.1.17.17.4.2 Output

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

#### 2.1.17.18.5.3 Return Type

BOOL

#### 2.1.17.19.6 Related Functions

[MLTrigsTriggered](#)

[MLTrigReadPos](#)

[MLTrigClearFlag](#)

[MLAxisRstFastIn](#)

#### 2.1.17.20 See Also

- [Fast Inputs with Pipe Network Motion](#)
- [Fast Inputs with PLCOpen](#)
- [Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block](#)
- [Fast Homing Example with the PLCOpen Motion Engine](#)
- [Pipe Network Registration and Fast Homing](#)
- [Registration Position Capture Example with Pipe Network Trigger Block](#)

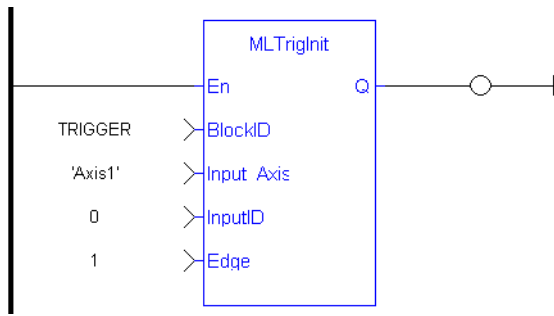
#### 2.1.17.21.1 Example

##### 2.1.17.22.2.1 Structured Text

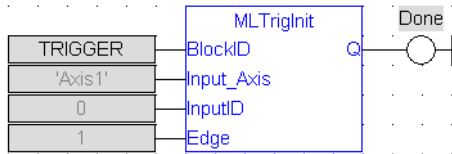
```
//Create and Initiate a Trigger Pipe Block named "Trigger" and
set it up to receive the trigger signal from Axis1, capture
engine 0, and the rising edge of the signal
TRIGGER := MlBlkCreate( 'TRIGGER', 'TRIGGER' );
MLTrigInit( TRIGGER, 'Axis1', 0, 1 );
```

#### 2.1.17.23.3.2 Ladder Diagram





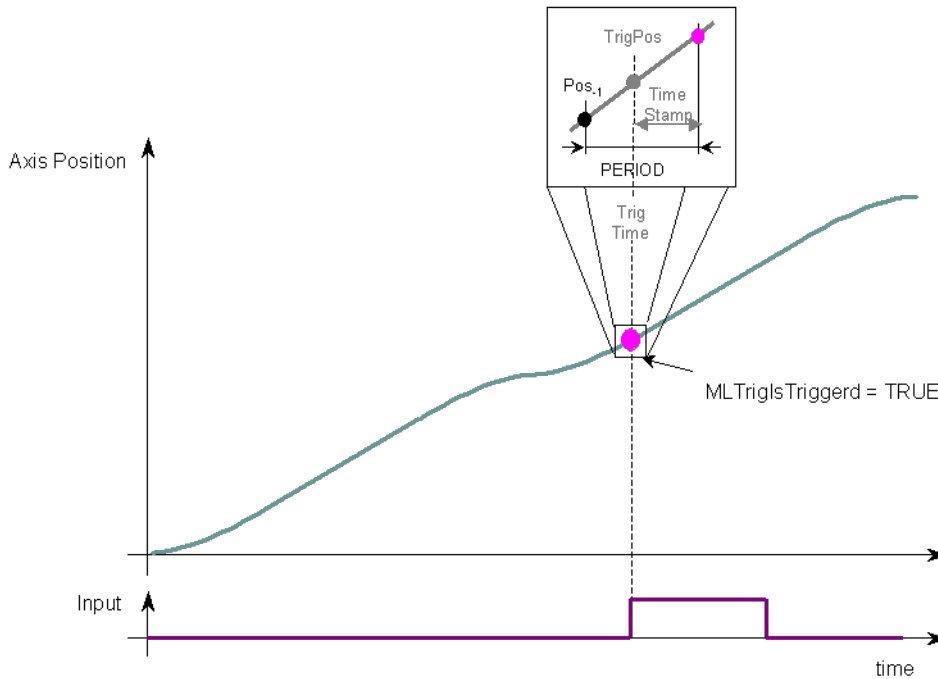
**2.1.17.24.4.3 Function Block Diagram**



**2.1.17.25 MLTrigsTriggered** Pipe Network ✓

**2.1.17.26.1 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 1-51:** 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.

**2.1.17.27.2 Arguments**

**2.1.17.28.3.1 Input**

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

**2.1.17.29.4.2 Output**

<b>Default (.Q)</b>	<b>Description</b>	Returns TRUE if the selected Trigger Object has Triggered
---------------------	--------------------	---

<b>Data type</b>	BOOL
<b>Unit</b>	n/a

### 2.1.17.30.5.3 Return Type

BOOL

### 2.1.17.31.6 Related Functions

[MLTrigReadPos](#)

[MLTrigReadTime](#)

### 2.1.17.32 See Also

- [Fast Inputs with Pipe Network Motion](#)
- [Fast Inputs with PLCOpen](#)
- [Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block](#)
- [Fast Homing Example with the PLCopen Motion Engine](#)
- [Pipe Network Registration and Fast Homing](#)
- [Registration Position Capture Example with Pipe Network Trigger Block](#)

### 2.1.17.33.1 Example

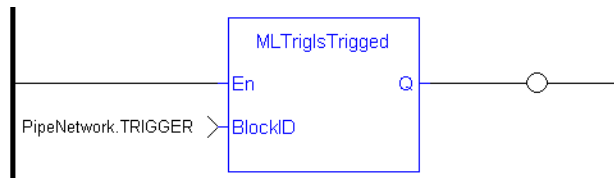
//Check if a Trigger Block has been triggered, then save position

IF MLTrigsTriggered( PipeNetwork.TRIGGER ) THEN

Trig\_Position := MLTrigReadPos( PipeNetwork.TRIGGER );

END\_IF

### 2.1.17.34.2.1 Ladder Diagram



### 2.1.17.35.3.2 Function Block Diagram



### 2.1.17.36 MLTrigReadDelay Pipe Network ✓

#### 2.1.17.37.1 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.

#### 2.1.17.38.2.1 Input

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

#### 2.1.17.39.3.2 Output

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

#### 2.1.17.40.4 Related Functions

[MLTrigWriteDelay](#)

#### 2.1.17.41 See Also

- [Fast Inputs with Pipe Network Motion](#)
- [Fast Inputs with PLCOpen](#)
- [Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block](#)
- [Fast Homing Example with the PLCOpen Motion Engine](#)
- [Pipe Network Registration and Fast Homing](#)
- [Registration Position Capture Example with Pipe Network Trigger Block](#)

2.1.17.42 MLTrigReadPos Pipe Network ✓

2.1.17.43.1 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.

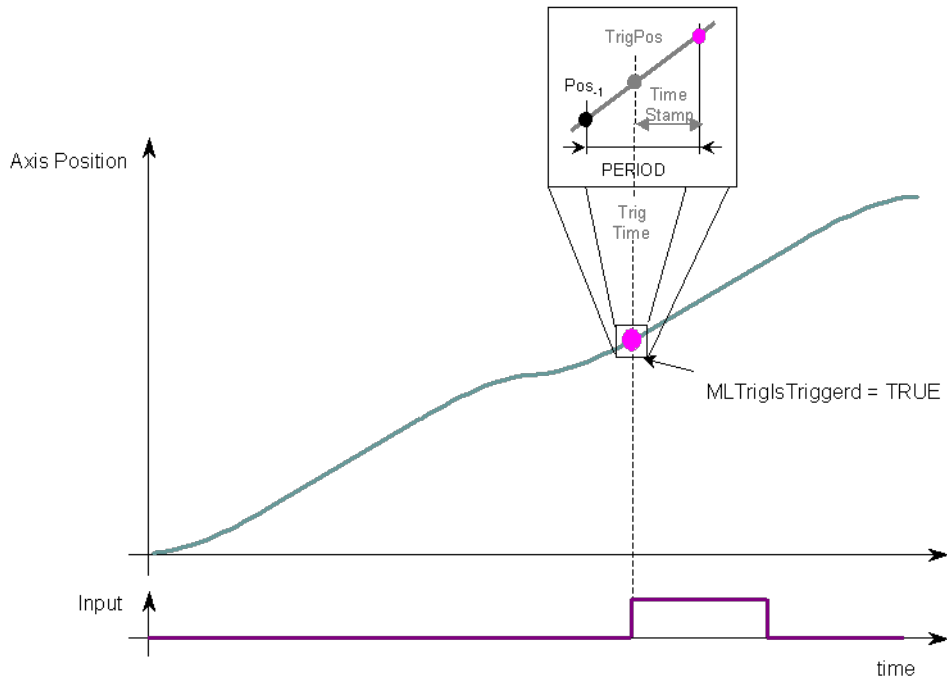


Figure 1-52: MLTrigReadPos

2.1.17.44.2 Arguments

2.1.17.45.3.1 Input

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

2.1.17.46.4.2 Output

<b>Position</b>	<b>Description</b>	Returns the position of the selected block's Axis at the moment when it was triggered
	<b>Data type</b>	LREAL

Unit	User unit
------	-----------

### 2.1.17.47.5 Related Functions

[MLTrigsTriggered](#)

[MLTrigReadTime](#)

[MLTrigClearFlag](#)

[MLAxisRstFastIn](#)

### 2.1.17.48 See Also

- [Fast Inputs with Pipe Network Motion](#)
- [Fast Inputs with PLCOpen](#)
- [Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block](#)
- [Fast Homing Example with the PLCOpen Motion Engine](#)
- [Pipe Network Registration and Fast Homing](#)
- [Registration Position Capture Example with Pipe Network Trigger Block](#)

### 2.1.17.49.1 Previous Function Name

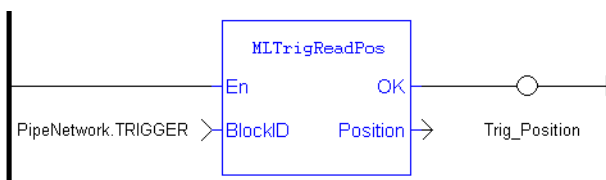
MLTrigGetPos

### 2.1.17.50.2 Example

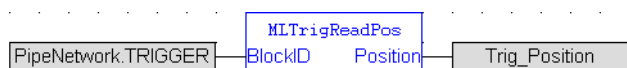
#### 2.1.17.51.3.1 Structured Text

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

### 2.1.17.52.4.2 Ladder Diagram



### 2.1.17.53.5.3 Function Block Diagram



**2.1.17.54 MLTrigReadTime** Pipe Network ✓

**2.1.17.55.1 Description**

Returns the time of the moment where the block was triggered in milliseconds. This value is only valid when TrigIsTriggered() 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.

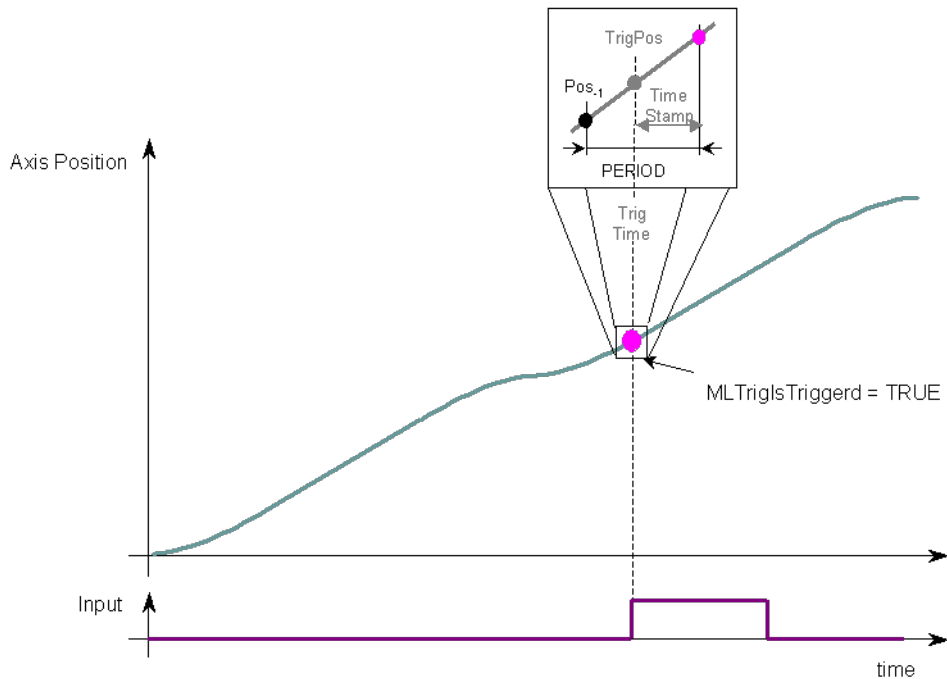


Figure 1-53: MLTrigReadTime

**2.1.17.56.2 Arguments**

**2.1.17.57.3.1 Input**

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

**2.1.17.58.4.2 Output**

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

### 2.1.17.59.5 Related Functions

[MLTrigsTriggered](#)

[MLTrigReadPos](#)

[MLTrigClearFlag](#)

[MLAxisRstFastIn](#)

### 2.1.17.60 See Also

- [Fast Inputs with Pipe Network Motion](#)
- [Fast Inputs with PLCOpen](#)
- [Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block](#)
- [Fast Homing Example with the PLCOpen Motion Engine](#)
- [Pipe Network Registration and Fast Homing](#)
- [Registration Position Capture Example with Pipe Network Trigger Block](#)

### 2.1.17.61.1 Previous Function Name

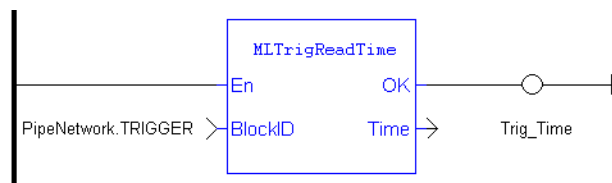
MLTrigGetTime

### 2.1.17.62.2 Example

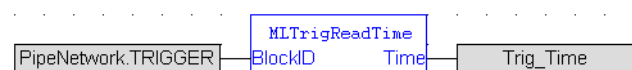
//Save time when Fast Input event occurs

```
Trig_Time := MLTrigReadTime( PipeNetwork.TRIGGER );
```

### 2.1.17.63.3.1 Ladder Diagram



### 2.1.17.64.4.2 Function Block Diagram





**2.1.17.65 MLTrigSetEdge** **2.1.17.66.1 Description**

Sets the edge configuration (rising, falling, etc.) for a trigger block. This block should be called prior to calling "MLAxisCfgFastIn" (p. 132). Also the value at the Edge input must match the value at MLAxisCfgFastIn's Mode input.

**2.1.17.67.2 Arguments****2.1.17.68.3.1 Input**

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

**2.1.17.69.4.2 Output**

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

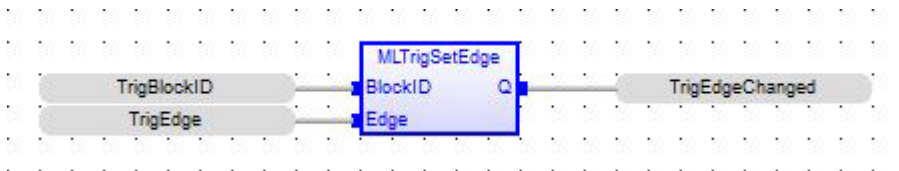
**2.1.17.70.5.3 Return Type**

BOOL

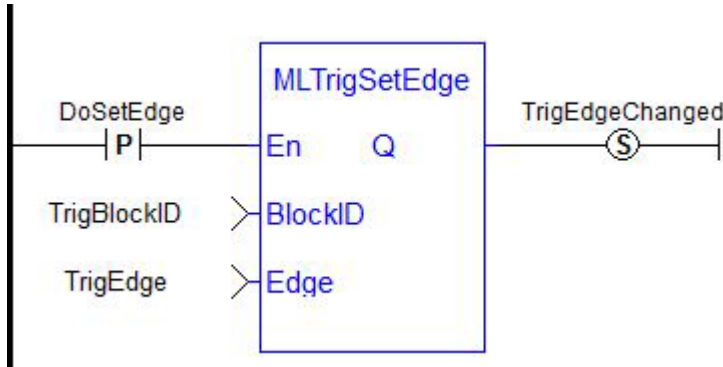
**2.1.17.71 See Also**

- [Fast Inputs with Pipe Network Motion](#)
- [Fast Inputs with PLCOpen](#)
- [Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block](#)
- [Fast Homing Example with the PLCOpen Motion Engine](#)
- [Pipe Network Registration and Fast Homing](#)
- [Registration Position Capture Example with Pipe Network Trigger Block](#)

**2.1.17.72.1 Examples****2.1.17.73.2.1 Function Block Diagram**



### 2.1.17.74.3.2 Ladder Diagram



### 2.1.17.75.4.3 Structured Text

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

### 2.1.17.76 MLTrigWriteDelay Pipe Network ✓

#### 2.1.17.77.1 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.

#### 2.1.17.78.2.1 Input

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

#### 2.1.17.79.3.2 Output

<b>Default (.Q)</b>	<b>Description</b>	Returns TRUE if the delay is successfully set
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

#### 2.1.17.80.4.3 Return Type

BOOL

#### 2.1.17.81.5 Related Functions

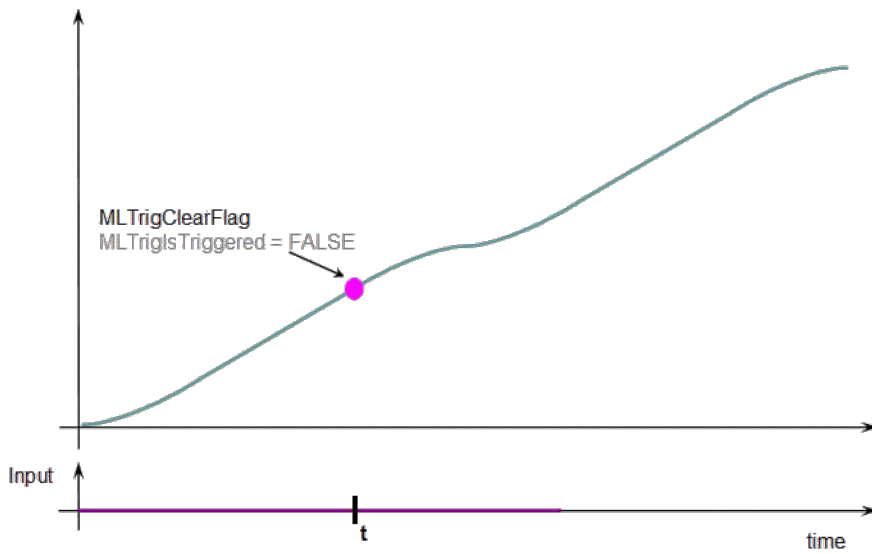
[MLTrigReadDelay](#)

#### 2.1.17.82 See Also

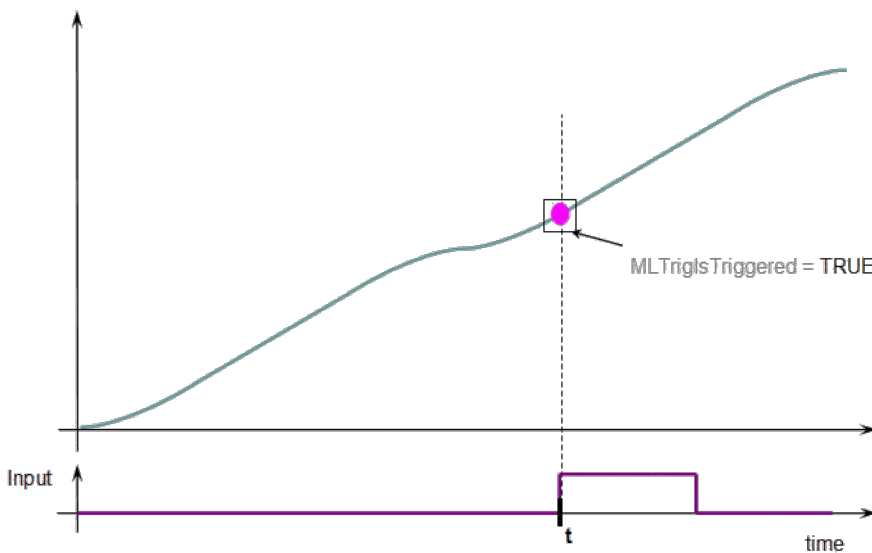
- [Fast Inputs with Pipe Network Motion](#)
- [Fast Inputs with PLCOpen](#)
- [Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block](#)
- [Fast Homing Example with the PLCOpen Motion Engine](#)
- [Pipe Network Registration and Fast Homing](#)
- [Registration Position Capture Example with Pipe Network Trigger Block](#)

### 2.1.17.83 Usage example of Trigger Functions

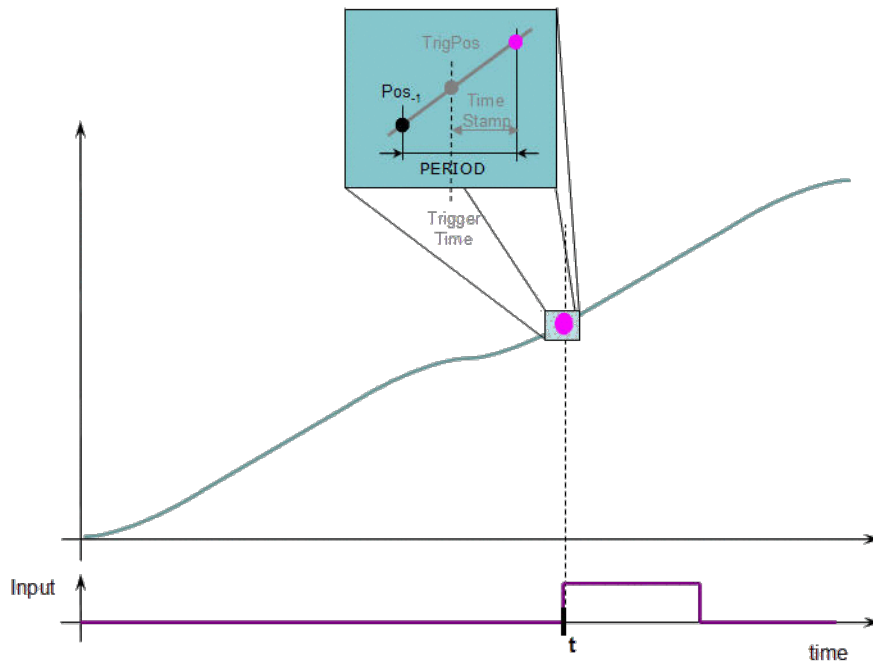
When you call the **MLTrigClearFlag** function, the flag for trigger is reset to False.



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 1-54:** 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](#).

## 2.2 Motion Library / PLCopen

Functions sorted in alphabetical order:

Name	Description
<a href="#">MC_AbortTrigger</a>	Abort MC_TouchProbe
"MC_AddSuperAxis" (p. 545)	Add an axis to the axis's list of assigned, superimposed axes.
<a href="#">MC_CamIn</a>	Performs a slave axis move based on the Cam Table
<a href="#">MC_CamOut</a>	Disengages the slave axis from a MC_CamIn move
"MC_CamResumePos" (p. 490)	Returns the slave axis position for resuming an MC_CamIn move
"MC_CamStartPos" (p. 493)	Returns the slave axis position for starting an MC_CamIn move
<a href="#">MC_CamTblSelect</a>	Defined to read and initialize the specified profile
<a href="#">MC_ClearFaults</a>	Clear Drive Faults
<a href="#">MC_CreateAxis</a>	Creates a PLCopen Axis
<a href="#">MC_EStop</a>	Performs a Emergency stop
"MC_ErrorDescription" (p. 549)	Converts the PLCopen error IDs into message strings.
<a href="#">MC_GearIn</a>	Performs a slave axis move based on the ratio
<a href="#">MC_GearInPos</a>	Performs a slave axis move based on the ratio
<a href="#">MC_GearOut</a>	Disengages the slave axis from a MC_GearIn or MC_GearInPos move
<a href="#">MC_Halt</a>	Decelerates an axis to zero velocity
<a href="#">MC_InitAxis</a>	Initializes a PLCopen Servo Axis' data
<a href="#">MC_MachRegist</a>	Runs Mark-to-Machine registration
<a href="#">MC_MarkRegist</a>	Runs Mark-to-Mark registration
<a href="#">MC_MoveAbsolute</a>	Performs a single-axis move to a specified endpoint position
<a href="#">MC_MoveAdditive</a>	Performs a single-axis move for a specified distance from the endpoint of the previous move
"MC_MoveRelative" (p. 458)	Performs a single-axis move for a specified distance
<a href="#">MC_MoveSuperimp</a>	Performs a single-axis move which is superimposed upon the active move
<a href="#">MC_MoveVelocity</a>	Performs a single-axis non-ending move at a specified velocity
<a href="#">MC_Phasing</a>	Performs a master position phase shift for the slave axis
<a href="#">MC_Power</a>	Requests to enable the drive and close the loop, or disable the drive and open the loop
<a href="#">MC_ReadActPos</a>	Reads the actual position of the axis
<a href="#">MC_ReadActVel</a>	Reads the actual velocity of the axis
<a href="#">MC_ReadAxisErr</a>	Returns the error status of the specified axis
<a href="#">MC_ReadBoolPar</a>	Returns the value of the specified Boolean axis parameter
<a href="#">MC_ReadParam</a>	Returns the value of the specified axis parameter
<a href="#">MC_ReadStatus</a>	Returns the state of the specified axis
<a href="#">MC_Reference</a>	Defines the position at the reference location for PLCopen Axis

Name	Description
<a href="#">"MC_RemSuperAxis" (p. 547)</a>	Remove an axis from the axis's list of assigned, superimposed axes.
<a href="#">MC_ResetError</a>	Resets the errors of the specified axis
<a href="#">MC_SetOverride</a>	Writes velocity and acceleration override factors
<a href="#">MC_SetPosition</a>	Deprecated by <a href="#">"MC_SetPos" (p. 527)</a>
<a href="#">"MC_SetPos" (p. 527)</a>	Sets a new axis position
<a href="#">MC_Stop</a>	Aborts the active move, removes the next move from the queue, performs a controlled stop, and switches the axis to Stopping state
<a href="#">MC_StopRegist</a>	Turns off registration for the specified axis
<a href="#">MC_SyncSlaves</a>	Specifies synchronized slaves
<a href="#">MC_TouchProbe</a>	Arm a Fast Input and capture an axis position
<a href="#">MC_WriteBoolPar</a>	Writes the specified axis Boolean parameter
<a href="#">MC_WriteParam</a>	Writes the specified axis parameter

## 2.2.1 Control Functions

This set of functions provide general controls to drives and axes.

### 2.2.1.1 MC\_ClearFaults

#### 2.2.1.2.1 Description

The MC\_ClearFaults function 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.

#### TIP

This function does **not** reset axis errors. [MC\\_ResetError](#) is required to reset axis errors and possibly to re-enable or turn power on to the servo axis after the fault condition is cleared.

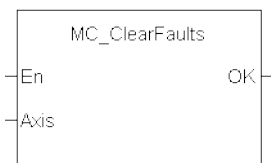


Figure 1-55: MC\_ClearFaults

#### 2.2.1.3.2 Arguments

##### 2.2.1.4.3.1 Input

En	Description
	Function enable – execute function. This Input must be on shot.
	<b>Data type</b> BOOL
	<b>Range</b> 0, 1
	<b>Unit</b> n/a

	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	AXIS_REF.AXIS_NUM is the master axis number
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 2.2.1.5.4.2 Output

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

#### 2.2.1.6.5 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.

#### 2.2.1.7.6 Related Functions

[MC\\_ResetError](#)

#### 2.2.1.8.7 Example

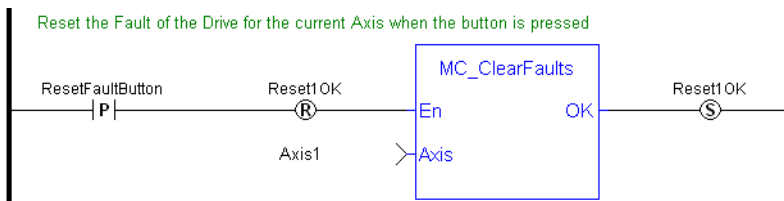
#### 2.2.1.9.8.1 Structured Text

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

#### 2.2.1.10.9.2 Function Block Diagram



#### 2.2.1.11.10.3 Ladder Diagram





### 2.2.1.12 MC\_CreateAxis

#### 2.2.1.13.1 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" (p. 571) and "MLMotionStart" (p. 574).

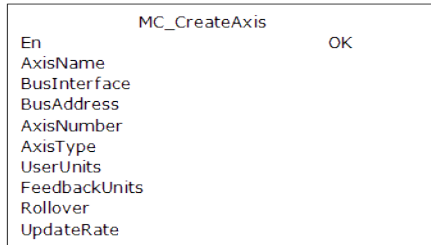


Figure 1-56: MC\_CreateAxis

#### 2.2.1.14.2 Arguments

##### 2.2.1.15.3.1 Input

<b>En</b>	<b>Description</b>	Requests to create a PLCopen axis
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxisName</b>	<b>Description</b>	Axis name
	<b>Data type</b>	STRING
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>BusInterface</b>	<b>Description</b>	Bus interface identifier: "EtherCATDriver" = EtherCAT interface "MSBusDriver" = KAS Simulator interface
	<b>Data type</b>	STRING
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>BusAddress</b>	<b>Description</b>	Address of the drive on the bus

	<b>Data type</b>	DINT
	<b>Range</b>	bus dependent
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxisNumber</b>	<b>Description</b>	Axis number
	<b>Data type</b>	UINT
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxisType</b>	<b>Description</b>	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
	<b>Data type</b>	USINT
	<b>Range</b>	[0,1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>UserUnits</b>	<b>Description</b>	User unit portion of the user unit/feedback unit ratio
	<b>Data type</b>	DINT
	<b>Range</b>	[1, 2147483647]
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>FeedbackUnits</b>	<b>Description</b>	Feedback unit portion of the user unit/feedback unit ratio
	<b>Data type</b>	DINT
	<b>Range</b>	[1, 1073741824]
	<b>Unit</b>	feedback units. <b>Note:</b> <i>The FeedbackUnits input must be a power of 2.</i> If input FeedbackUnits is not a power of two, the axis will not be created, and the OK output will be FALSE.
	<b>Default</b>	—
<b>Rollover</b>	<b>Description</b>	Rollover position (0 = no rollover)
	<b>Data type</b>	LREAL
	<b>Range</b>	[0, 4294967296]

	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>UpdateRate</b>	<b>Description</b>	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
	<b>Data type</b>	UINT
	<b>Range</b>	[3,9]
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 2.2.1.16.4.2 Output

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

#### 2.2.1.17.5 Example

##### 2.2.1.18.6.1 Structured Text

```
(* MC_CreateAxis ST Example *)
AxisName1      := 'PLCOpenAxis1';
BusName1       := 'EtherCATDriver';
BusAddress1    := 1001;
AxisNumber1    := 1;
AxisType1      := MC_AXIS_TYPE_SERVO;
UserUnits1     := 360;
FeedbackUnits1 := 1048576;
Rollover1      := 0;
UpdateRate1    := 3;
MC_CreateAxis(AxisName1, BusName1, BusAddress1, AxisNumber1,
AxisType1, UserUnits1, FeedbackUnits1, Rollover1, UpdateRate1);
```

#### 2.2.1.19.7.2 Function Block Diagram



**2.2.1.21 MC\_EStop** **2.2.1.22.1 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 1-57: MC\_EStop

**2.2.1.23.2 Arguments****2.2.1.24.3.1 Input**

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

**2.2.1.25.4.2 Output**

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

**2.2.1.26.5 Usage**

Call MC\_EStop to generate an emergency stop for an axis.

Call "[MC\\_ResetError](#)" (p. 415) to reset the emergency stop.

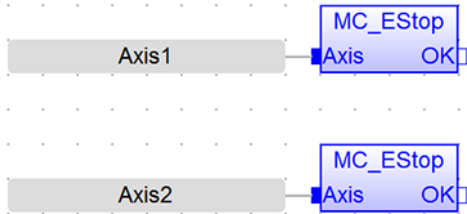
**2.2.1.27.6 Related Functions**

["MC\\_ResetError"](#) (p. 415)

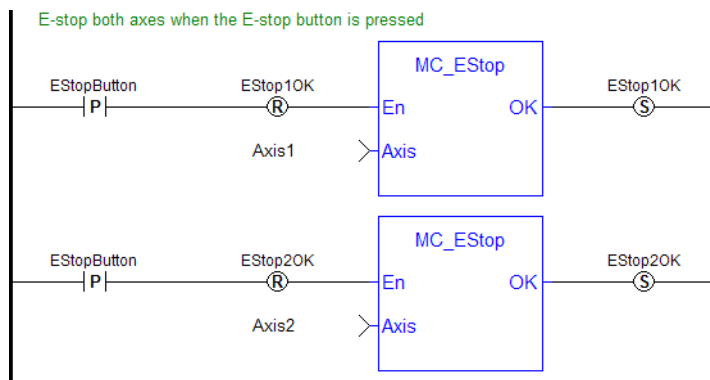
**2.2.1.28.7 Example****2.2.1.29.8.1 Structured Text**

```
(* MC_EStop ST example *)
ON EStopButton DO
MC_EStop( Axis1 );
MC_EStop( Axis2 );
END_DO;
```

### 2.2.1.30.9.2 Function Block Diagram



### 2.2.1.31.10.3 Ladder Diagram



### 2.2.1.32 MC\_InitAxis

#### 2.2.1.33.1 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.

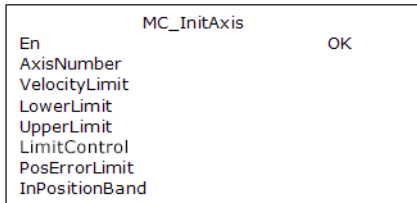


Figure 1-58: MC\_InitAxis

#### 2.2.1.34.2 Arguments

##### 2.2.1.35.3.1 Input

<b>En</b>	<b>Description</b>	Request to initialize a PLCopen servo axis
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxisNumber</b>	<b>Description</b>	Servo axis number
	<b>Data type</b>	UINT
	<b>Range</b>	[1,256]
	<b>Unit</b>	none
	<b>Default</b>	—
<b>VelocityLimit</b>	<b>Description</b>	Velocity limit
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	—
<b>LowerLimit</b>	<b>Description</b>	Lower position limit
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—

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

#### 2.2.1.36.4.2 Output

<b>OK</b>	<b>Description</b>	Indicates the initialization is complete
	<b>Data type</b>	BOOL

#### 2.2.1.37.5 Example

#### 2.2.1.38.6.1 Structured Text

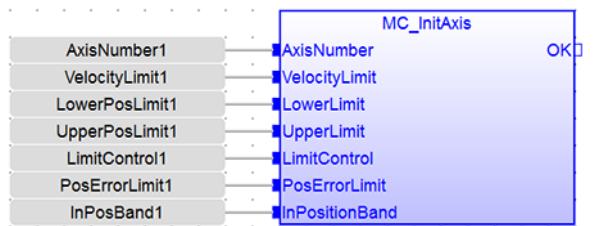


```

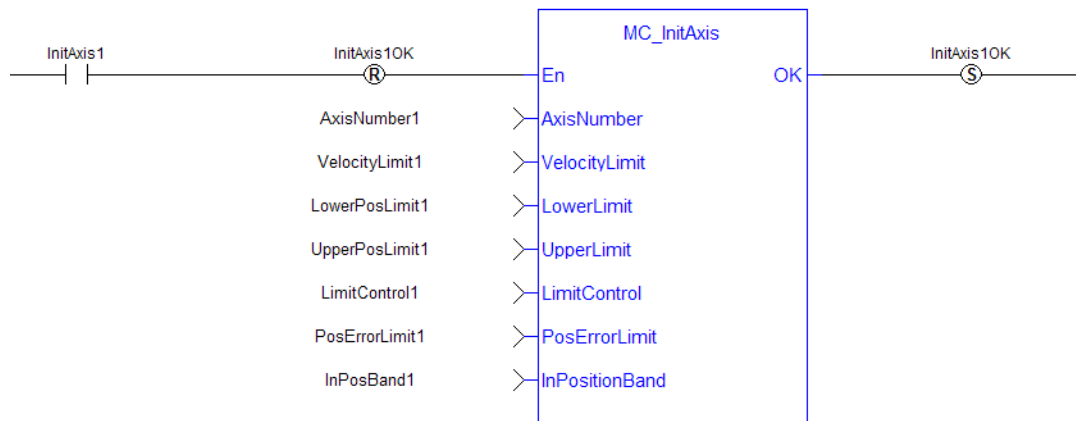
(* MC_InitAxis ST example *)
AxisNumber1      := 1;
VelocityLimit1   := 10000; (*User unit/second*)
LowerPosLimit1   := 0;
UpperPosLimit1   := 0;
LimitControl1    := 0; (* Ignore lower and upper pos limit*)
PosErrorLimit1   := 10; (*User unit*)
InPosBand1       := 0;
MC_InitAxis(AxisNumber1, VelocityLimit1, LowerPosLimit1,
UpperPosLimit1, LimitControl1, PosErrorLimit1, InPosBand1);

```

### 2.2.1.39.7.2 Function Block Diagram



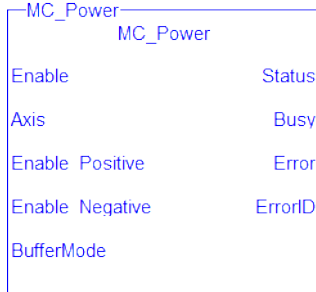
### 2.2.1.40.8.3 Ladder Diagram



**2.2.1.41 MC\_Power** PLCopen

**2.2.1.42.1 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 1-59:** 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.

**2.2.1.43.2 Arguments**

**2.2.1.44.3.1 Input**

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

	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Enable Negative</b>	<b>Description</b>	<i>for future enhancement</i>
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>BufferMode</b>	<b>Description</b>	Unused
	<b>Data type</b>	SINT
	<b>Range</b>	[0]
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 2.2.1.45.4.2 Output

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

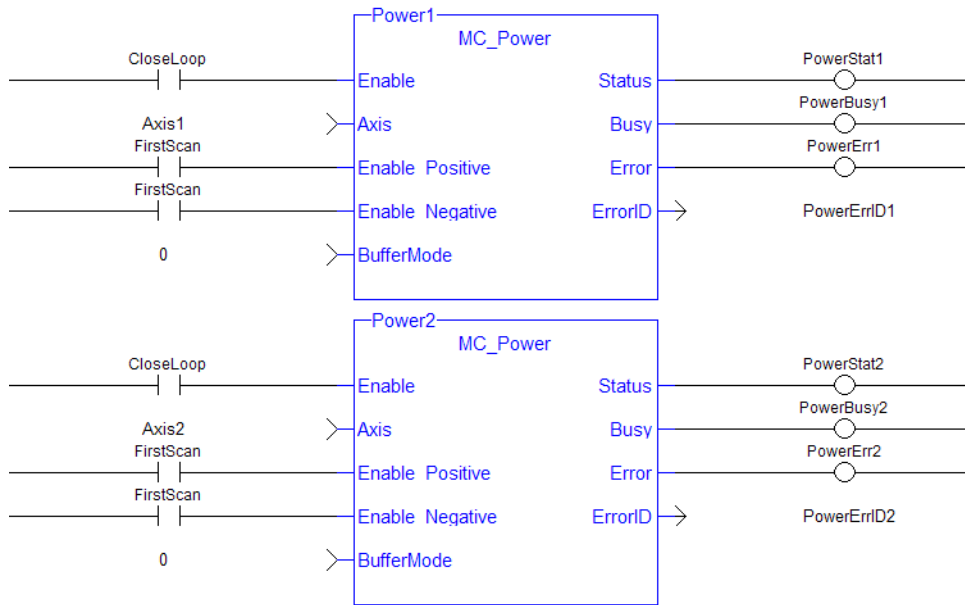
#### 2.2.1.46.5 Example

##### 2.2.1.47.6.1 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
```

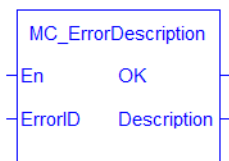
### 2.2.1.48.7.2 Ladder Diagram

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



**2.2.1.49 MC\_ErrorDescription** PLCopen ✓ Pipe Network ✓

This function converts the PLCopen error IDs into message strings which can be used for display or logging.



**Figure 1-60:** MC\_ErrorDescription Function Block

**2.2.1.50.1 Arguments**

**2.2.1.51.2.1 Inputs**

<b>En</b>	<b>Description</b>	If True, then this function will convert the Error Id into a string message
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ErrorID</b>	<b>Description</b>	Error ID generated from a PLCopen Function Block. See <a href="#">PLCopen Function Block ErrorID Output</a> for output details.
	<b>Data type</b>	INT
	<b>Range</b>	0,69
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.2.1.52.3.2 Outputs**

<b>OK</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Description</b>	<b>Description</b>	String error description
	<b>Data type</b>	STRING
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 2.2.1.53.4 Examples

#### 2.2.1.54.5.1 Structured Text

```
Description := MC_ErrorDescription(ErrorID);
```

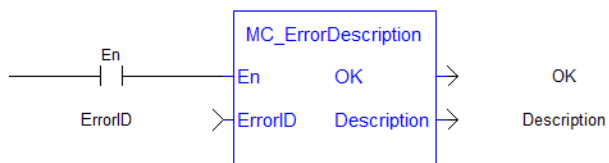
#### 2.2.1.55.6.2 IL

Not applicable

#### 2.2.1.56.7.3 Function Block



#### 2.2.1.57.8.4 Ladder Diagram



**2.2.1.58 MC\_ResetError** **2.2.1.59.1 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.

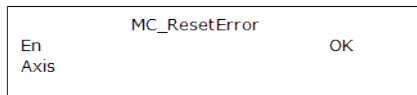


Figure 1-61: MC\_ResetError

**2.2.1.60.2 Arguments****2.2.1.61.3.1 Input**

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

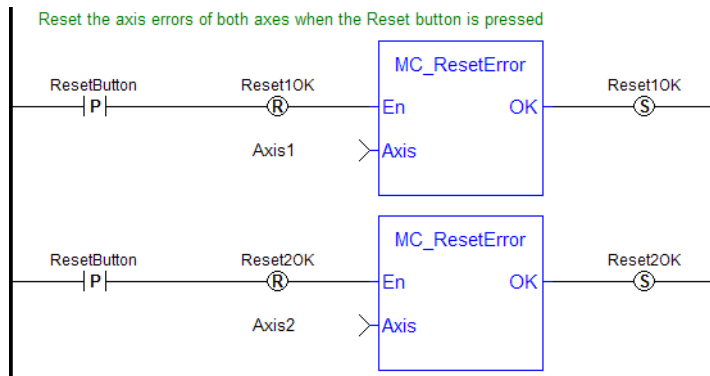
**2.2.1.62.4.2 Output**

<b>OK</b>	<b>Description</b>	Indicates the function has completed successfully
	<b>Data type</b>	BOOL

**2.2.1.63.5 Example****2.2.1.64.6.1 Structured Text**

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

### 2.2.1.65.7.2 Ladder Diagram





**2.2.1.66 MC\_Stop** PLCopen ✓

**2.2.1.67.1 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.

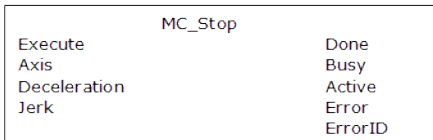


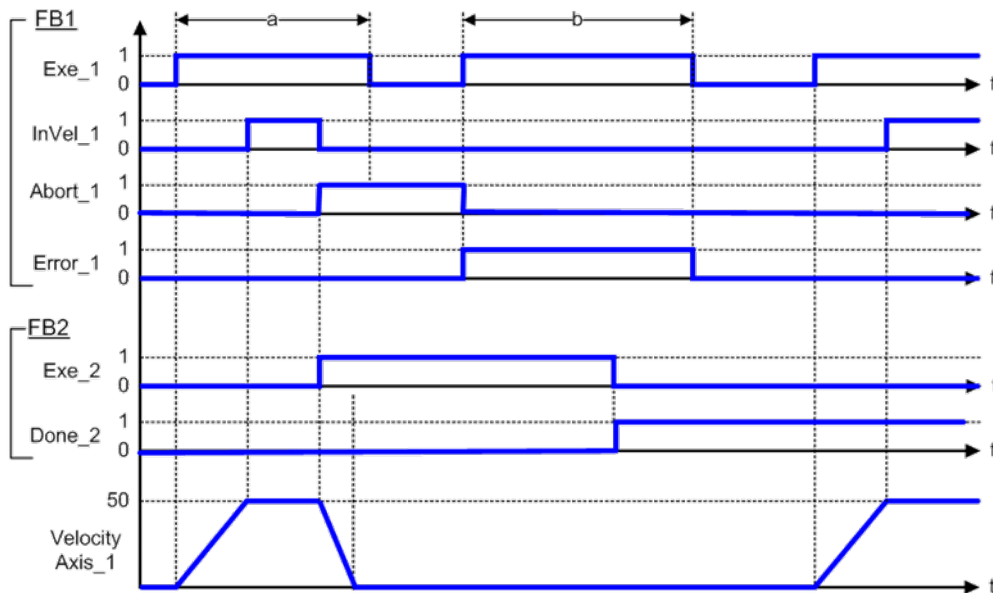
Figure 1-62: MC\_Stop

**2.2.1.68.2 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.



**2.2.1.69.3 Arguments**

**2.2.1.70.4.1 Input**

<b>Execute</b>	<b>Description</b>	Requests to stop the axis. It can be held high to prevent any other moves from being queued
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1

	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function.
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Trapezoidal: Deceleration rate S-curve: Maximum deceleration
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Trapezoidal: 0 S-curve: Constant jerk
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—

#### 2.2.1.71.5.2 Output

<b>Done</b>	<b>Description</b>	Indicates the axis has reached zero velocity AND the Execute input is low
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	High from the time the Execute input goes high until the axis reaches zero velocity AND the Execute input is low
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	High from the time the MC_Stop move becomes the active move, until the axis reaches zero velocity AND the Execute input is low
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified
	<b>Data type</b>	BOOL

<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE
	<b>Data type</b>	INT

### 2.2.1.72.6 Example

### 2.2.1.73.7.1 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

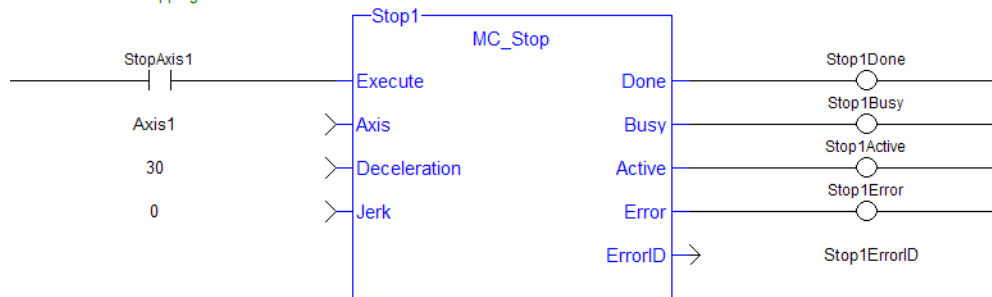
StopComplete := Inst_MC_Stop.Done;           //store the Done output
into a user defined variable

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
```

### 2.2.1.74.8.2 Ladder Diagram

Put Axis 1 into Stopping Mode



## 2.2.2 I/O Functions

This set of functions provides I/O control over TouchProbe functions.

### 2.2.2.1 MC\_AbortTrigger

#### 2.2.2.2.1 Description

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

#### 2.2.2.3.2 Arguments

##### 2.2.2.4.3.1 Input

<b>Execute</b>	<b>Description</b>	Enables execution
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Specifies the axis that was specified in the MC_TouchProbe function block which is to be aborted
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>TriggerInput</b>	<b>Description</b>	<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><b>InputID</b> INT;            0 = Capture Engine 0            1 = Capture Engine 1            Range is [0,1]            For information on configuring the capture engines, refer to <a href="#">AKD Capture Engine Configuration</a>.</p> <p><b>Direction</b> INT; 1 = rising edge, 2 = falling edge            Range is [1,2]</p> <p><b>TrigID</b> INT; 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>
		<p><b>NOTE</b></p> <p><b>TrigMode</b> INT (TriggerInput.TrigMode) is not presently supported by this function. The TriggerInput.Mode may be supported in a future software version.</p>
	<b>Data type</b>	TRIGGER_REF

<b>Range</b>	See Description above
<b>Unit</b>	n/a
<b>Default</b>	—

#### 2.2.2.5.4.2 Output

<b>Done</b>	<b>Description</b>	Function block has completed
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	Indicates the function block is currently executing
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates the function block did not complete due to an error. The ErrorID output indicates the type of error when this output is high
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	When the Error output is high, this output indicates the type of error. When the Error output is low, this output is undefined
	<b>Data type</b>	INT

#### 2.2.2.6.5 Usage

This function block is used to abort an MC\_TouchProbe function block.

#### 2.2.2.7.6 Related Functions

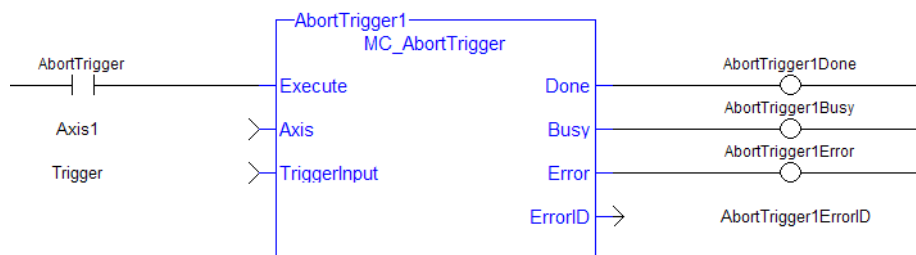
[MC\\_TouchProbe](#)

#### 2.2.2.8.7 Example

##### 2.2.2.9.8.1 Structured Text

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

#### 2.2.2.10.9.2 Ladder Diagram



**2.2.2.11 MC\_TouchProbe** PLCopen

**2.2.2.12.1 Description**

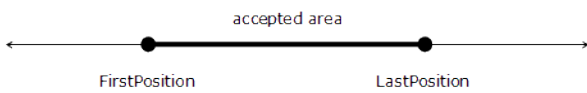
This function block arms a Fast Input and returns the latched position when the Fast Input event occurs. This function block causes no motion.

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.

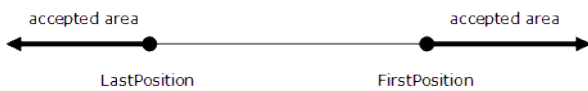
If First Position  $\leq$  LastPosition, the window in which a Fast Input is accepted is:

$$\text{FastInputPosition} \geq \text{FirstPosition} \text{ AND } \text{FastInputPosition} \leq \text{LastPosition}.$$

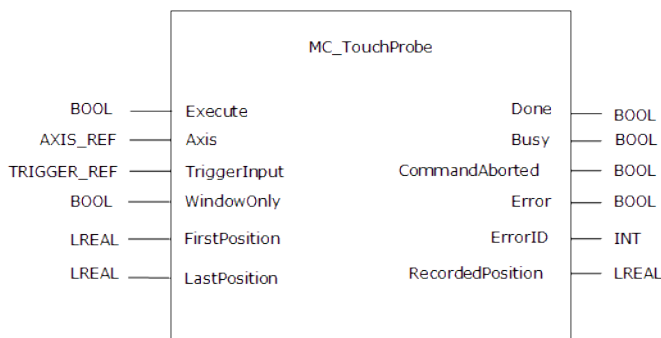


If First Position  $>$  LastPosition, the window in which a Fast Input is accepted is:

$$\text{FastInputPosition} \geq \text{FirstPosition} \text{ OR } \text{FastInputPosition} \leq \text{LastPosition}.$$



The following figure shows the ladder diagram view of the MC\_TouchProbe function block:



**Figure 1-63:** MC\_TouchProbe

**TIP**

The accuracy of captured position data depends on the travel velocity. Please see the article [MC\\_TouchProbe and Time-Based Capture](#) on [KDN](#) for more information and how to correct for timing.

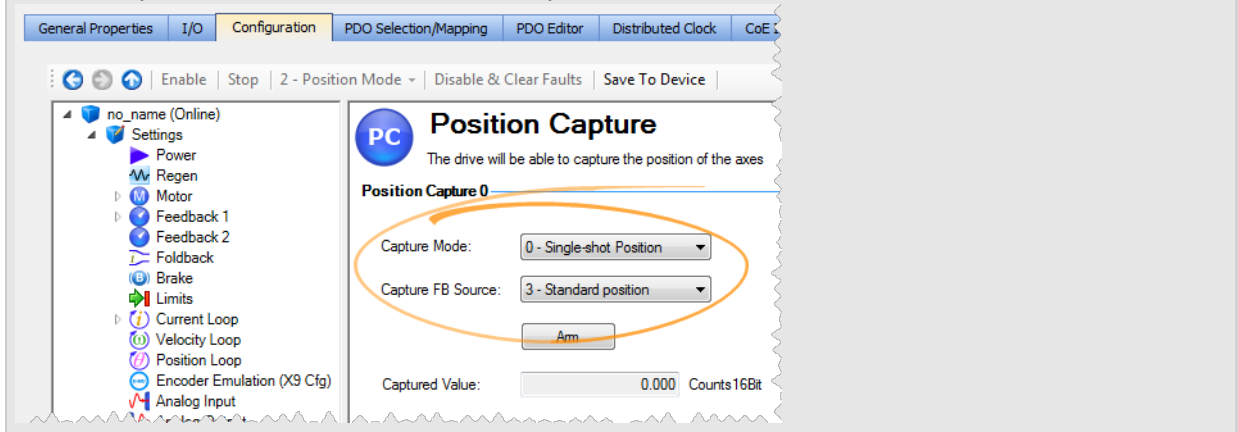
**TIP**

To use Capture Engine 1 modify the input PDOs that are used and add the Latch Position 1 parameter.

**TIP**

When using position-based capture, the proper Capture Mode and FB Source may need to be set up in the

drive. One place to do that is in the Position Capture Screen in the KAS IDE embedded WorkBench:



**TIP**

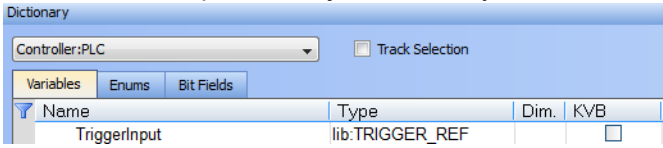
When setting up Position Capture, check the CoE-Init Command settings shown below. This is to verify they do not overwrite the corresponding drive parameters with unwanted values when the EtherCAT network initializes.

Index	Subindex	Value	Comment	Direction	Source
0x6060	0	7	Opmode	Write	ESI File
0x60C2	1	1	Cycle time	Write	ESI File
0x60C2	2	-3	Cycle exp	Write	ESI File
0x3460	1	0	Latching engine 0 config to F10, CAP0.TRIGGER=0, 0x3460-1:=0 (1 byte)	Write	ESI File
0x3460	2	1	Latching engine 1 config to F11, CAP1.TRIGGER=1, 0x3460-2:=1 (1 byte)	Write	ESI File
0x60FE	2	196608	Digital Outputs Mask, 0x60fe:=0x30000 (4 bytes)	Write	ESI File
0x36E6	0	1	Set FBUS.PARAM02 to activate synchronization with the interrupt	Write	ESI File
0x36E8	0	1	Set FBUS.PARAM04 to disabled the drive on a motion error	Write	ESI File
0x3506	0	1	Set DRV.HWENMODE to disabled the rising edge of the hardware enable from clearing the drive faults	Write	ESI File
0x35CA	0	1048576	Set UNIT.PIN to set gear IN to the correct unit conversion.	Write	ESI File
0x365F	0	0	Set UNIT.VROTARY to set the velocity units to RPM.	Write	ESI File
0x3460	3	2	Capture mode to distributed clock time (DCT), CAP0.MODE=2, 0x3460-3:=0 (1 byte)	Write	ESI File
0x3460	4	2	Capture mode to distributed clock time (DCT), CAP1.MODE=2, 0x3460-4:=0 (1 byte)	Write	ESI File
0x50E2	0	1000	Set ILKBUSFF (1.0 = 1000) UINT32	Write	ESI File
0x3498	0	1	Set FBUS.PROTECTION (available only since FW 01-07-03-000)	Write	ESI File

**2.2.2.13.2 Arguments**

**2.2.2.14.3.1 Input**

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

<b>TriggerInput</b>	<b>Description</b>	Sets up the mechanism for the capture input signal.
	<b>Data type</b>	<p>TRIGGER_REF - an instance of the TRIGGER_REF reference function must first be setup in the Project Dictionary, as seen here.</p> 
	<b>Elements</b>	<p><u>Capture Engine (drive capture engine to be used)</u>                      INT TriggerInput.InputID                      0 = Capture Engine 0                      1 = Capture Engine 1                      Range is [0,1]                      For information on configuring the capture engines, refer to <a href="#">AKD Capture Engine Configuration</a>.</p> <p><u>Trigger Direction (input signal's edge to capture)</u>                      INT TriggerInput.Direction                      1 = rising edge                      2 = falling edge                      Range is [1,2]</p> <p><u>Axis Number (where input comes from)</u>                      INT TriggerInput.TrigID                      0 = trigger axis is to be the same as Axis.AXIS_NUM.                      Range is [0,256]</p> <p><u>Trigger Mode (capture method)</u>                      INT TriggerInput.TrigMode                      0 = time based capture                      1 = position based capture. For position based capture the TrigID must be the same as the Axis_Ref.                      Range is [0,1]</p>
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>WindowOnly</b>	<b>Description</b>	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
	<b>Data type</b>	BOOL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>FirstPosition</b>	<b>Description</b>	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



	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>LastPosition</b>	<b>Description</b>	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
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—

#### 2.2.2.15.4.2 Output

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

#### 2.2.2.16.5 Usage

This function block can be used to:

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

### 2.2.2.17.6 Limitations

- Both high speed inputs cannot be used at the same time.
- TheTrigMode option is only used by MC\_TouchProbe.

### 2.2.2.18.7 Related Functions

[MC\\_AbortTrigger](#)

### 2.2.2.19.8 See Also

- [Fast Inputs with Pipe Network Motion](#)
- [Fast Inputs with PLCOpen](#)
- [Fast Homing Example with the Pipe Network Motion Engine Axis Pipe Block](#)
- [Fast Homing Example with the PLCOpen Motion Engine](#)
- [Pipe Network Registration and Fast Homing](#)
- [Registration Position Capture Example with Pipe Network Trigger Block](#)

### 2.2.2.20.9 Example

#### 2.2.2.21.10.1 Structured Text

```
(* MC_TouchProbe ST example *)
TriggerInputRef.InputID := 1; //configure InputID
TriggerInputRef.Direction := 1; //configure Direction
TriggerInputRef.TrigID := 0; //configure TrigID
TriggerInputRef.TrigMode := 0; //Capture trigger based on
distributed clock time

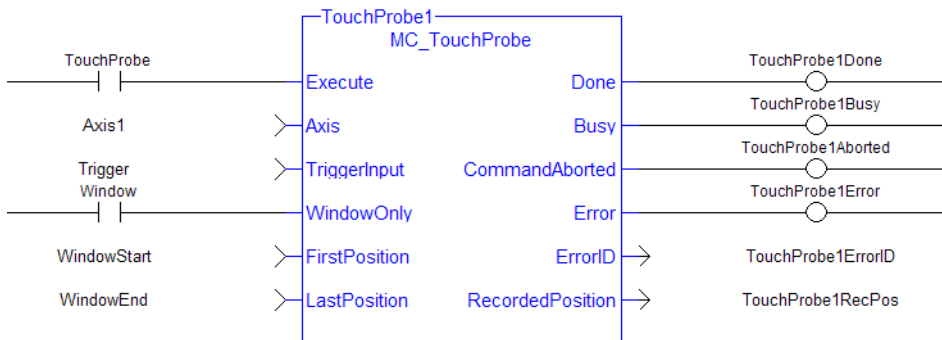
Inst_MC_TouchProbe( ArmProbe, Axis1, TriggerInputRef,
FALSE, 0.0, 0.0 );

//Inst_MC_TouchProbe is an instance of MC_TouchProbe function
block

ProbeIsDone := Inst_MC_TouchProbe.Done; //store Done output
into a user defined variable

ProbeValue := Inst_MC_TouchProbe.RecordedPosition; //store
RecordedPosition output into a user defined variable
```

#### 2.2.2.22.11.2 Ladder Diagram



## 2.2.3 Information Functions

This set of functions provides feedback and allows you to write parameters.

### 2.2.3.1 MC\_ReadActPos

#### 2.2.3.2.1 Description

The MC\_ReadActPos function block reads the actual position of the axis.

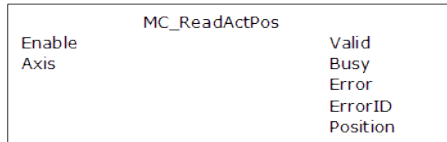


Figure 1-64: MC\_ReadActPos

#### 2.2.3.3.2 Arguments

##### 2.2.3.4.3.1 Input

<b>Enable</b>	<b>Description</b>	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
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function. )
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—

##### 2.2.3.5.4.2 Output

<b>Valid</b>	<b>Description</b>	Indicates the value at the Position output is available
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	Indicates this function block is executing
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input
	<b>Data type</b>	BOOL

<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE.
	<b>Data type</b>	INT
<b>Position</b>	<b>Description</b>	Actual position of the axis.
	<b>Unit</b>	User unit
	<b>Data type</b>	LREAL

**2.2.3.6.5 Example**

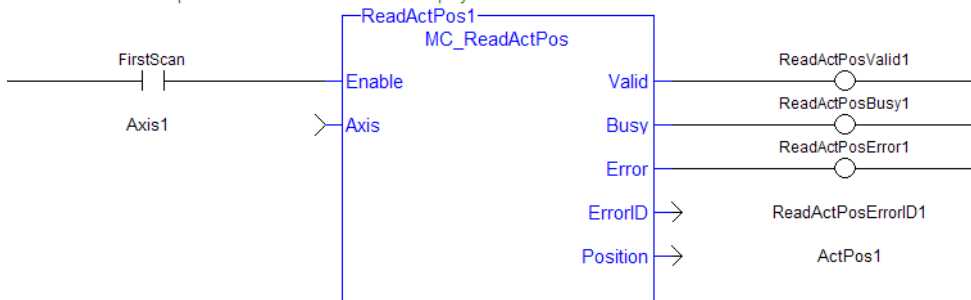
**2.2.3.7.6.1 Structured Text**

```
(* 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
```

**2.2.3.8.7.2 Ladder Diagram**

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



### 2.2.3.9 MC\_ReadActVel

#### 2.2.3.10.1 Description

The MC\_ReadActVel function block reads the actual velocity of the axis.

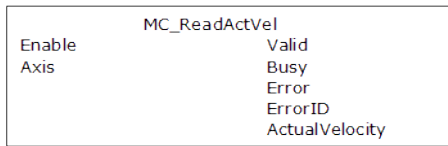


Figure 1-65: MC\_ReadActVel

#### 2.2.3.11.2 Arguments

##### 2.2.3.12.3.1 Input

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

##### 2.2.3.13.4.2 Output

<b>Valid</b>	<b>Description</b>	Indicates the value at the ActualVelocity output is available
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	Indicates this function block is executing
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE.
	<b>Data type</b>	INT
<b>ActualVelocity</b>	<b>Description</b>	Actual velocity of the axis. Please note that oscillations may be seen due to this being an instant velocity, not an average velocity.
	<b>Unit</b>	User unit/sec

**Data type** LREAL

**2.2.3.14.5 Example**

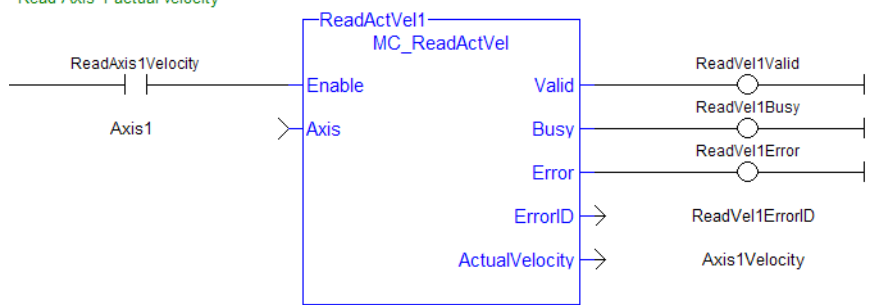
**2.2.3.15.6.1 Structured Text**

```
* 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
```

**2.2.3.16.7.2 Ladder Diagram**

Read Axis 1 actual velocity



### 2.2.3.17 MC\_ReadAxisErr

#### 2.2.3.18.1 Description

The Function Block MC\_ReadAxisErr returns the error status of the specified axis.

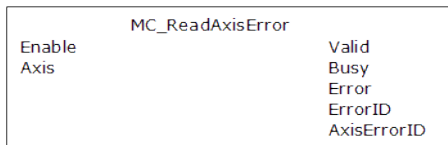


Figure 1-66: MC\_ReadAxisErr

#### 2.2.3.19.2 Arguments

##### 2.2.3.20.3.1 Input

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

##### 2.2.3.21.4.2 Output

<b>Valid</b>	<b>Description</b>	Indicates the AxisErrorID output is valid
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	Indicates this function block is executing
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE
	<b>Data type</b>	INT

<b>AxisErrorID</b>	<b>Description</b>	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.
	<b>Data type</b>	INT

Hexadecimal	Decimal	Description
0000H	0	No Error
0001H	1	User-set E-stop via MC_EStop, E-stop
0002H	2	Loss of Feedback, E-stop
0004H	4	Drive Fault, E-stop
0008H	8	Drive Communication Failure, E-stop
0400H	1024	Synchronization Error, C-stop

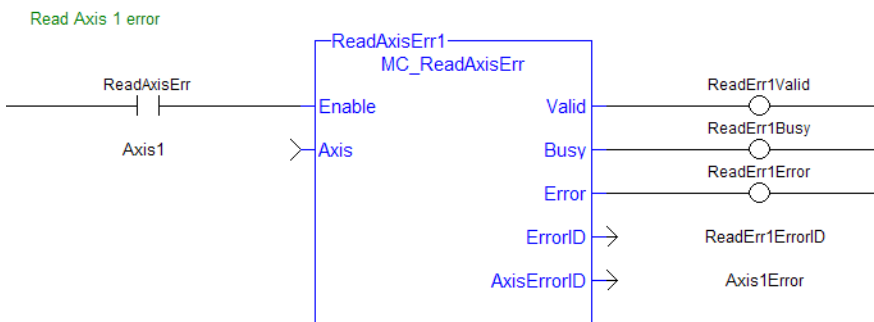
**NOTE**  
 Multiple errors can be active at the same time. For example, if a User-set E-stop and an Excess Position Error E-stop are both active, the value would be 00000011H (17 decimal).

**2.2.3.22.5 Example**

**2.2.3.23.6.1 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
```

**2.2.3.24.7.2 Ladder Diagram**





### 2.2.3.25 MC\_ReadBoolPar

#### 2.2.3.26.1 Description

The MC\_ReadBoolPar function block returns the value of the specified Boolean axis parameter.

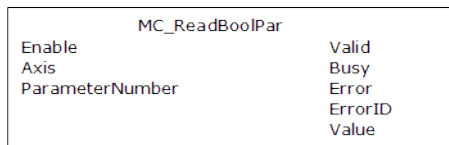


Figure 1-67: MC\_ReadBoolPar

#### 2.2.3.27.2 Arguments

##### 2.2.3.28.3.1 Input

<b>Enable</b>	<b>Description</b>	Requests to read the Boolean axis parameter
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function. )
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ParameterNumber</b>	<b>Description</b>	Parameter number, see table in <a href="#">Axis Parameters</a>
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

##### 2.2.3.29.4.2 Output

<b>Valid</b>	<b>Description</b>	Indicates the Value output is valid
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	Indicates this function block is executing
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input

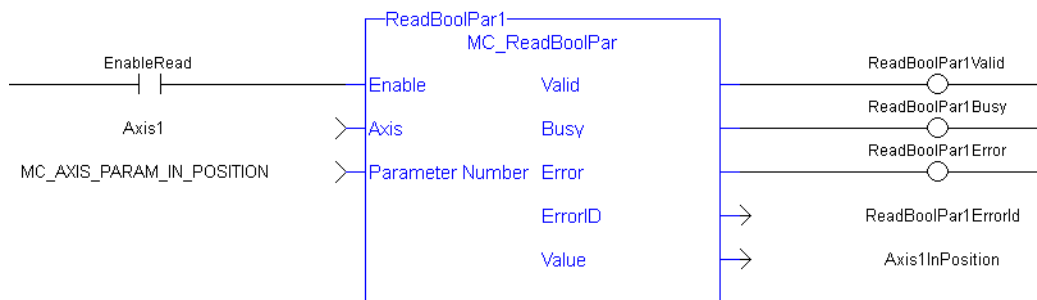
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE
	<b>Data type</b>	INT
<b>Value</b>	<b>Description</b>	State of the Boolean parameter
	<b>Data type</b>	BOOL

### 2.2.3.30.5 Example

#### 2.2.3.31.6.1 Structured Text

```
(* MC_ReadBoolPar ST example *)
Inst_MC_ReadBoolPar( EnableRead, Axis1, MC_AXIS_PARAM_IN_
POSITION );
Axis1InPosition := Inst_MC_ReadBoolPar.Value;
```

#### 2.2.3.32.7.2 Ladder Diagram



### 2.2.3.33 MC\_ReadParam

#### 2.2.3.34.1 Description

The MC\_ReadParam function block returns the value of the specified axis parameter.

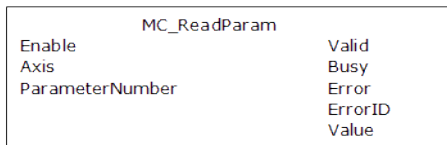


Figure 1-68: MC\_ReadParam

#### 2.2.3.35.2 Arguments

##### 2.2.3.36.3.1 Input

<b>Enable</b>	<b>Description</b>	Requests to read the axis parameter
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function. )
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ParameterNumber</b>	<b>Description</b>	Parameter number, see table in <a href="#">Axis Parameters</a>
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

##### 2.2.3.37.4.2 Output

<b>Valid</b>	<b>Description</b>	Indicates the Value output is valid
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	Indicates this function block is executing
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input

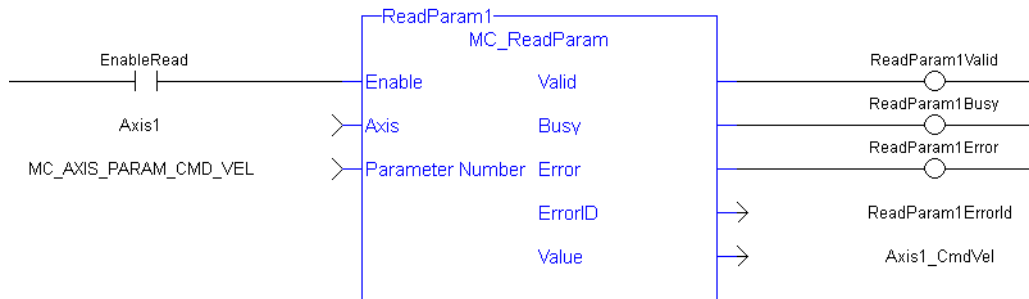
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE
	<b>Data type</b>	INT
<b>Value</b>	<b>Description</b>	Value of the parameter
	<b>Data type</b>	LREAL

### 2.2.3.38.5 Example

#### 2.2.3.39.6.1 Structured Text

```
(* MC_ReadParam ST example *)
Inst_MC_ReadParam( EnableRead, Axis1, MC_AXIS_PARAM_CMD_VEL );
Axis1_CmdVel := Inst_MC_ReadParam.Value;
```

#### 2.2.3.40.7.2 Ladder Diagram



### 2.2.3.41 MC\_ReadStatus

#### 2.2.3.42.1 Description

The function block MC\_ReadStatus returns the state of the specified axis.

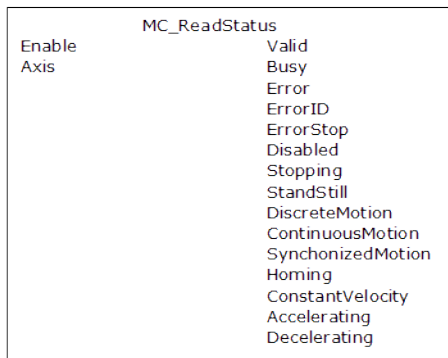


Figure 1-69: MC\_ReadStatus

#### 2.2.3.43.2 Arguments

##### 2.2.3.44.3.1 Input

<b>Enable</b>	<b>Description</b>	Requests to read and return the axis status
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function. <a href="#">click here...</a>
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—

##### 2.2.3.45.4.2 Output

<b>Valid</b>	<b>Description</b>	Indicates the outputs are valid
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	Indicates this function block is executing
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input
	<b>Data type</b>	BOOL

<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE
	<b>Data type</b>	INT
<b>ErrorStop</b>	<b>Description</b>	Indicates Error Stop state – E-stop or C-stop
	<b>Data type</b>	BOOL
<b>Disabled</b>	<b>Description</b>	Indicates Disabled state – open loop and drive is disabled
	<b>Data type</b>	BOOL
<b>Stopping</b>	<b>Description</b>	Indicates Stopping state – MC_Stop command
	<b>Data type</b>	BOOL
<b>StandStill</b>	<b>Description</b>	Indicates Stand Still state – no move, closed loop, drive enabled
	<b>Data type</b>	BOOL
<b>DiscreteMotion</b>	<b>Description</b>	Indicates Discrete Motion state – programmed endpoint move is active
	<b>Data type</b>	BOOL
<b>ContinuousMotion</b>	<b>Description</b>	Indicates Continuous Motion state – unending, single-axis move is active
	<b>Data type</b>	BOOL
<b>SynchronizedMotion</b>	<b>Description</b>	Indicates Synchronized Motion state – slave move is active
	<b>Data type</b>	BOOL
<b>Homing</b>	<b>Description</b>	Indicates Homing state – a homing cycle is currently executing
	<b>Data type</b>	BOOL
<b>ConstantVelocity</b>	<b>Description</b>	Indicates the axis is moving at a constant velocity
	<b>Data type</b>	BOOL
<b>Accelerating</b>	<b>Description</b>	Indicates the axis is accelerating
	<b>Data type</b>	BOOL
<b>Decelerating</b>	<b>Description</b>	Indicates the axis is decelerating
	<b>Data type</b>	BOOL

#### 2.2.3.46.5 Example

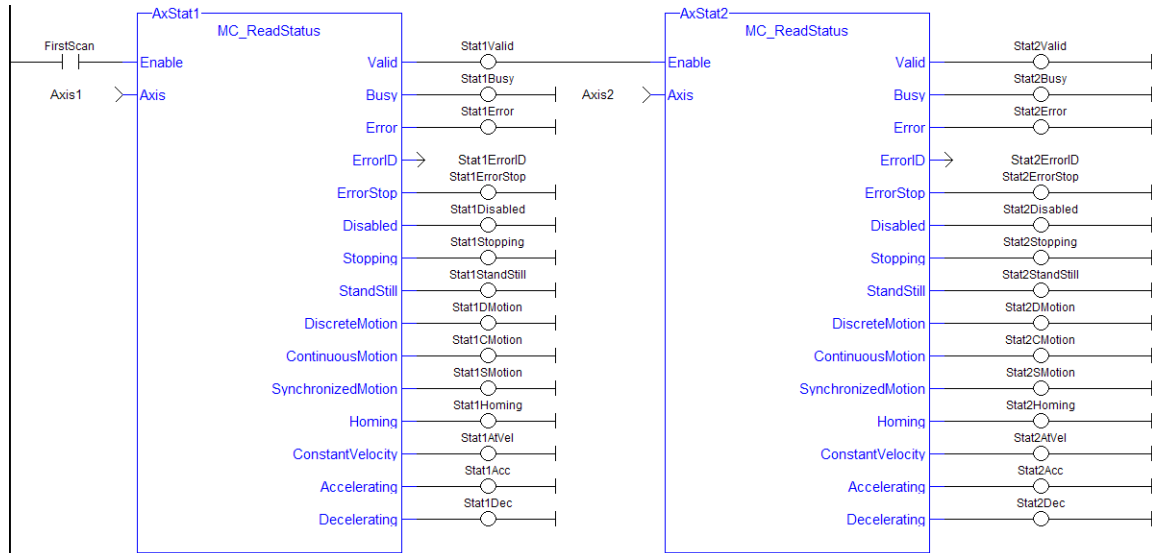
##### 2.2.3.47.6.1 Structured Text

```
(* 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
    
```

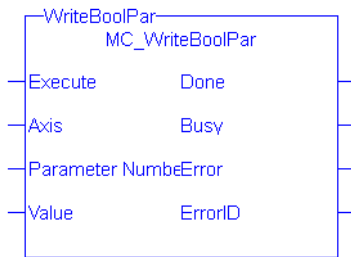
### 2.2.3.48.7.2 Ladder Diagram



### 2.2.3.49 MC\_WriteBoolPar

#### 2.2.3.50.1 Description

The MC\_WriteBoolPar function block writes the specified axis Boolean parameter.



The MC\_WriteBoolPar function block

#### 2.2.3.51.2 Arguments

##### 2.2.3.52.3.1 Input

<b>Execute</b>	<b>Description</b>	Requests to write a Boolean axis parameter
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function.
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ParameterNumber</b>	<b>Description</b>	Parameter number, see table in <a href="#">Axis Parameters</a>
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Value</b>	<b>Description</b>	State to write
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—



### 2.2.3.53.4.2 Output

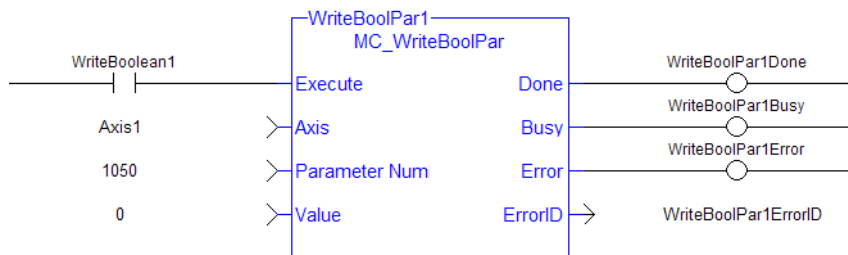
<b>Done</b>	<b>Description</b>	Indicates the Boolean parameter has been written
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	Indicates this function block is executing
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description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

### 2.2.3.54.5 Example

#### 2.2.3.55.6.1 Structured Text

```
(* MC_WriteBoolPar ST example *)
WriteBool := FALSE;
Inst_MC_WriteBoolPar( WriteReq, Axis1, 1050, WriteBool );
```

#### 2.2.3.56.7.2 Ladder Diagram



#### NOTE

Currently, MC\_WriteBoolPar does not support any parameters (1050 is an arbitrary number chosen for example)

### 2.2.3.57 MC\_WriteParam

#### 2.2.3.58.1 Description

The MC\_WriteParam function block writes the specified axis parameter.

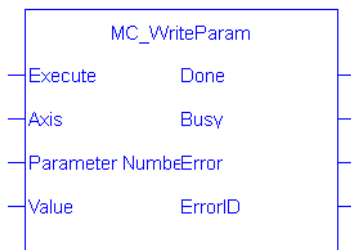


Figure 1-70: The MC\_WriteParam function block

#### 2.2.3.59.2 Arguments

##### 2.2.3.60.3.1 Input

<b>Execute</b>	<b>Description</b>	Requests to write the axis parameter
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function.
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ParameterNumber</b>	<b>Description</b>	Parameter number, see table in <a href="#">Axis Parameters</a>
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Value</b>	<b>Description</b>	Value to write
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 2.2.3.61.4.2 Output

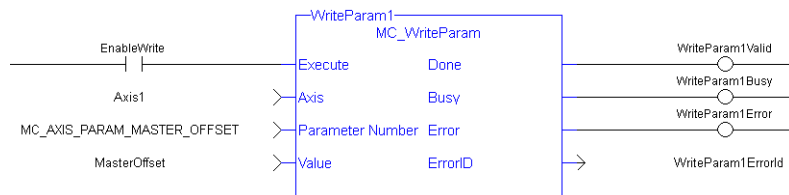
<b>Done</b>	<b>Description</b>	Indicates the parameter has been written
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	Indicates this function block is executing
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description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

### 2.2.3.62.5 Example

#### 2.2.3.63.6.1 Structured Text

```
(* MC_WriteParam ST example *)
MasterOffset := 12.34;
Inst_MC_WriteParam( EnableWrite, Axis1, MC_AXIS_PARAM_MASTER_
OFFSET, MasterOffset);
```

### 2.2.3.64.7.2 Ladder Diagram



### 2.2.4 PLCOpenMotion Functions

This set of functions provides control over an axis.

#### 2.2.4.1 MC\_Halt PLCopen ✓

##### 2.2.4.2.1 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](#).

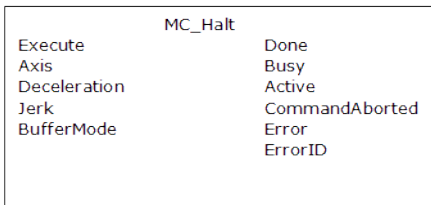


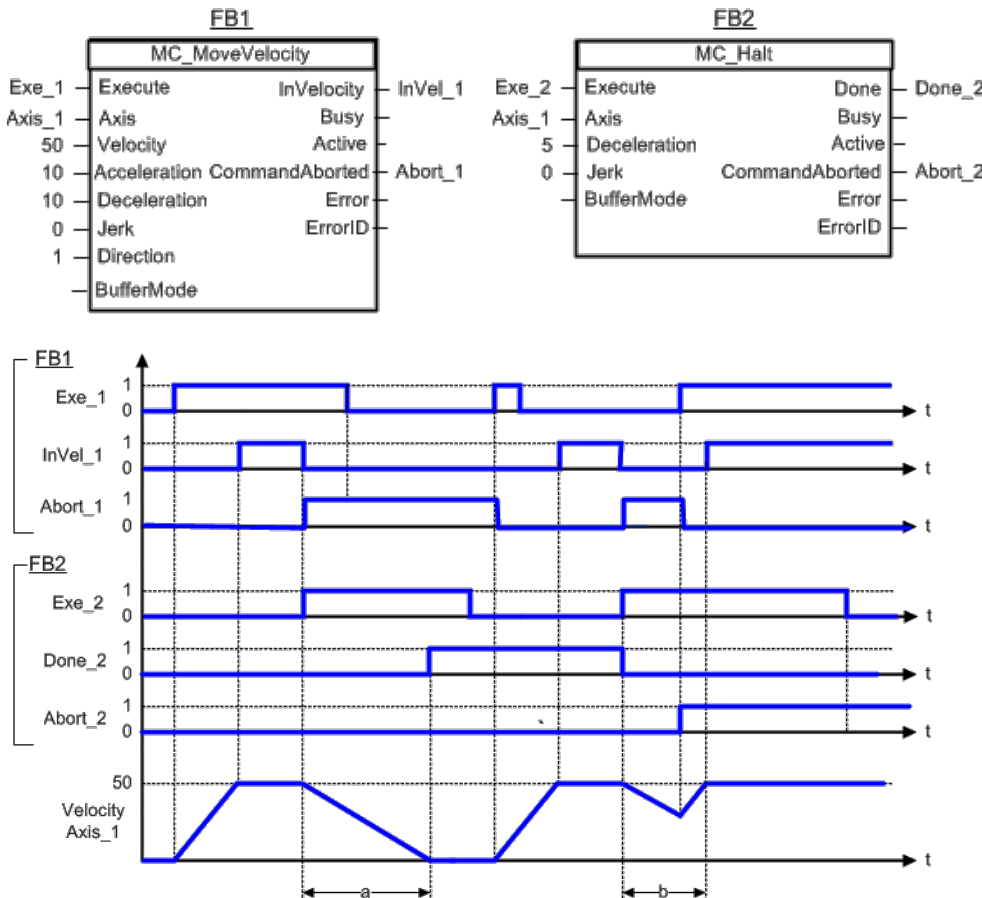
Figure 1-71: MC\_Halt

##### 2.2.4.3.2 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.



##### 2.2.4.4.3 Arguments

## 2.2.4.5.4.1 Input

<b>Execute</b>	<b>Description</b>	Requests to queue the move
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Trapezoidal: Deceleration rate S-curve: Maximum deceleration
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Trapezoidal: 0 S-curve: Constant jerk
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—
<b>BufferMode</b>	<b>Description</b>	0 = abort 1 = buffer 2 = blend to active 3 = blend to next 4 = blend to low velocity 5 = blend to high velocity
	<b>Data type</b>	SINT
	<b>Range</b>	[0,5]
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.2.4.6.5.2 Output**

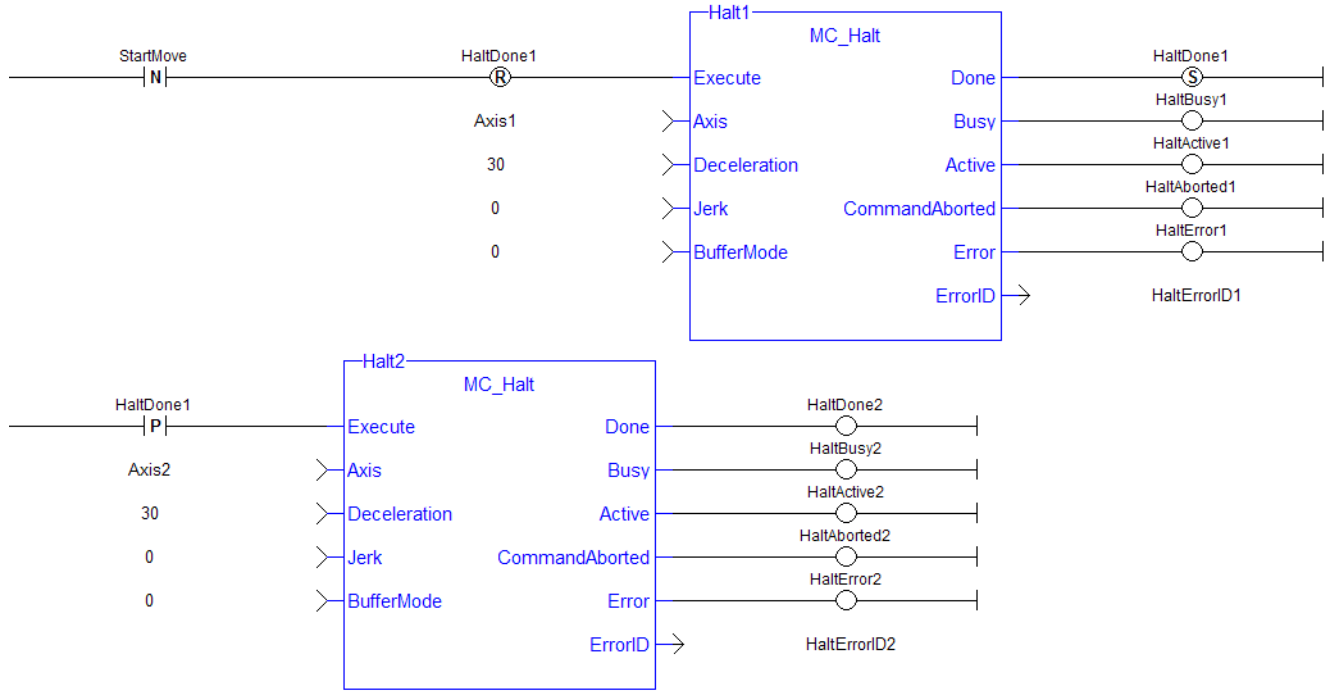
<b>Done</b>	<b>Description</b>	Indicates the move completed successfully. The Command Position has reached the endpoint.
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	High from the moment the Execute input is one-shot to the time the move is ended
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description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

**2.2.4.7.6 Example****2.2.4.8.7.1 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
```

**2.2.4.9.8.2 Ladder Diagram**

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



2.2.4.10 MC\_MoveAbsolute PLCopen ✔

2.2.4.11.1 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.

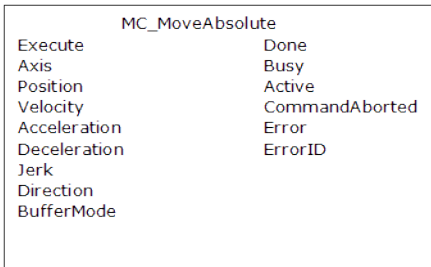
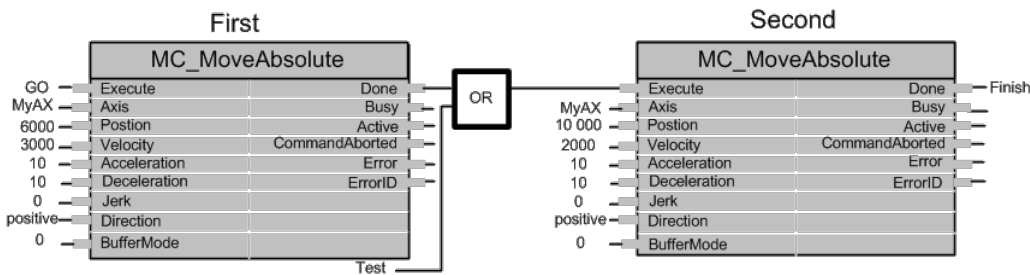


Figure 1-72: MC\_MoveAbsolute

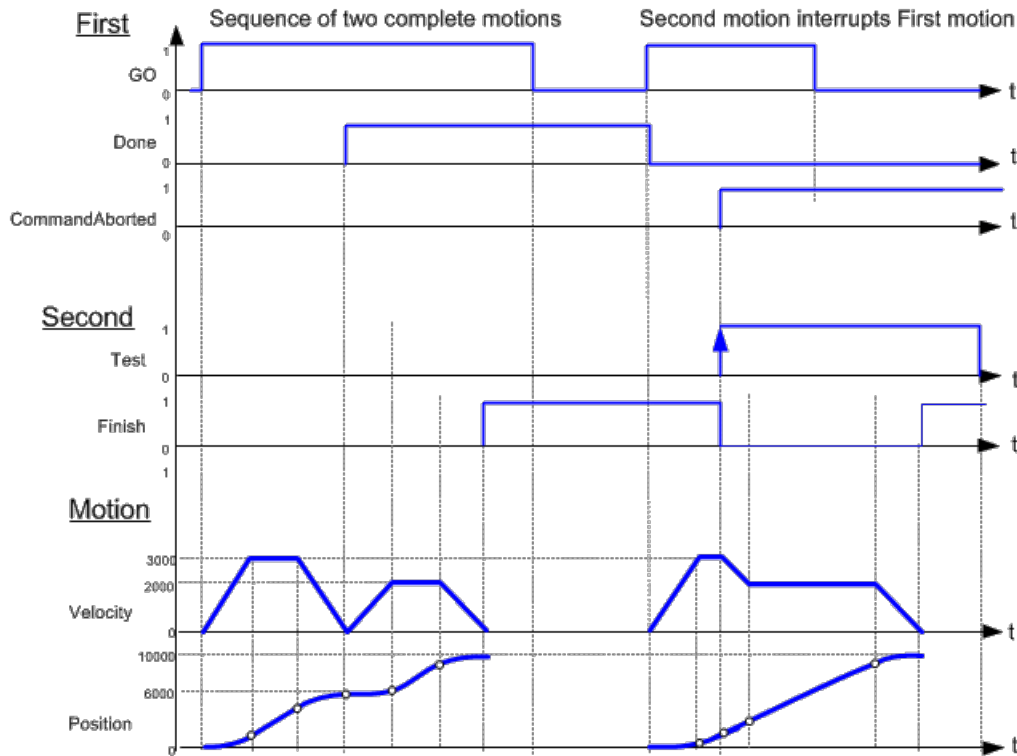
2.2.4.12.2 Time Diagram

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







### 2.2.4.13.3 Arguments

#### 2.2.4.14.4.1 Input

<b>Execute</b>	<b>Description</b>	Requests to queue the move
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Position</b>	<b>Description</b>	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.

	<b>Data type</b>	LREAL
	<b>Range</b>	[see Description]
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>Velocity</b>	<b>Description</b>	Velocity setpoint
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Trapezoidal: Acceleration rate S-curve: Maximum acceleration If Acceleration is not valid, ErrorID is set to 21 Selection of Acceleration and Jerk Parameters for Function Blocks
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Trapezoidal: Deceleration rate S-curve: Unused
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Trapezoidal: 0 S-curve: Constant jerk If Jerk is not valid, ErrorID is set to <a href="#">21</a> Selection of Acceleration and Jerk Parameters for Function Blocks
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>

	<b>Default</b>	—												
<b>Direction</b>	<b>Description</b>	<p>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.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>no direction specification</td> </tr> <tr> <td>1</td> <td>positive direction. The axis travels in the positive direction to the endpoint</td> </tr> <tr> <td>2</td> <td>shortest distance. The axis travels in the direction that provides the shortest distance to the endpoint</td> </tr> <tr> <td>3</td> <td>negative direction. The axis travels in the negative direction to the endpoint</td> </tr> <tr> <td>4</td> <td>last direction. The axis travels to the endpoint in the same direction as its previous move</td> </tr> </tbody> </table> <p>If the Position input is the same as the axis's current position, then:</p> <ul style="list-style-type: none"> <li>when Direction = <b>2</b> (shortest distance), the axis does not move and the Done output goes high indicating that the move has been completed.</li> <li>when Direction = <b>1, 3, or 4</b>, the axis travels in the specified direction, through one rollover cycle, and arrives back at the same position.</li> </ul>	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
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													
	<b>Data type</b>	SINT												
	<b>Range</b>	[0,4]												
	<b>Unit</b>	n/a												
	<b>Default</b>	—												
<b>BufferMode</b>	<b>Description</b>	<p>0 = abort  1 = buffer  2 = blend to active  3 = blend to next  4 = blend to low velocity  5 = blend to high velocity</p>												
	<b>Data type</b>	SINT												
	<b>Range</b>	[0,5]												
	<b>Unit</b>	n/a												
	<b>Default</b>	—												

#### 2.2.4.15.5.2 Output

<b>Done</b>	<b>Description</b>	<p>Indicates the move completed successfully. The Command Position has reached the endpoint.</p>
	<b>Data type</b>	BOOL

<b>Busy</b>	<b>Description</b>	High from the moment the Execute input is one-shot to the time the move is ended
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	Indicates this move is the active move
	<b>Data type</b>	BOOL
<b>CommandAborted</b>	<b>Description</b>	Indicates the move was aborted
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified or the move was terminated due to an error
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE
	<b>Data type</b>	INT

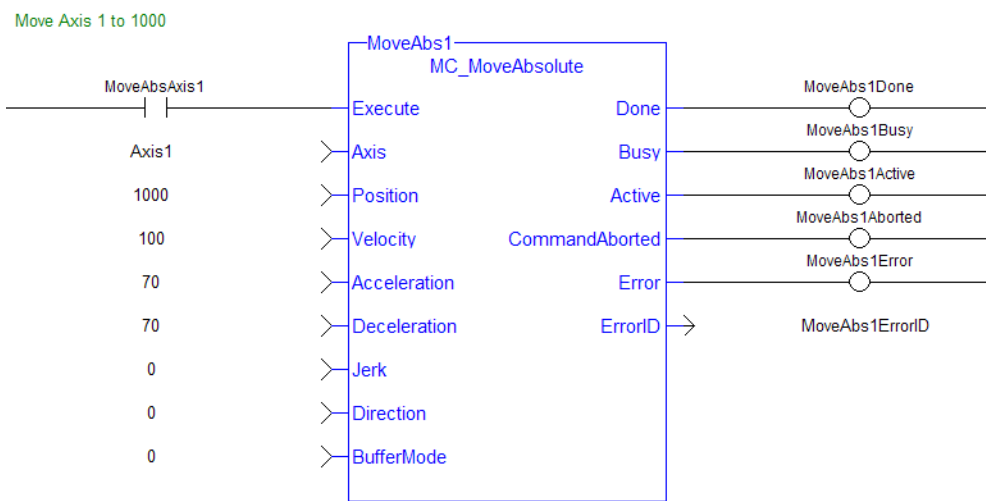
**2.2.4.16.6 Example**

**2.2.4.17.7.1 Structured Text**

```

(* MC_MoveAbsolute S
T example *)
Inst_MC_MoveAbsolute( MovAbsReq, Axis1, 1234.567, 100.0, 100.0,
100.0, 0, 0, 0 ); //instance of MC_MoveAbsolute
MovAbsDone := Inst_MC_MoveAbsolute.Done; //store done output
into user defined variable
MovAbsBusy := Inst_MC_MoveAbsolute.Busy;
MovAbsActive := Inst_MC_MoveAbsolute.Active;
MovAbsAborted := Inst_MC_MoveAbsolute.CommandAborted;
MovAbsError := Inst_MC_MoveAbsolute.Error;
MovAbsErrID := Inst_MC_MoveAbsolute.ErrorID;
    
```

**2.2.4.18.8.2 Ladder Diagram**



2.2.4.19 MC\_MoveAdditive PLCopen ✓

2.2.4.20.1 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.

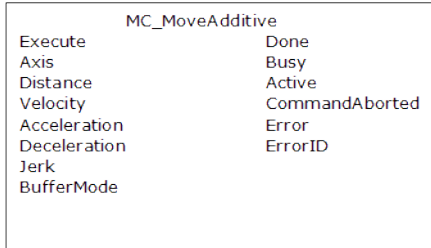
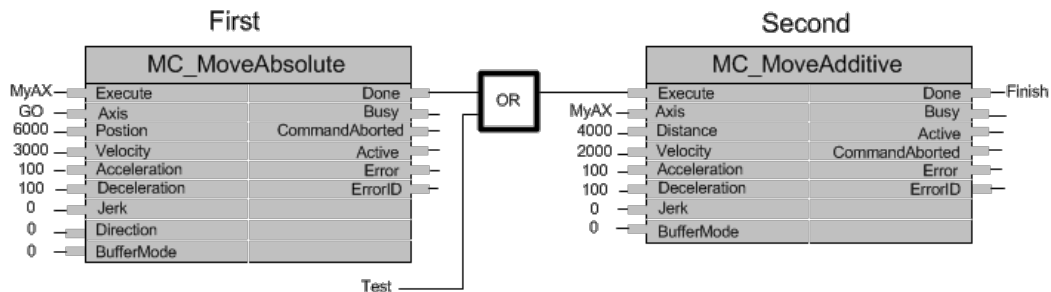


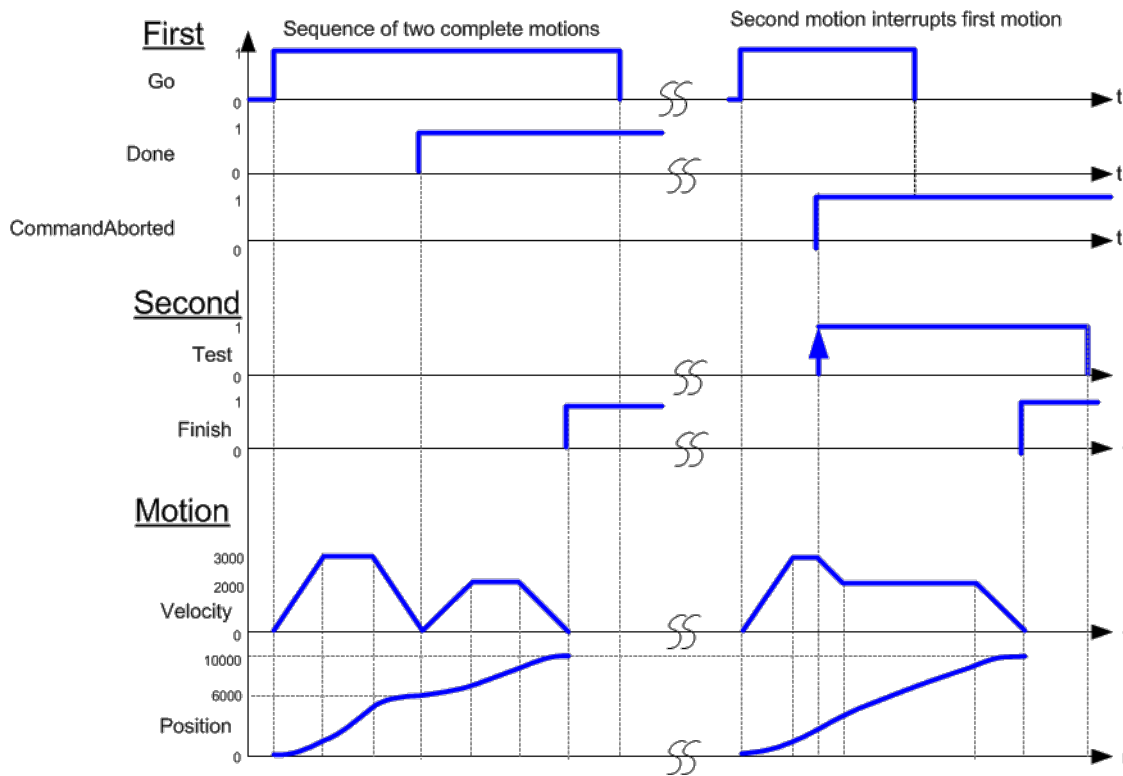
Figure 1-73: MC\_MoveAdditive

2.2.4.21.2 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





### 2.2.4.22.3 Arguments

#### 2.2.4.23.4.1 Input

<b>Execute</b>	<b>Description</b>	Requests to queue the move
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function. )
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Distance</b>	<b>Description</b>	Distance to add to the endpoint of the previous move
	<b>Data type</b>	REAL
	<b>Range</b>	—
	<b>Unit</b>	User unit

	<b>Default</b>	—
<b>Velocity</b>	<b>Description</b>	Velocity setpoint
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Trapezoidal: Deceleration rate S-curve: Unused
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Trapezoidal: 0 S-curve: Constant jerk
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—
<b>BufferMode</b>	<b>Description</b>	0 = abort 1 = buffer 2 = blend to active 3 = blend to next 4 = blend to low velocity 5 = blend to high velocity
	<b>Data type</b>	SINT
	<b>Range</b>	[0,5]
	<b>Unit</b>	n/a

<b>Default</b>	—
----------------	---

#### 2.2.4.24.5.2 Output

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

#### 2.2.4.25.6 Example

##### 2.2.4.26.7.1 Structured Text

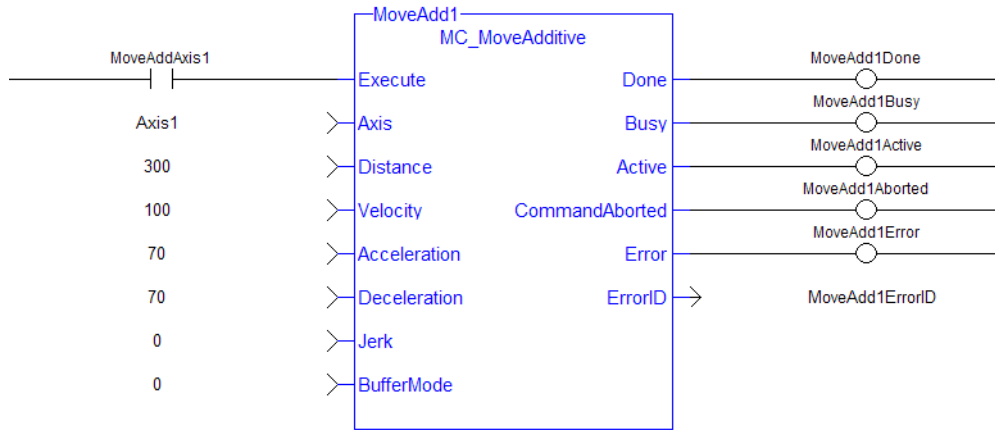
```
(* MC_MoveAdditive ST example *)

Inst_MC_MoveAdditive( MovAddReq, Axis1, 123.456, 100.0, 100.0,
100.0, 0, 0 );
  //Inst_MC_MoveAdditive is an instance of MC_MoveAdditive
function block
MovAddDone := Inst_MC_MoveAdditive.Done;
  //store Done output into user defined variable
```

#### 2.2.4.27.8.2 Ladder Diagram



Move Axis 1 an additive distance of 300



2.2.4.28 MC\_MoveRelative PLCopen ✔

2.2.4.29.1 Description

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

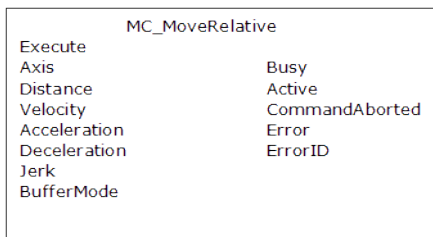
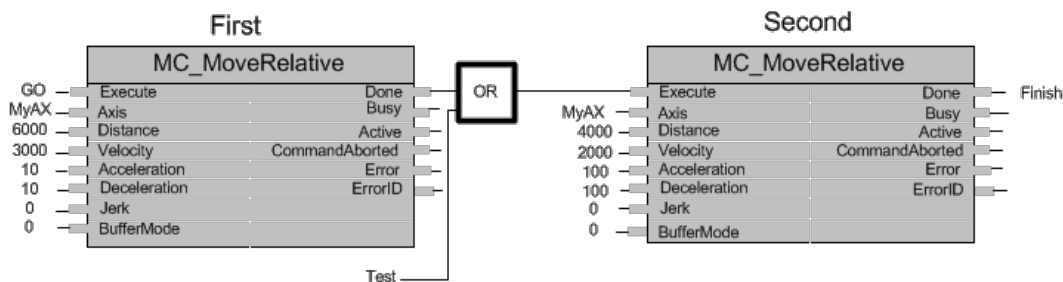


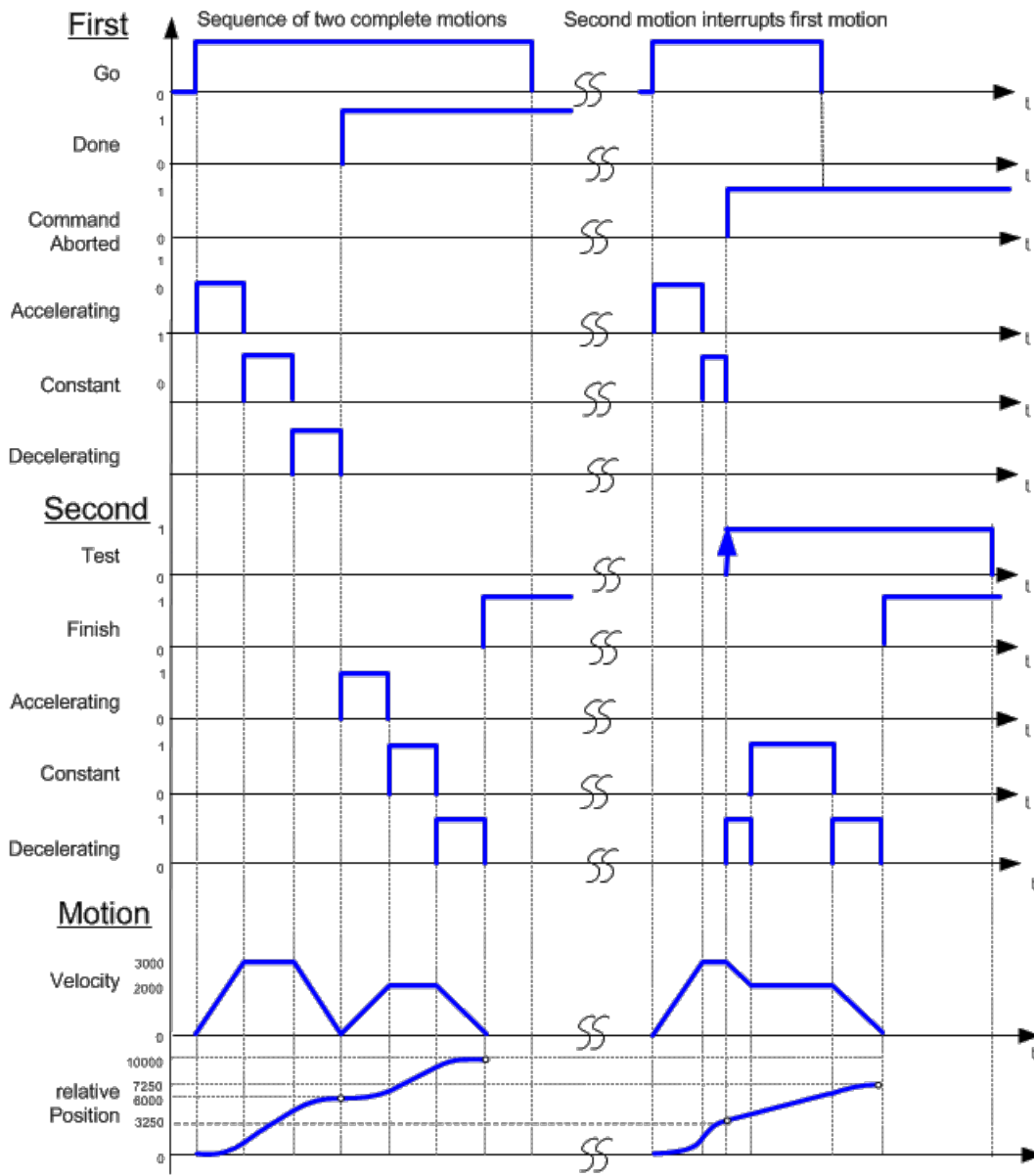
Figure 1-74: MC\_MoveRelative

2.2.4.30.2 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





### 2.2.4.31.3 Arguments

#### 2.2.4.32.4.1 Input

<b>Execute</b>	<b>Description</b>	Requests to queue the move
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function.
	<b>Data type</b>	AXIS_REF

	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Distance</b>	<b>Description</b>	Distance
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>Velocity</b>	<b>Description</b>	Velocity setpoint
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Trapezoidal: Deceleration rate S-curve: Unused
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Trapezoidal: 0 S-curve: Constant jerk
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>

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

#### 2.2.4.33.5.2 Output

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

#### 2.2.4.34.6 Example

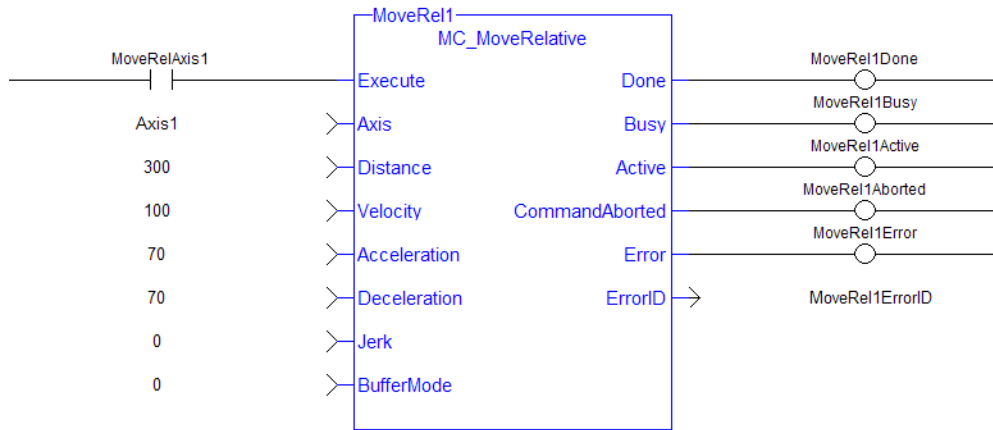
#### 2.2.4.35.7.1 Structured Text

```
(* 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](#)

### 2.2.4.36.8.2 Ladder Diagram

Move Axis 1 a relative distance of 300



2.2.4.37 MC\_MoveSuperimp 

2.2.4.38.1 Description

This function block provides the ability to cause additional axis motion superimposed upon a currently executing move. A superimposed move is executed like an "MC\_MoveRelative" (p. 458) move using the specified **Distance**, **Velocity** (i.e. VelocityDiff), **Acceleration**, **Deceleration**, and **Jerk** values. The interpolated command generated by a superimposed move is added to the command of the currently executing move. Subsequent calls to MC\_MoveSuperimp can abort or blend to an executing MC\_MoveSuperimp move.

This function block provides a way to smoothly apply a shift in axis position while it is executing a move.

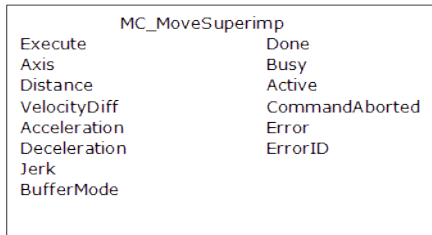
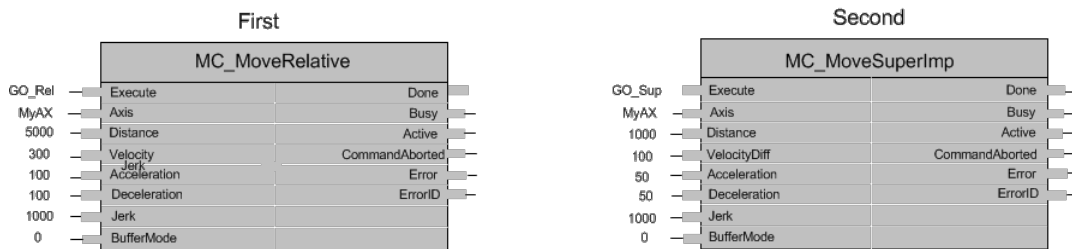
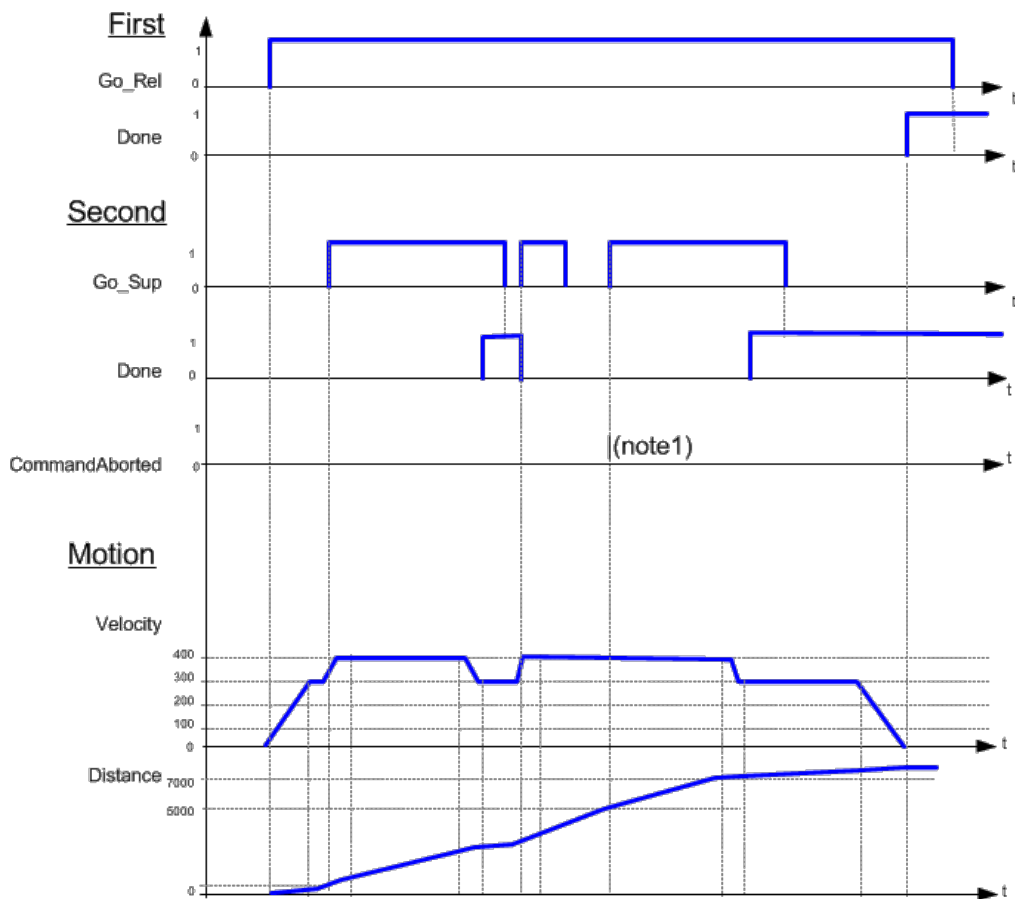


Figure 1-75: MC\_MoveSuperimp

2.2.4.39.2 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

**2.2.4.40.3 Arguments**

**2.2.4.41.4.1 Input**

<b>Execute</b>	<b>Description</b>	Requests to queue the superimposed move
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function.
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—



<b>Distance</b>	<b>Description</b>	Distance
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>VelocityDiff</b>	<b>Description</b>	Velocity rate
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Trapezoidal: Deceleration rate S-curve: Unused
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Trapezoidal: 0 S-curve: Constant jerk
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—

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

#### 2.2.4.42.5.2 Output

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

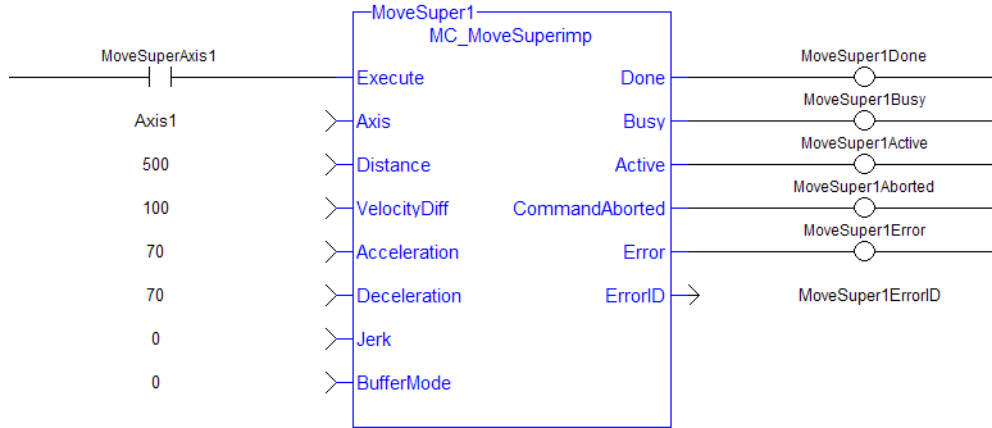
#### 2.2.4.43.6 Example

##### 2.2.4.44.7.1 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
```

#### 2.2.4.45.8.2 Ladder Diagram

Move Axis 1 a superimposed distance of 500



**2.2.4.46 MC\_MoveVelocity** PLCopen

**2.2.4.47.1 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.

**TIP**

Consider using the "MC\_MoveContVel" (p. 472) function block. It is more flexible and allows for the continuous update of motion parameters.

**Figure 1-76:** MC\_MoveVelocity

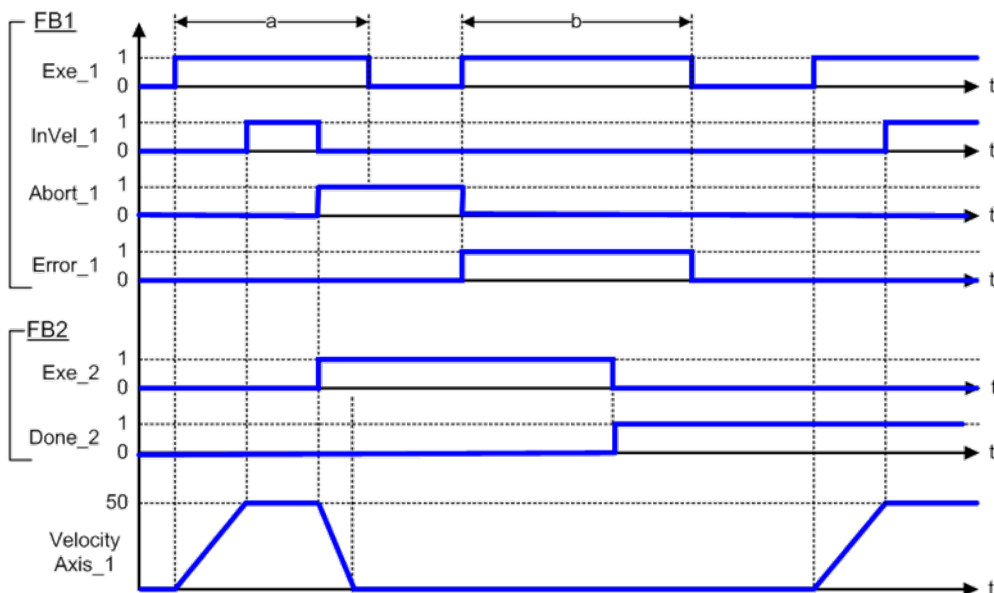
"MC\_MoveContVel" (p. 472)

**2.2.4.48.2 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.



**2.2.4.49.3 Arguments**

**2.2.4.50.4.1 Input**

<b>Execute</b>	<b>Description</b>	Requests to queue the move
	<b>Data type</b>	BOOL
	<b>Range</b>	False, True
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>Axis</b>	<b>Description</b>	Identifier of a declared instance of the AXIS_REF library function. For more details,.
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Speed</b>	<b>Description</b>	The target axis speed. Direction is specified by the <b>Direction</b> input parameter.
	<b>Data type</b>	LREAL
	<b>Range</b>	Positive values
	<b>Unit</b>	User unit/sec
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	<b>Data type</b>	LREAL
	<b>Range</b>	Positive values
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Trapezoidal: Deceleration rate S-curve: Unused
	<b>Data type</b>	LREAL
	<b>Range</b>	Positive values
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Trapezoidal: 0 S-curve: Constant jerk
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—

<b>Direction</b>	<b>Description</b>	A 0 or False value specifies that the axis should move in the positive direction. A 1 or True value specifies that the axis should move in the negative direction.
	<b>Data type</b>	BOOL
	<b>Range</b>	False, True
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>BufferMode</b>	<b>Description</b>	The specified buffer mode. For more information see " <a href="#">Buffer Modes</a> ".
	<b>Data type</b>	SINT
	<b>Range</b>	MC_BUFFER_MODE_ABORTING MC_BUFFER_MODE_BUFFERED MC_BUFFER_MODE_BLENDED_PREVIOUS MC_BUFFER_MODE_BLENDED_NEXT MC_BUFFER_MODE_BLENDED_LOW MC_BUFFER_MODE_BLENDED_HIGH
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 2.2.4.51.5.2 Output

<b>InVelocity</b>	<b>Description</b>	Indicates the command velocity has reached the programmed velocity
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	High from the moment the Execute input is one-shot to the time the move is ended
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	Indicates this move is the active move
	<b>Data type</b>	BOOL
<b>CommandAborted</b>	<b>Description</b>	Indicates the move was aborted
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified or the move was terminated due to an error
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See table in <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data type</b>	INT

### 2.2.4.52.6 Example

#### 2.2.4.53.7.1 Structured Text

```
(* MC_MoveVelocity ST example *)  
Inst_MC_MoveVelocity( MovVelReq , Axis1, 200.0, 100.0, 100.0, 0,  
True, MC_BUFFER_MODE_ABORTING );
```

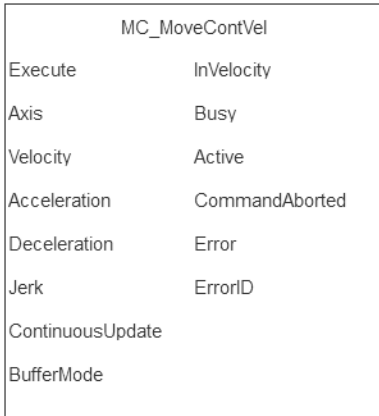
#### 2.2.4.54.8.2 Ladder Diagram

**2.2.4.55 MC\_MoveContVel** PLCopen

**2.2.4.56.1 Description**

This function block performs a single-axis non-ending move at a specified velocity with the option of continually updating the ongoing motion with the current input parameters. After **MC\_MoveContVel** execution begins (**Execute** input - low to high), follow up changes to input parameters immediately affect the ongoing motion, without requiring an additional low to high transition on the **Execute** input.

This type of move can be terminated with the **"MC\_Halt"** (p. 444) function block or by aborting it with another move.



**Figure 1-77:** MC\_MoveContVel

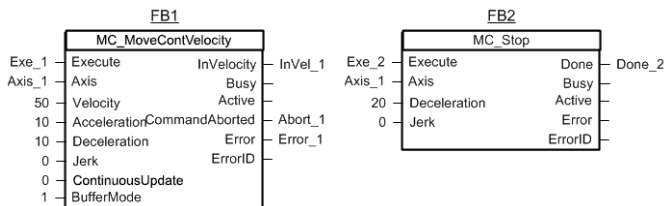
["MC\\_MoveVelocity"](#) (p. 468)

**2.2.4.57.2 Time Diagram**

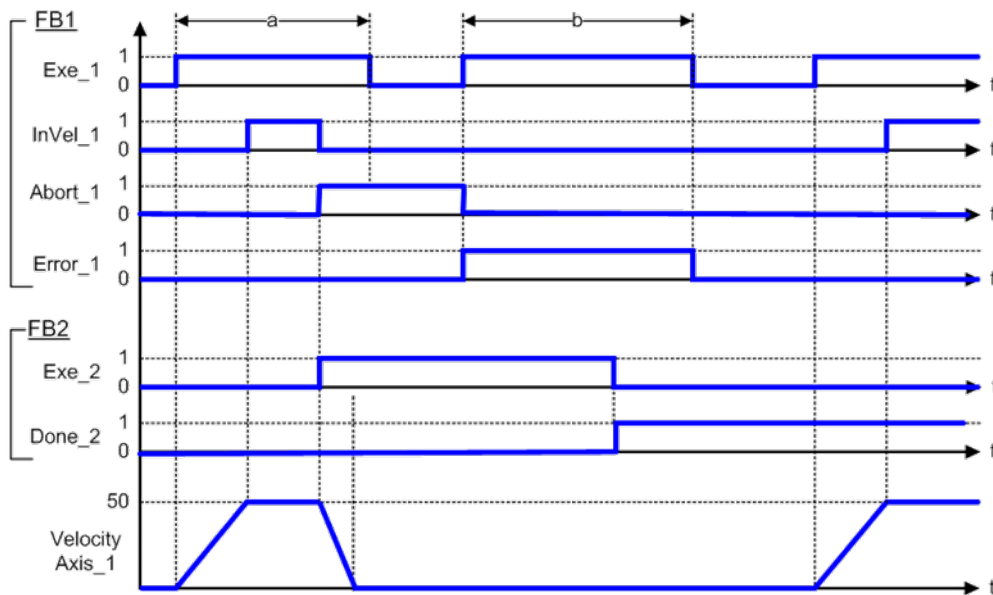
The example below shows the behavior of the combination of a **"MC\_Stop"** (p. 417) function block with a **MC\_MoveContVel** function block.

- A rotating axis is ramped down with FB2 **"MC\_Stop"** (p. 417)
- The axis rejects motion commands as long as **"MC\_Stop"** (p. 417) parameter **"Execute"** = TRUE

FB1 **MC\_MoveContVel** reports an error indicating the busy **"MC\_Stop"** (p. 417) command.







### 2.2.4.58.3 Arguments

#### 2.2.4.59.4.1 Input

<b>Execute</b>	<b>Description</b>	Requests to queue the move.
	<b>Data type</b>	BOOL
	<b>Range</b>	False, True
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Identifier of a declared instance of the the AXIS_REF library function (for more details, .
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Velocity</b>	<b>Description</b>	The target axis velocity. Negative values of velocity will move the axis in the negative direction. Positive values will move the axis in the positive direction. A value of 0 is valid and indicates a deceleration to zero velocity.
	<b>Data type</b>	LREAL
	<b>Range</b>	All finite values
	<b>Unit</b>	User unit/sec
	<b>Default</b>	—

<b>Acceleration</b>	<b>Description</b>	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	<b>Data type</b>	LREAL
	<b>Range</b>	Positive values
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Trapezoidal: Deceleration rate S-curve: Unused
	<b>Data type</b>	LREAL
	<b>Range</b>	Positive values
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Trapezoidal: 0 S-curve: Constant jerk
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—
<b>ContinuousUpdate</b>	<b>Description</b>	Determines if the inputs of the function block are re-evaluated every cycle or if they are only evaluated on the rising edge of <b>Execute</b> . If TRUE when the function block is triggered (on the rising edge of <b>Execute</b> ), the function block uses the current updated values of the input variables and apply it to the ongoing movement of the axis. This will continue as long as <b>ContinuousUpdate</b> stays TRUE. The impact of <b>ContinuousUpdate</b> ends as soon as the function block is no longer busy ( <b>Busy</b> output is FALSE) or <b>ContinuousUpdate</b> is set to FALSE.
	<b>Data type</b>	BOOL
	<b>Range</b>	False, True
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>BufferMode</b>	<b>Description</b>	The specified buffer mode. For more information see " <a href="#">Buffer Modes</a> ".
	<b>Data type</b>	SINT

<b>Range</b>	MC_BUFFER_MODE_ABORTING MC_BUFFER_MODE_BUFFERED MC_BUFFER_MODE_BLENDING_PREVIOUS MC_BUFFER_MODE_BLENDING_NEXT MC_BUFFER_MODE_BLENDING_LOW MC_BUFFER_MODE_BLENDING_HIGH
<b>Unit</b>	n/a
<b>Default</b>	—

#### 2.2.4.60.5.2 Output

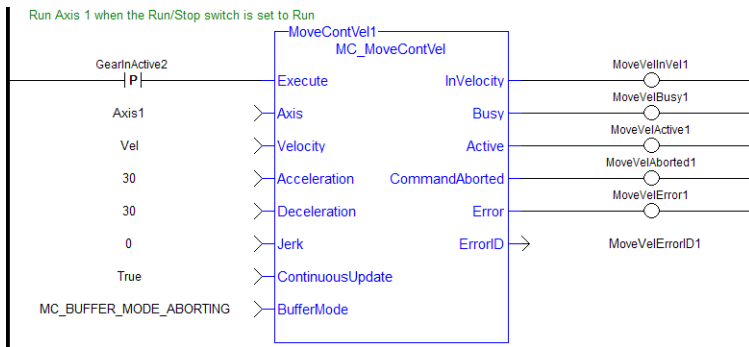
<b>InVelocity</b>	<b>Description</b>	Indicates the command velocity has reached the programmed velocity
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	High from the moment the Execute input is one-shot to the time the move is ended
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	Indicates this move is the active move
	<b>Data type</b>	BOOL
<b>CommandAborted</b>	<b>Description</b>	Indicates the move was aborted
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified or the move was terminated due to an error
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See table in <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data type</b>	INT

#### 2.2.4.61.6 Example

##### 2.2.4.62.7.1 Structured Text

```
(* MC_MoveContVel ST example *)
Inst_MC_MoveContVel( MovVelReq , Axis1, Vel, 100.0, 100.0, 0,
True, MC_BUFFER_MODE_ABORTING );
```

#### 2.2.4.63.8.2 Freeform Ladder Diagram



**2.2.4.64 MC\_SetOverride** PLCopen **2.2.4.65.1 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 (of a "MC\_MoveAbsolute" (p. 448), "MC\_MoveAdditive" (p. 453), "MC\_MoveRelative" (p. 458), "MC\_MoveSuperimp" (p. 463), or "MC\_MoveVelocity" (p. 468) function block) to determine the command velocity:

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

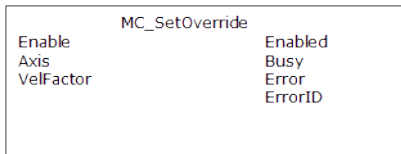


Figure 1-78: MC\_SetOverride

**2.2.4.66.2 Arguments****2.2.4.67.3.1 Input**

<b>Enable</b>	<b>Description</b>	Request to write the override factors
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function.
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>VelFactor</b>	<b>Description</b>	Velocity override factor
	<b>Data type</b>	REAL
	<b>Range</b>	[0.0, 2.0]
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.2.4.68.4.2 Output**

<b>Enabled</b>	<b>Description</b>	Indicates the override values have been written
----------------	--------------------	---

	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	Indicates the function block is executing
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input is specified
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE
	<b>Data type</b>	INT

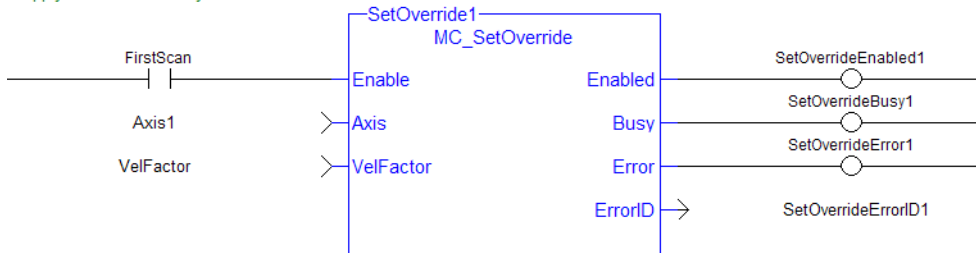
**2.2.4.69.5 Example**

**2.2.4.70.6.1 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
```

**2.2.4.71.7.2 Ladder Diagram**

Apply the Axis 1 velocity override factor



## 2.2.5 Profile Functions

This set of functions provides commands for slave axes, such as cams and gearing.

### 2.2.5.1 MC\_CamIn PLCopen

#### 2.2.5.2.1 Description

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

This function block is used to either initiate a new MC\_CamIn move or to resume a previously programmed MC\_CamIn move. Refer to "MC\_CamStartPos" (p. 493) and "MC\_CamResumePos" (p. 490) for information on positioning the slave axis prior to calling MC\_CamIn.



Figure 1-79: MC\_CamIn

#### Aborting Camming

There are two common options to stop camming after MC\_CamIn has been called.

- "MC\_CamOut" (p. 486) will continue motion at the instantaneous final actual velocity of the slave axis when it is called, and axis motion will continue at that final actual velocity.
- "MC\_Halt" (p. 444) (with buffer mode input = 0) will decelerate axis motion to 0 speed and stop motion.

The master / slave relationship between the two axes is ended when MC\_CamOut or MC\_Halt is called.

#### NOTE

Ending camming is also possible with other single axis function blocks such as "MC\_MoveRelative" (p. 458) and "MC\_MoveAbsolute" (p. 448).

#### 2.2.5.3.2 Arguments

##### 2.2.5.4.3.1 Input

<b>Execute</b>	<b>Description</b>	Requests to queue the CamIn move
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a

<b>Master</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	1 - 256
	<b>Unit</b>	n/a
<b>Slave</b>	<b>Description</b>	AXIS_REF.AXIS_NUM is the slave axis number
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	1-256
	<b>Unit</b>	n/a
<b>MasterOffset</b>	<b>Description</b>	Profile shift along the master axis. This input is not used if the StartMode input is set to 1 for Resume Mode.
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
<b>SlaveOffset</b>	<b>Description</b>	Profile shift along the slave axis. This input is not used if the StartMode input is set to 1 for Resume Mode.
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
<b>MasterScaling</b>	<b>Description</b>	Master axis profile range. This input is not used if the StartMode input is set to 1 for Resume Mode. Scaling must be a positive value that is greater than 0.
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
<b>SlaveScaling</b>	<b>Description</b>	Slave axis profile range. This input is not used if the StartMode input is set to 1 for Resume Mode. Scaling must be a positive value that is greater than 0.
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit



<b>Startmode</b>	<b>Description</b>	Starting mode of the cam profile. 0 = Start Mode. Start a cam profile move. 1 = Resume Mode. Resume the most recent MC_CamIn move.  This input indicates whether the axis should start a MC_CamIn move as an initial cam start (StartMode = 0) or if the axis should resume the most recently programmed MC_CamIn move (StartMode = 1).  It should be noted that in the case of Resume Mode (StartMode = 1) that the inputs MasterOffset, SlaveOffset, MasterScaling, and SlaveScaling are not used. The function block will use the values that were in effect during the most recently programmed MC_CamIn move for the slave axis.
	<b>Data type</b>	INT
	<b>Range</b>	[0,1]
	<b>Unit</b>	n/a
<b>CamTableID</b>	<b>Description</b>	ID number of the profile to be used with MC_CamIn
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
<b>BufferMode</b>	<b>Description</b>	Buffer mode for CamIn block.
	<b>Data type</b>	SINT
	<b>Range</b>	0-5
	<b>Unit</b>	n/a

#### 2.2.5.5.4.2 Output

<b>InSync</b>	<b>Description</b>	Indicates the slave axis is in sync with the profile
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
<b>Busy</b>	<b>Description</b>	Indicates this function block is executing
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
<b>Active</b>	<b>Description</b>	Indicates this move is the Active move

	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
<b>CommandAborted</b>	<b>Description</b>	Indicates the move was aborted
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
<b>Error</b>	<b>Description</b>	Indicates an invalid input, or the move was terminated due to an error
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
<b>ErrorID</b>	<b>Description</b>	Indicates the error if the Error output is high.
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
<b>EndOfProfile</b>	<b>Description</b>	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.
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a

#### 2.2.5.6.5 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

### 2.2.5.7.6 Related Functions

["MC\\_CamResumePos"](#) (p. 490)

["MC\\_CamStartPos"](#) (p. 493)

[MC\\_CamTblSelect](#)

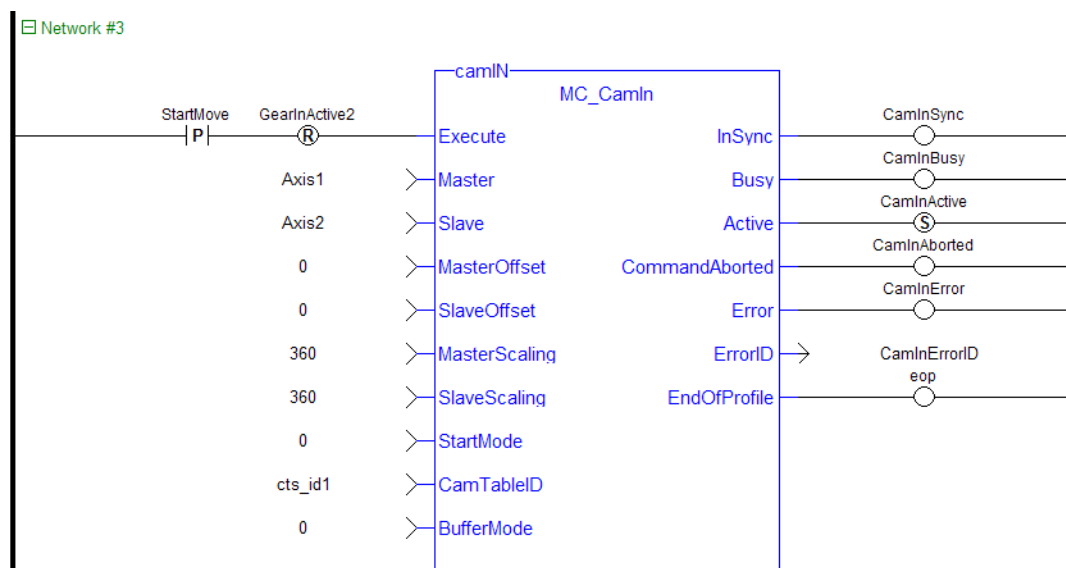
[MC\\_CamOut](#)

### 2.2.5.8.7 Examples

#### 2.2.5.9.8.1 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 );
```

#### 2.2.5.10.9.2 Ladder Diagram



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

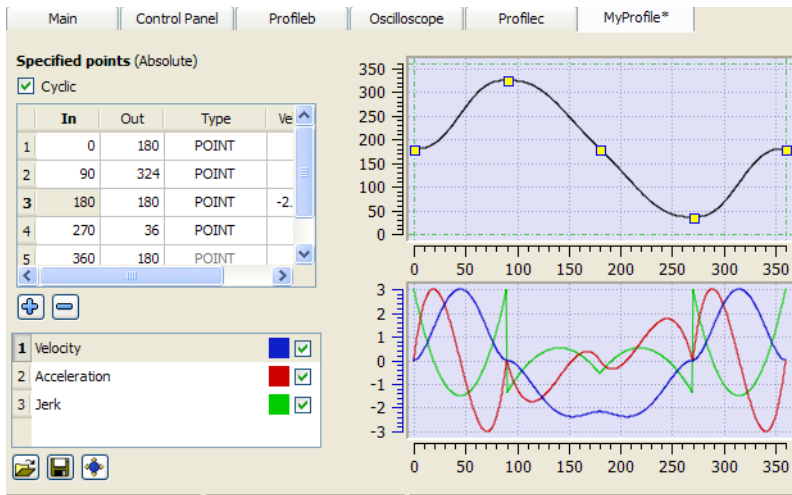
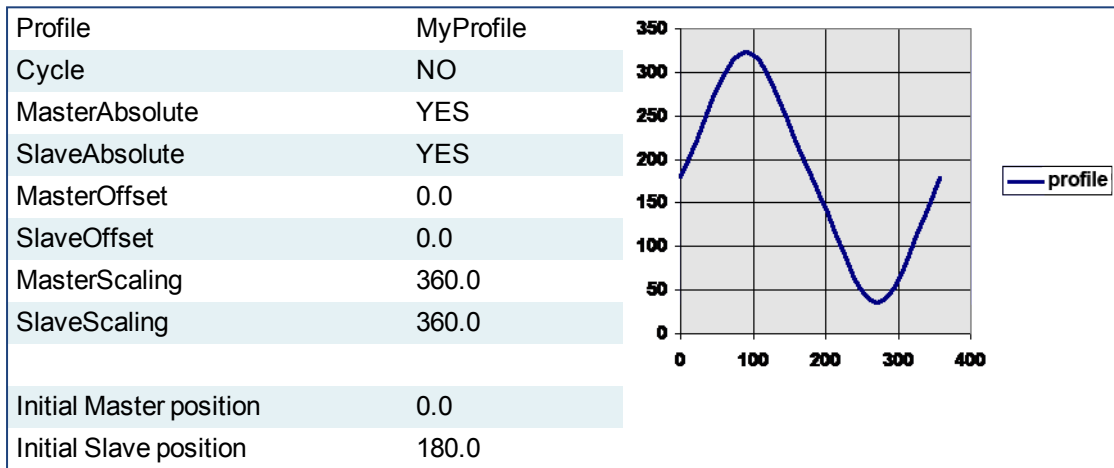


Figure 1-80: MC\_CamIn examples

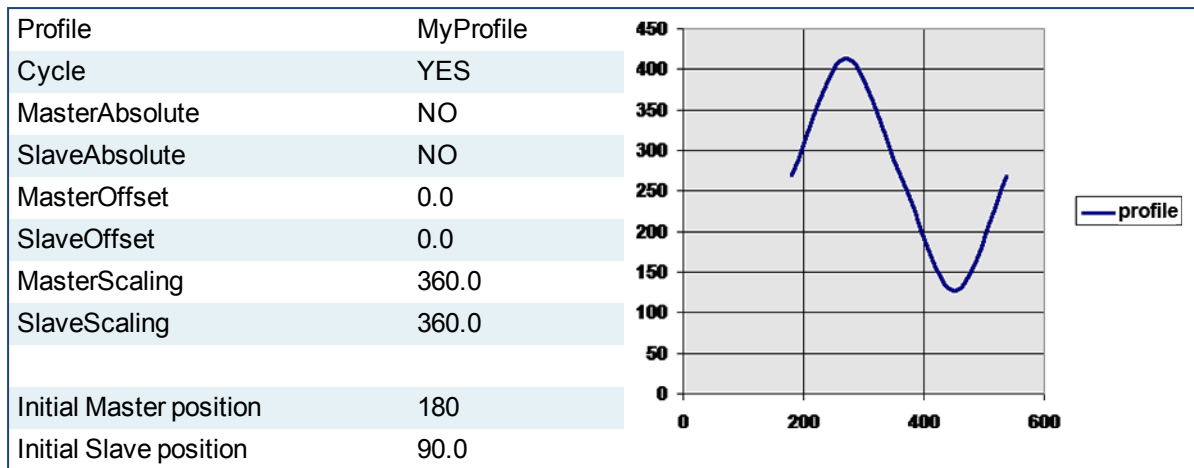
2.2.5.11.10.3 Example 1



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.

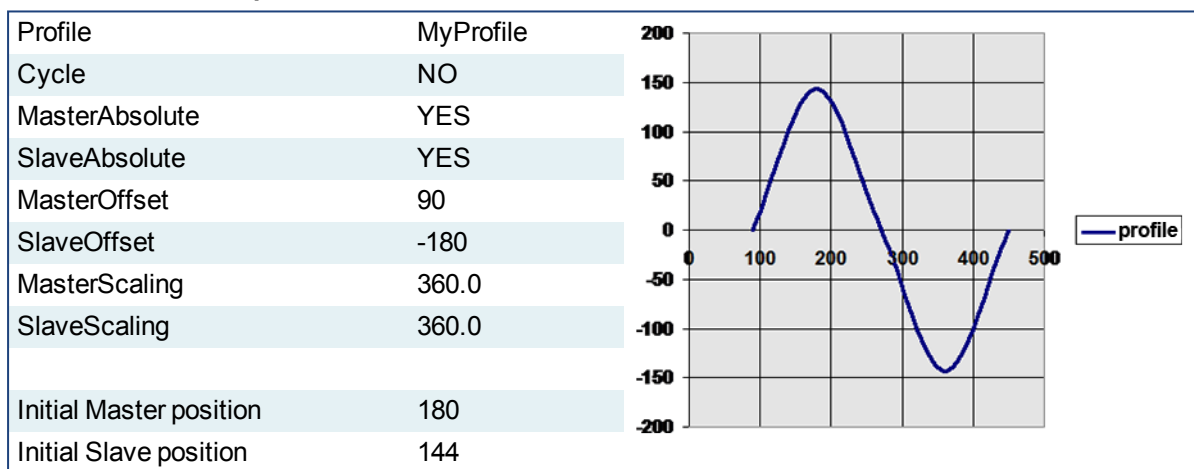
2.2.5.12.11.4 Example 2



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).

### 2.2.5.13.12.5 Example 3



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.

**2.2.5.14 MC\_CamOut** 

**2.2.5.15.1 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

**TIP**

The current velocity is calculated by taking the average of the actual velocity during the previous 16 cycles.

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.

**TIP**

As an alternative method to cancel the cam motion, a single axis move ("MC\_MoveAbsolute" (p. 448), "MC\_MoveRelative" (p. 458), "MC\_MoveAdditive" (p. 453), "MC\_MoveVelocity" (p. 468), and "MC\_Halt" (p. 444)) with the **buffermode** input set to **0** can be called. This will cancel the "MC\_CamIn" (p. 479) function and start the new motion function on the slave axis. Many applications prefer calling MC\_Halt instead of MC\_CamOut because it will not send a velocity command to the slave axis.

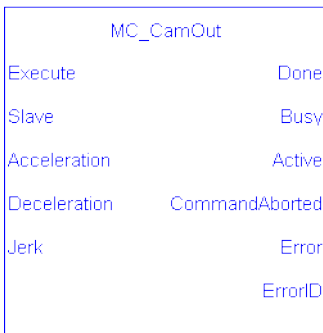


Figure 1-81: MC\_CamOut

**2.2.5.16.2 Arguments**

**2.2.5.17.3.1 Input**

<b>Execute</b>	<b>Description</b>	Requests to queue the CamOut move
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Slave</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	1 – 256
	<b>Unit</b>	n/a

	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Trapezoidal: Deceleration rate S-curve: Unused
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Trapezoidal: 0 S-curve: Constant jerk
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—

#### 2.2.5.18.4.2 Output

<b>Done</b>	<b>Description</b>	Indicates the axis is disengaged from its master
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
<b>Busy</b>	<b>Description</b>	Indicates this function block is executing
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
<b>Active</b>	<b>Description</b>	Indicates this move is the Active move
	<b>Data type</b>	BOOL

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

#### 2.2.5.19.5 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.

#### 2.2.5.20.6 Related Functions

[MC\\_CamIn](#)

[MC\\_CamTblSelect](#)

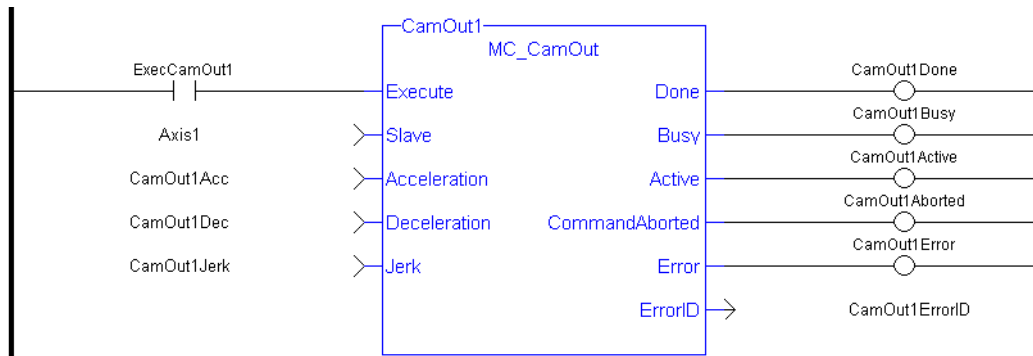
#### 2.2.5.21.7 Example

#### 2.2.5.22.8.1 Structured Text

```
(* MC_CamOut ST example *)
Inst_MC_CamOut
(ExecCamOut1, Axis1, CamOut1Acc, CamOut1Dec, CamOut1Jerk);
//Inst_MC_CamOut is an instance of MC_CamOut
```

#### 2.2.5.23.9.2 Ladder Diagram





See also [MC\\_CamIn](#) for examples.

**2.2.5.24 MC\_CamResumePos** PLCopen

**2.2.5.25.1 Description**

This function block returns the slave axis position for the most recently executed "MC\_CamIn" (p. 479) profile, based on the current position of the master axis. This slave axis position can be used to command the slave axis to return to the proper location prior to resuming a MC\_CamIn function. When calculating the slave axis position, MC\_CamResumePos will utilize the master offset, slave offset, master scaling, and slave scaling of the most recently executed MC\_CamIn function block for the slave axis.

The typical application of MC\_CamResumePos is to aid in returning a slave axis back to its profile position after an event (e.g. E-stop) caused the slave axis to go off path. See [Resuming Camming After an E-Stop](#) for complete instructions.

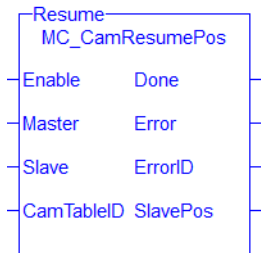


Figure 1-82: MC\_CamStartPos

**2.2.5.26.2 Related Functions**

["MC\\_CamIn" \(p. 479\)](#)

["MC\\_CamStartPos" \(p. 493\)](#)

**2.2.5.27.3 Arguments**

**2.2.5.28.4.1 Inputs**

<b>Enable</b>	<b>Description</b>	Enables execution of the function block
	<b>Data Type</b>	BOOL
	<b>Range</b>	n/a
	<b>Units</b>	n/a
	<b>Default</b>	—
<b>Master</b>	<b>Description</b>	Master axis. This must be the same as the Master Axis specified for the most recently executed MC_CamIn function block.
	<b>Data Type</b>	AXIS_REF
	<b>Range</b>	[1,256] for .AXIS_NUM
	<b>Units</b>	n/a
	<b>Default</b>	—
<b>Slave</b>	<b>Description</b>	Slave axis. This must be the same as the Slave Axis specified for the most recently executed MC_CamIn function block.
	<b>Data Type</b>	AXIS_REF

	<b>Range</b>	[1,256] for .AXIS_NUM
	<b>Units</b>	n/a
	<b>Default</b>	—
<b>CamTableID</b>	<b>Description</b>	Profile ID number. This value was generated by " <a href="#">MC_CamTblSelect</a> " (p. 497). This must be the same as the CamTableID specified for the most recently executed MC_CamIn function block.
	<b>Data Type</b>	INT
	<b>Range</b>	[0,255]
	<b>Units</b>	n/a
	<b>Default</b>	—

#### 2.2.5.29.5.2 Outputs

<b>Done</b>	<b>Description</b>	TRUE = the function block has successfully calculated the slave position. The slave position is available at the SlavePos output.
	<b>Data Type</b>	BOOL
	<b>Units</b>	n/a
<b>Error</b>	<b>Description</b>	TRUE = an invalid input was specified or an error occurred in the calculations. The value at the SlavePos output is undefined.
	<b>Data Type</b>	BOOL
	<b>Units</b>	n/a
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See table in <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data Type</b>	INT
	<b>Units</b>	n/a
<b>SlavePos</b>	<b>Description</b>	If the Done output is TRUE, this output returns the position for the slave axis given the profile, the current master axis position, and the previously programmed master and slave offsets and scaling.
	<b>Data Type</b>	LREAL
	<b>Units</b>	User Units

#### 2.2.5.30.6 Examples

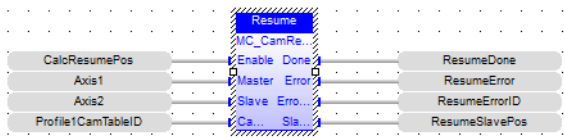
#### 2.2.5.31.7.1 ST

```
Inst_MC_CamStartPos( TRUE, Axis1, Axis2, Profile1CamTableID);
```

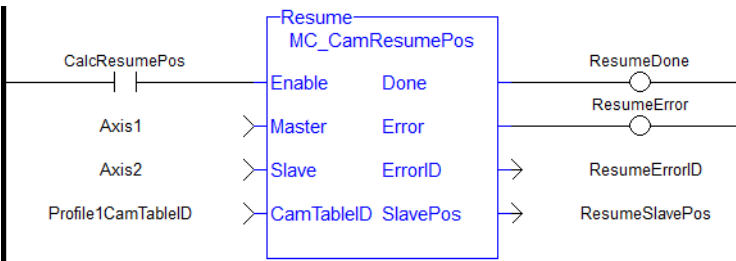
**2.2.5.32.8.2 IL**

```
CAL Inst_MC_CamResumePos( TRUE, Axis1, Axis2, Profile1CamTableID)
```

**2.2.5.33.9.3 FBD**



**2.2.5.34.10.4 FFLD**



### 2.2.5.35 MC\_CamStartPos

#### 2.2.5.36.1 Description

This function block returns the slave axis position for the specified profile, based on the current position of the master axis. This slave axis position can be used to command the slave axis to move to the proper location prior to commanding a "MC\_CamIn" (p. 479) move with StartMode = 0 (Start mode).

The typical application of MC\_CamStartPos is to aid in positioning a slave axis to its starting position for a MC\_CamIn move with a slave absolute profile. See [Positioning an Axis Before Starting Camming](#) for complete instructions.

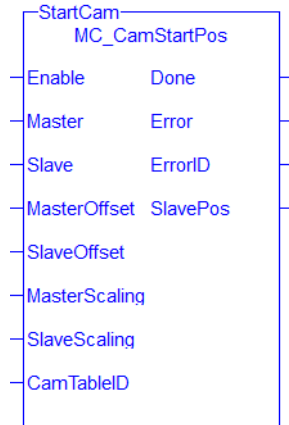


Figure 1-83: MC\_CamStartPos

#### 2.2.5.37.2 Arguments

##### 2.2.5.38.3.1 Inputs

<b>Enable</b>	<b>Description</b>	Enables execution of the function block
	<b>Data Type</b>	BOOL
	<b>Range</b>	n/a
	<b>Units</b>	n/a
	<b>Default</b>	—
<b>Master</b>	<b>Description</b>	Master axis
	<b>Data Type</b>	AXIS_REF
	<b>Range</b>	[1,256] for .AXIS_NUM
	<b>Units</b>	n/a
	<b>Default</b>	—
<b>Slave</b>	<b>Description</b>	Slave axis
	<b>Data Type</b>	AXIS_REF
	<b>Range</b>	[1,256] for .AXIS_NUM
	<b>Units</b>	n/a

	<b>Default</b>	—
<b>MasterOffset</b>	<b>Description</b>	Master axis offset
	<b>Data Type</b>	LREAL
	<b>Range</b>	—
	<b>Units</b>	User Units
	<b>Default</b>	—
<b>SlaveOffset</b>	<b>Description</b>	Slave axis offset
	<b>Data Type</b>	LREAL
	<b>Range</b>	—
	<b>Units</b>	User Units
	<b>Default</b>	—
<b>MasterScaling</b>	<b>Description</b>	Master axis scale factor. Scaling must be a positive value that is greater than 0.
	<b>Data Type</b>	LREAL
	<b>Range</b>	—
	<b>Units</b>	User Units
	<b>Default</b>	—
<b>SlaveScaling</b>	<b>Description</b>	Slave axis scale factor. Scaling must be a positive value that is greater than 0.
	<b>Data Type</b>	LREAL
	<b>Range</b>	—
	<b>Units</b>	User Units
	<b>Default</b>	—
<b>CamTableID</b>	<b>Description</b>	Profile ID number. This number was generated by " <a href="#">MC_CamTblSelect</a> " (p. 497).
	<b>Data Type</b>	INT
	<b>Range</b>	[0,255]
	<b>Units</b>	n/a
	<b>Default</b>	—

#### 2.2.5.39.4.2 Outputs

<b>Done</b>	<b>Description</b>	TRUE = the function block has successfully calculated the slave position. The slave position is available at the SlavePos output.
	<b>Data Type</b>	BOOL
	<b>Units</b>	n/a
<b>Error</b>	<b>Description</b>	TRUE = an invalid input was specified or an error occurred in the calculations. The value at the SlavePos output is undefined.
	<b>Data Type</b>	BOOL
	<b>Units</b>	n/a
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See table in <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data Type</b>	INT
	<b>Units</b>	n/a
<b>SlavePos</b>	<b>Description</b>	If the Done output is TRUE, this output returns the position for the slave axis given the profile and the current master axis position.
	<b>Data Type</b>	LREAL
	<b>Units</b>	User Units

### 2.2.5.40.5 Examples

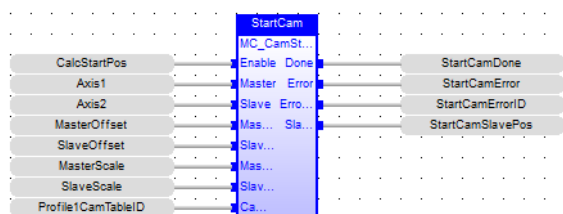
#### 2.2.5.41.6.1 ST

```
Inst_MC_CamStartPos( TRUE, Axis1, Axis2, MasterOffset,
SlaveOffset, MasterScale, SlaveScale, Profile1CamTableID);
```

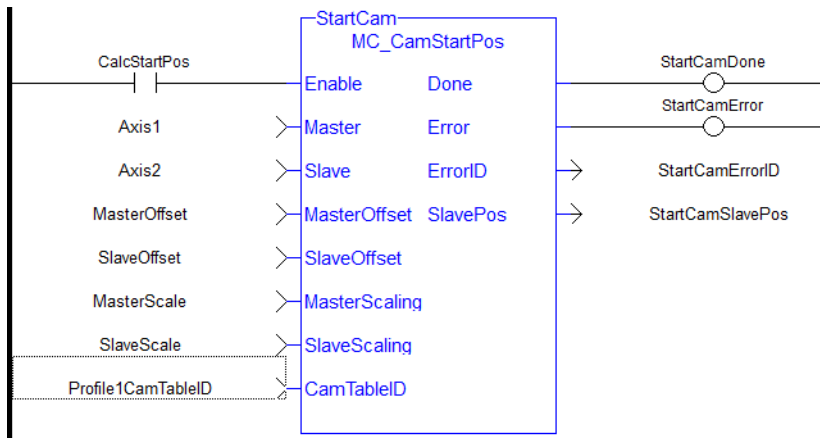
#### 2.2.5.42.7.2 IL

```
CAL Inst_MC_CamStartPos( TRUE, Axis1, Axis2, MasterOffset,
SlaveOffset, MasterScale, SlaveScale, Profile1CamTableID)
```

#### 2.2.5.43.8.3 FBD



#### 2.2.5.44.9.4 FFLD





### 2.2.5.45 MC\_CamTblSelect

#### 2.2.5.46.1 Description

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

#### 2.2.5.47.2 Arguments

##### 2.2.5.48.3.1 Input

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

<b>Range</b>	0, 1
<b>Unit</b>	n/a
<b>Default</b>	—

#### 2.2.5.49.4.2 Output

<b>Done</b>	<b>Description</b>	Indicates the function block has completed successfully
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
<b>Busy</b>	<b>Description</b>	Indicates this function block is executing
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
<b>ErrorID</b>	<b>Description</b>	Indicates the error if the Error output is high
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
<b>CamTableID</b>	<b>Description</b>	Indicates the ID number of the profile to be used with MC_CamIn
	<b>Data type</b>	INT
	<b>Range</b>	0 - 255
	<b>Unit</b>	n/a

#### 2.2.5.50.5 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.

**2.2.5.51.6 Related Functions**

[MC\\_CamIn](#)

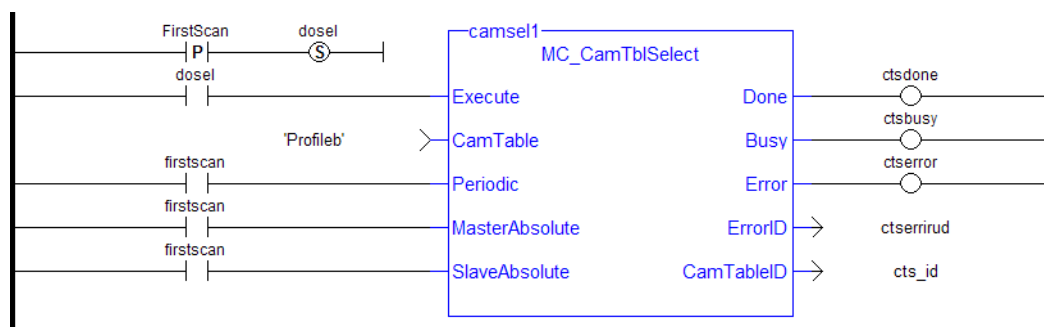
[MC\\_CamOut](#)

**2.2.5.52.7 Example****2.2.5.53.8.1 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](#)

**2.2.5.54.9.2 Ladder Diagram**

See also [MC\\_CamIn](#) for examples.

**2.2.5.55 MC\_GearIn** PLCopen **2.2.5.56.1 Description**

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

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

When this command is executed, the slave axis accelerates or decelerates (using the Acceleration, Deceleration, and Jerk) to the target velocity determined by the master axis velocity and the ratio. When the slave axis reaches a velocity within the ["In Gear" bandwidth](#) around the target velocity, it locks on to the master, and the InGear output goes high. When the slave is locked to the master, the slave motion is no longer affected by the acceleration, deceleration, and jerk inputs.

For example if the "In Gear" bandwidth is set to 0.1 User Units per second, the InGear output will turn on if the slave velocity is within +/- 0.1 User Units per second of the target velocity.

The slave axis then continues to follow the master axis until this move is aborted.

**Aborting Gearing**

Gearing functions can generate large accelerations while following the master. If the aborting function block has small, non-zero Jerk, or small acceleration values, it can take a long time for an accelerating axis to reach the target velocity, or position of the aborting function block. If the Jerk and/or acceleration of the aborting function cannot be increased to suitable values, it may be desirable to:

- Abort the gearing function with an ["MC\\_GearOut"](#) (p. 512) with higher accelerations and/or Jerk values (or zero jerk value),
- Execute the next MC motion function block such as ["MC\\_Halt"](#) (p. 444).

**Time to Reach the Target Velocity**

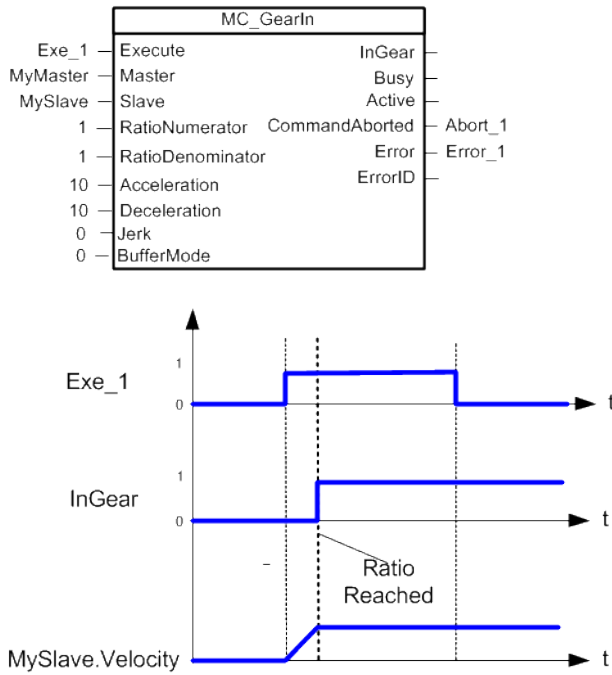
While following the master, gearing functions can generate large accelerations. If the gearing function is aborted while the axis is currently accelerating, and the aborting function block has small non-zero Jerk or small acceleration values, it can take a long time to reach the target velocity, or position 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 such as ["MC\\_Halt"](#) (p. 444).

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

**Figure 1-84:** MC\_GearIn

**2.2.5.57.2 Time Diagram**



**2.2.5.58.3 Arguments**

**2.2.5.59.4.1 Input**

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

	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483647]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>RatioDenominator</b>	<b>Description</b>	Denominator of master/slave ratio
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483647]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Trapezoidal: Deceleration rate S-curve: Unused
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Trapezoidal: 0 S-curve: Constant jerk
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—
<b>BufferMode</b>	<b>Description</b>	0 = abort 1 = buffer
	<b>Data type</b>	SINT
	<b>Range</b>	[0,1]

<b>Unit</b>	n/a
<b>Default</b>	—

#### 2.2.5.60.5.2 Output

<b>InGear</b>	<b>Description</b>	Indicates the slave axis is locked on to the master axis
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	High from the moment the Execute input goes high until the time the move is ended
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	Indicates this move is the Active move
	<b>Data type</b>	BOOL
<b>CommandAborted</b>	<b>Description</b>	Indicates the move was aborted
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified or the move was terminated due to an error
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE
	<b>Data type</b>	INT

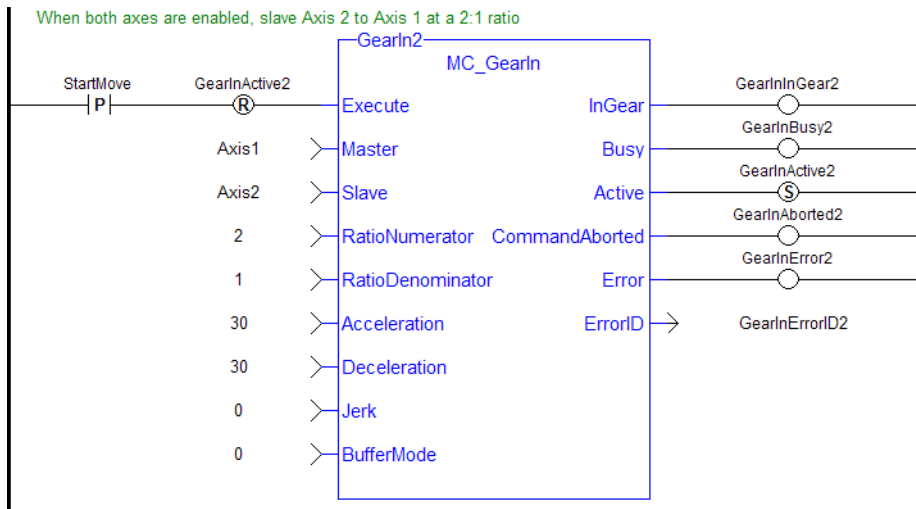
#### 2.2.5.61.6 Example

##### 2.2.5.62.7.1 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](#)

#### 2.2.5.63.8.2 Ladder Diagram





**2.2.5.64 MC\_GearInPos** PLCopen **2.2.5.65.1 Description**

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

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

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.

**Aborting Gearing**

Gearing functions can generate large accelerations while following the master. If the aborting function block has small, non-zero Jerk, or small acceleration values, it can take a long time for an accelerating axis to reach the target velocity, or position of the aborting function block. If the Jerk and/or acceleration of the aborting function cannot be increased to suitable values, it may be desirable to:

- Abort the gearing function with an ["MC\\_GearOut"](#) (p. 512) with higher accelerations and/or Jerk values (or zero jerk value),
- Execute the next MC motion function block such as ["MC\\_Halt"](#) (p. 444).

**Time to Reach the Target Velocity**

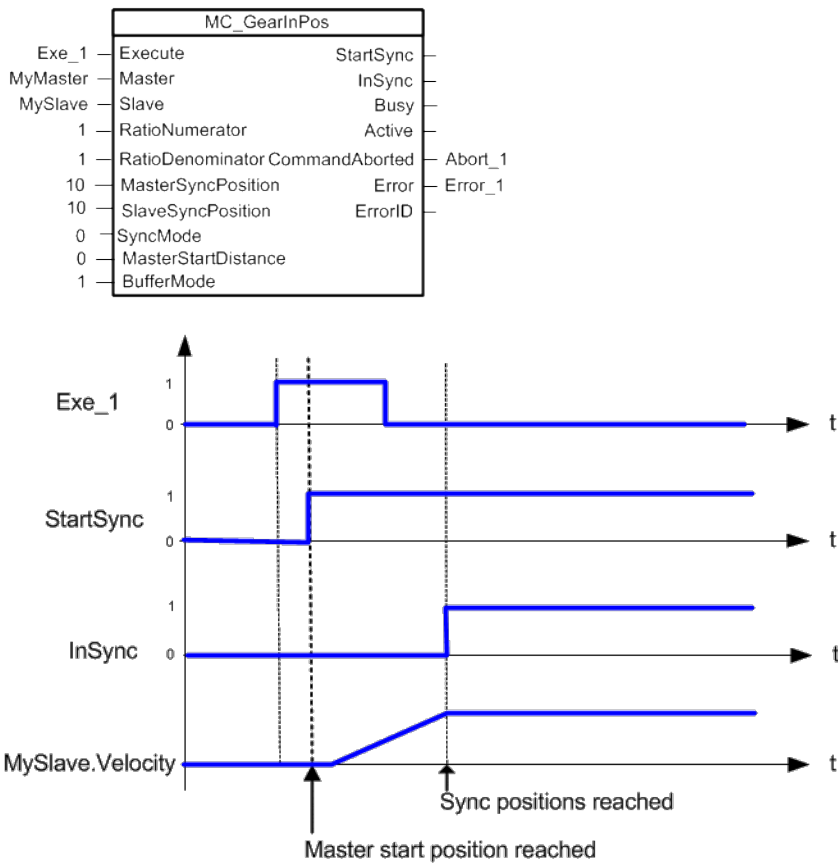
While following the master, gearing functions can generate large accelerations. If the gearing function is aborted while the axis is currently accelerating, and the aborting function block has small non-zero Jerk or small acceleration values, it can take a long time to reach the target velocity, or position 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 such as ["MC\\_Halt"](#) (p. 444).

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

**Figure 1-85:** MC\_GearInPos

**2.2.5.66.2 Time Diagram**



### 2.2.5.67.3 Arguments

#### 2.2.5.68.4.1 Input

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

	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>RatioNumerator</b>	<b>Description</b>	Numerator of master/slave ratio. The sign (+ or -) indicates the direction for the slave axis.
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483647]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>RatioDenominator</b>	<b>Description</b>	Denominator of master/slave ratio. The sign (+ or -) indicates the direction for the master axis. The direction determines the sync trigger comparison direction for the slave. <ul style="list-style-type: none"> <li>• For a master moving in the positive direction, use a positive RatioDenominator.</li> <li>• For a master moving in the negative direction, use a negative RatioDenominator.</li> </ul>
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483647]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>MasterSyncPosition</b>	<b>Description</b>	Master axis sync position
	<b>Data type</b>	LREAL
	<b>Range</b>	$-1.7^{308}$ to $1.7^{308}$ (14 to 15 significant digits of accuracy)
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>SlaveSyncPosition</b>	<b>Description</b>	Slave axis sync position
	<b>Data type</b>	LREAL
	<b>Range</b>	$-1.7^{308}$ to $1.7^{308}$ (14 to 15 significant digits of accuracy)
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>SyncMode</b>	<p><b>Description</b></p> <p>SyncMode determines the allowed conditions for synchronization:</p> <p><b>0 = Normal synchronization.</b> Prior to executing the MC_GearInPos function block, the <i>Master</i> axis position must be before the <i>MasterSyncPosition</i> by a distance greater than the <i>MasterStartDistance</i>. The <i>Slave</i> axis position must be before the <i>SlaveSyncPosition</i>.</p> <p>In the case of axes that have a non-zero rollover, the MC_GearInPos function block will always assume the axes meet these conditions by assuming the sync point is in the next occurrence of the sync position. <i>MasterStartDistance</i> must be positive and greater than the distance the master axis is currently moving per axis update. If the master start distance and the slave axis distance from the <i>SlaveSyncPosition</i> are sufficiently large enough, the slave axis will ramp to the sync position. If not sufficiently large enough, acceleration of the slave axis may be excessive.</p> <p><b>1 = Immediate synchronization allowed.</b> This mode is only allowed if both the master and slave axes have rollover = 0. If the conditions of SyncMode = 0 are not met, Synchronization is allowed even though the axis positions may be beyond their respective Sync Positions. The <i>MasterStartDistance</i> may be 0. If the <i>MasterStartDistance</i> is zero, the Slave axis will synchronize with the master the instant the master axis crosses the <i>MasterSyncPosition</i>.</p> <p>If either the master or slave axis are beyond their respective sync start positions, the slave axis will immediately synchronize to the master axis. If the master start distance and the slave axis distance from the <i>SlaveSyncPosition</i> are sufficiently large enough, the slave axis will ramp to the sync position. If not sufficiently large enough or immediate synchronization occurs, slave axis acceleration may be excessive.</p> <p>Excessive slave acceleration may also occur if the master axis velocity is large or the master and slave axes have disproportionately different distances to their respective sync positions. If the slave axis is ahead of the master axis at the time of synchronization, the slave axis will move backwards.</p>
	<p><b>Data type</b> INT</p>
	<p><b>Range</b> 0-1</p>
	<p><b>Unit</b> n/a</p>
	<p><b>Default</b> —</p>

<b>MasterStartDistance</b>	<b>Description</b>	When the master axis reaches this distance before MasterSyncPosition, the slave axis begins its lock-on process.
		<b>ⓘ IMPORTANT</b>
		The <b>MasterStartDistance</b> * ( <b>RatioNumerator/RatioDenominator</b> ) should be greater than (or equal to) the slave sync distance. The slave sync distance is defined as the distance between the slave position when MC_GearInPos executes and the <b>SlaveSyncPosition</b> . If the MasterStartDistance is too short, the MC_GearInPos may have excessive acceleration and a warning log message will be generated.
	<b>Data type</b>	LREAL
	<b>Range</b>	1.7 <sup>-308</sup> to 1.7 <sup>308</sup> (14 to 15 significant digits of accuracy)
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>BufferMode</b>	<b>Description</b>	1 = buffer
	<b>Data type</b>	SINT
	<b>Range</b>	[1]
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 2.2.5.69.5.2 Output

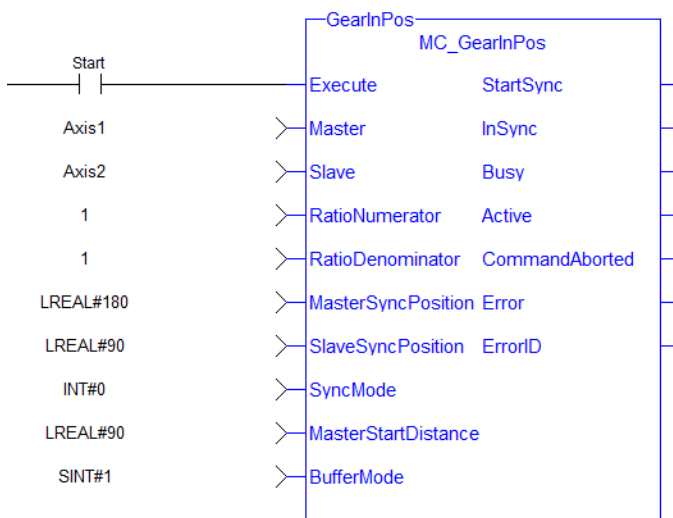
<b>StartSync</b>	<b>Description</b>	Indicates that the master axis has reached the MasterStartDistance from the MasterSyncPosition and the lock-on process has begun
	<b>Data type</b>	BOOL
<b>InSync</b>	<b>Description</b>	Indicated the slave axis is locked on to the master axis
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	High from the moment the Execute input goes high until the time the move is ended
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	Indicates this move is the Active move
	<b>Data type</b>	BOOL

<b>CommandAborted</b>	<b>Description</b>	Indicates the move was aborted. If the abort arises because the inputs cause inconsistent motion, then this FB: <ul style="list-style-type: none"> <li>• performs no motion</li> <li>• sets an error flag</li> <li>• set the ErrorID to <a href="#">13</a></li> </ul>
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified or the move was terminated due to an error
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE
	<b>Data type</b>	INT

### 2.2.5.70.6 Example

#### 2.2.5.71.7.1 Example Description

- Master and Slave are rotary axes with rollovers at 360 degrees.
- The Master initial position is 0 degrees and the slave initial position is 45 degrees.
- The GearInPos FB commands the slave to accelerate up to the geared ratio (1:1) during the master start distance (90 degrees) and be synchronized with the master at the master and slave sync positions.

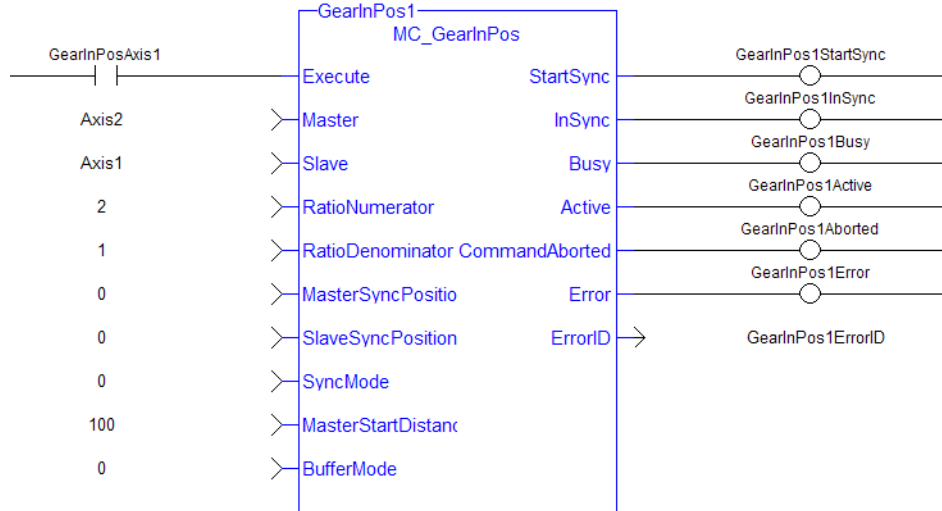


1. Master Axis
2. Slave Axis
3. Master Start Distance
4. Slave starts to accelerate
5. Master Sync Position
6. Slave Sync Position

#### 2.2.5.72.8.2 Structured Text

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

### 2.2.5.73.9.3 Ladder Diagram



**2.2.5.74 MC\_GearOut** PLCopen

**2.2.5.75.1 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.

**TIP**

The current velocity is calculated by taking the average of the actual velocity during the previous 16 cycles.

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.

**NOTE**

The MC\_GearOut is done when the slave axis is disengaged from the master axis. Once done, the MC\_GearOut will remain busy and active until it is aborted by a different motion function block. This is different behavior than most other motion function blocks. The MC\_GearOut function block represents an exception to the exclusivity rule as the **Done** and **Active** outputs may be true at the same time.

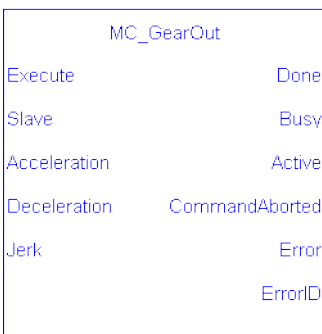


Figure 1-86: MC\_GearOut

**2.2.5.76.2 Arguments**

**2.2.5.77.3.1 Input**

<b>Execute</b>	<b>Description</b>	Requests to disengage the slave axis from a MC_GearIn or MC_GearInPos move
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Slave</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]



	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Trapezoidal: Deceleration rate S-curve: Unused
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Trapezoidal: 0 S-curve: Constant jerk
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—

#### 2.2.5.78.4.2 Output

<b>Done</b>	<b>Description</b>	Indicates the axis is disengaged from its master
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	Indicates the function is executing
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	Indicates this move is the Active move
	<b>Data type</b>	BOOL
<b>CommandAborted</b>	<b>Description</b>	Indicates the move was aborted
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified or no MC_GearIn or MC_GearInPos move is active

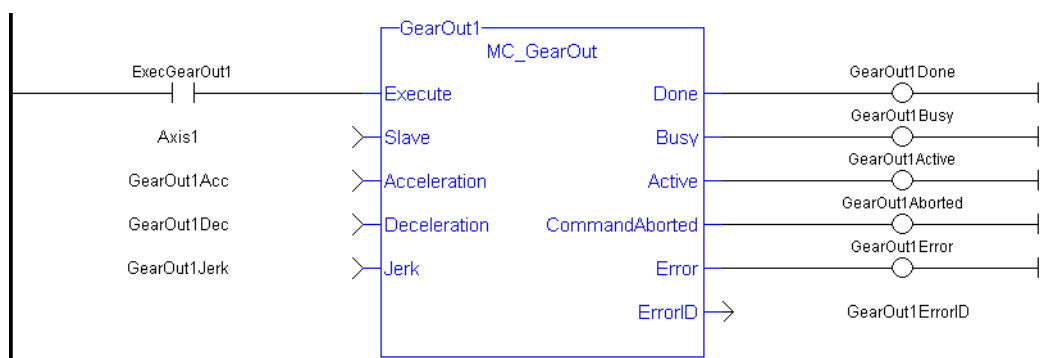
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE
	<b>Data type</b>	INT

**2.2.5.79.5 Example**

**2.2.5.80.6.1 Structured Text**

```
(* MC_GearOut ST example *)
Inst_MC_GearOut
(ExecGearOut1,Axis1,GearOut1Acc,GearOut1Dec,GearOut1Jerk);
//Inst_MC_GearOut is instance of MC_GearOut
```

**2.2.5.81.7.2 Ladder Diagram**



## 2.2.5.82 MC\_Phasing PLCopen

### 2.2.5.83.1 Description

The MC\_Phasing function block performs a master position phase shift for a slave axis. The distance entered at the **PhaseShift** input is iterated into the Slave axis's Master Offset. This distance is iterated like a "[MC\\_MoveRelative](#)" (p. 458) move using the specified **Velocity**, **Acceleration**, **Deceleration**, and **Jerk** values. The difference is that the interpolated command delta is not commanded to the axis but is, instead, added to the Slave axis's Master Offset. This will shift the Master axis's position as viewed by the Slave axis, causing a change in the Slave axis's physical position. This will only affect the Slave axis if it is executing a slave move. Subsequent calls to MC\_Phasing can abort or blend to an executing MC\_Phasing command.

This function block provides a way to smoothly apply a master offset instead of writing values directly to the Master Offset Parameter 1002.

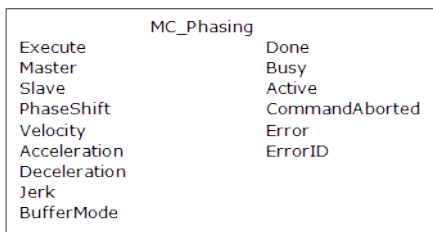


Figure 1-87: MC\_Phasing

### 2.2.5.84.2 Arguments

#### 2.2.5.85.3.1 Input

<b>Execute</b>	<b>Description</b>	Requests to queue the phase shift
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Master</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function. )
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Slave</b>	<b>Description</b>	AXIS_REF AXIS_NUM is the slave axis number
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>PhaseShift</b>	<b>Description</b>	Amount of phase shift

	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>Velocity</b>	<b>Description</b>	Velocity setpoint
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Trapezoidal: Deceleration rate S-curve: Unused
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Trapezoidal: 0 S-curve: Constant jerk
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—
<b>BufferMode</b>	<b>Description</b>	<ul style="list-style-type: none"> <li>0. abort</li> <li>1. buffer</li> <li>2. blend to active</li> <li>3. blend to next</li> <li>4. blend to low velocity</li> <li>5. blend to high velocity</li> </ul>
	<b>Data type</b>	SINT
	<b>Range</b>	[0,5]

<b>Unit</b>	n/a
<b>Default</b>	—

#### 2.2.5.86.4.2 Output

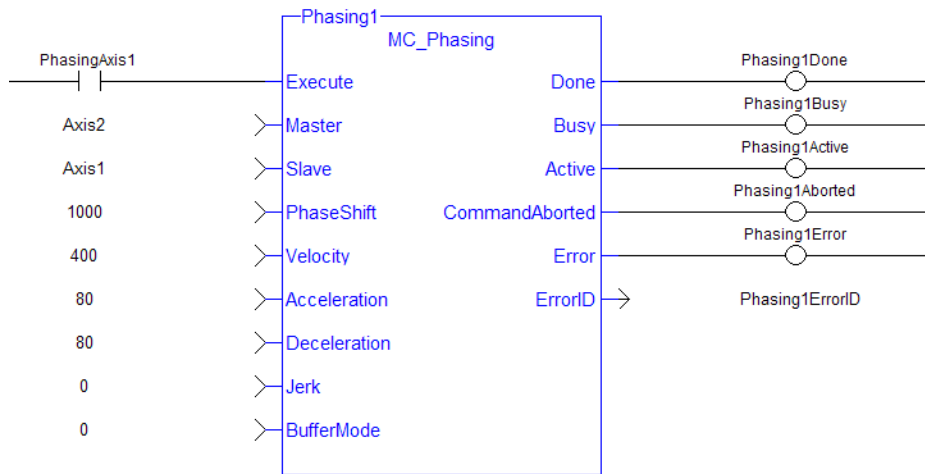
<b>Done</b>	<b>Description</b>	Indicates the phase shift has been completely applied
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	High from the moment the Execute input is one-shot to the time the move is ended
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	Indicates this phase shift is the active phase shift
	<b>Data type</b>	BOOL
<b>CommandAborted</b>	<b>Description</b>	Indicates the move was aborted
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified or the move was terminated due to an error
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE
	<b>Data type</b>	INT

#### 2.2.5.87.5 Example

##### 2.2.5.88.6.1 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 );
```

#### 2.2.5.89.7.2 Ladder Diagram



## 2.2.5.90 MC\_SyncSlaves

### 2.2.5.91.1 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.

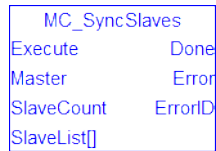


Figure 1-88: MC\_SyncSlaves

### 2.2.5.92.2 Arguments

#### 2.2.5.93.3.1 Input

<b>Execute</b>	<b>Description</b>	A positive transition of this input causes the function block to execute
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Master</b>	<b>Description</b>	Master axis identifier
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	1 - 256
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>SlaveCount</b>	<b>Description</b>	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.
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	1-256 The AXIS_NUM element of the AXIS_REF structure must be in the range [1-256]
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>SlaveList</b>	<b>Description</b>	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.
	<b>Data type</b>	UINT
	<b>Range</b>	1-32
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 2.2.5.94.4.2 Output

<b>Done</b>	<b>Description</b>	Indicates the synchronized slave assignments were completed without error
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE
	<b>Data type</b>	INT

#### 2.2.5.95.5 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.

#### 2.2.5.96.6 Related Functions

[MC\\_GearIn](#)

[MC\\_GearInPos](#)

[MC\\_CamIn](#)

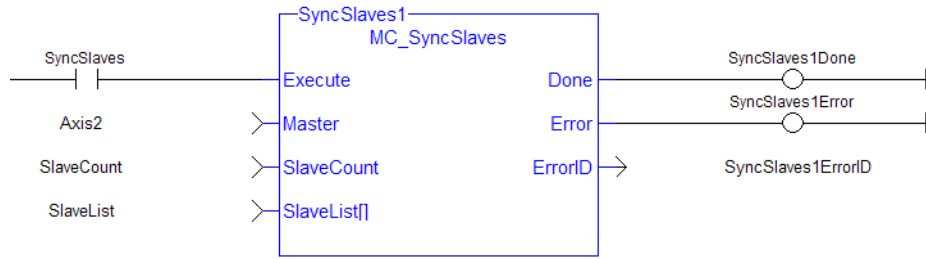
#### 2.2.5.97.7 Example

##### 2.2.5.98.8.1 Structured Text

```
(* MC_SyncSlaves ST example *)
// Inst_MC_SyncSlaves is an instance of MC_SyncSlaves function
block
Inst_MC_SyncSlaves( SyncSlaves, Axis1, SlaveCount, SlaveList );
```



**2.2.5.99.9.2 Ladder Diagram**



## 2.2.6 Reference Functions

This set of functions provides commands for reference points.

### 2.2.6.1 MC\_Reference PLCopen ✓

#### 2.2.6.2.1 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.

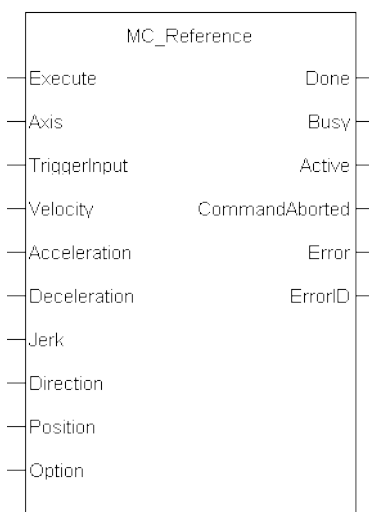


Figure 1-89: MC\_Reference

#### 2.2.6.3.2 Arguments

##### 2.2.6.4.3.1 Input

<b>Execute</b>	<b>Description</b>	Requests to queue the MC_Reference move and arms reference trigger events
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function. )
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a

	<b>Default</b>	—
<b>TriggerInput</b>	<b>Description</b>	<p>TRIGGER_REF structure defines the trigger</p> <p><b>InputID</b> INT :</p> <p>0 = Capture Engine 0 1 = Capture Engine 1 Range is [0, 1] For information on configuring the capture engines, refer to <a href="#">AKD Capture Engine Configuration</a>.</p> <p><b>Direction</b> INT; 1 = rising edge of trigger, 2 = falling edge of trigger</p> <p><b>Trigid</b> INT; must be zero</p> <p><b>NOTE</b></p> <p><b>TrigMode</b> INT (TriggerInput.TrigMode) is not presently supported by this function. The TriggerInput.Mode may be supported in a future software version.</p>
	<b>Data type</b>	TRIGGER_REF
	<b>Range</b>	See Description above
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Velocity</b>	<b>Description</b>	Commanded velocity for the reference move
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Commanded acceleration for the reference move
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Commanded deceleration for the reference move
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—

<b>Jerk</b>	<b>Description</b>	Commanded jerk for the reference move (if zero, then trapezoidal acc/dec is used)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—
<b>Direction</b>	<b>Description</b>	Commanded Direction of the reference
	<b>Data type</b>	SINT
	<b>Range</b>	[0, 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Position</b>	<b>Description</b>	The position the axis will be reset to when at the machine reference location
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>Option</b>	<b>Description</b>	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
	<b>Data type</b>	SINT
	<b>Range</b>	[0, 15]
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 2.2.6.5.4.2 Output

<b>Done</b>	<b>Description</b>	Indicates the reference move and position adjustment is complete
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	Indicates this function block is executing
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	Indicates this move is the Active move
	<b>Data type</b>	BOOL
<b>CommandAborted</b>	<b>Description</b>	Indicates the move was aborted
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input, or the move was terminated due to an error
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if the Error output is high
	<b>Data type</b>	INT

#### 2.2.6.6.5 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 [AKD Capture Engine Configuration](#). Example AKD capture engine configurations and reference examples are discussed in [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.

### 2.2.6.7.6 Related Functions

[ECATWriteSdo](#)

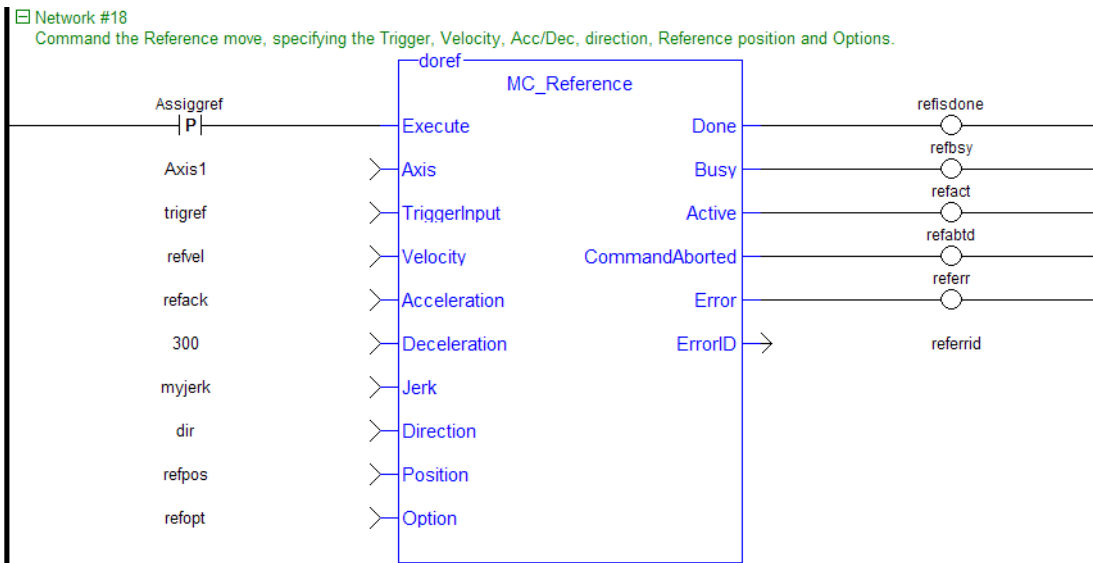
[MC\\_MoveAbsolute](#)

### 2.2.6.8.7 Example

#### 2.2.6.9.8.1 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 );
```

### 2.2.6.10.9.2 Ladder Diagram



### 2.2.6.11 MC\_SetPos

#### 2.2.6.12.1 Description

This function block changes the present actual position of the axis (as reported by "MC\_ReadActPos" (p. 427)) to the position specified by the **Position** and **Mode** inputs. If a motor is associated with the axis, it will not move when MC\_SetPos is executed. MC\_SetPos does not cause any motion. It applies an offset to the command and actual positions.

MC\_SetPos also sets the accumulated Superimposed distance value for the input axis to 0. See the table in [Axis Positions Data](#).

This function block replaces the MC\_SetPosition function.

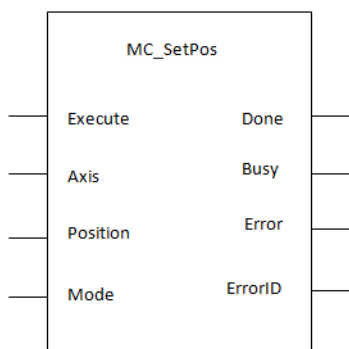


Figure 1-90: MC\_SetPos

#### 2.2.6.13.2 Arguments

For more detail on how inputs and outputs work, refer to PLCopen Function Blocks - General Rules.

##### 2.2.6.14.3.1 Inputs

<b>Execute</b>	<b>Description</b>	Requests to change the axis position
	<b>Data type</b>	BOOL
	<b>Range</b>	0,1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function. For more details <a href="#">Modify PLCopen Axis</a> .
	<b>Data type</b>	<a href="#">AXIS_REF Structure</a>
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Position</b>	<b>Description</b>	<b>Absolute Mode:</b> New Axis Position to replace the present position. <b>Relative Mode:</b> Position offset to apply to present position (typically used with multiturn absolute position feedback devices).

	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Mode</b>	<b>Description</b>	LOW = Position input is an absolute position HIGH = Position input is a relative position
	<b>Data type</b>	BOOL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 2.2.6.15.4.2 Outputs

<b>Done</b>	<b>Description</b>	Indicates the reference move and position adjustment is complete
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	Indicates this function block is executing
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input, or the move was terminated due to an error
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if the Error output is high See table in <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data type</b>	INT

#### 2.2.6.16.5 Example

##### 2.2.6.17.6.1 Structured Text

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

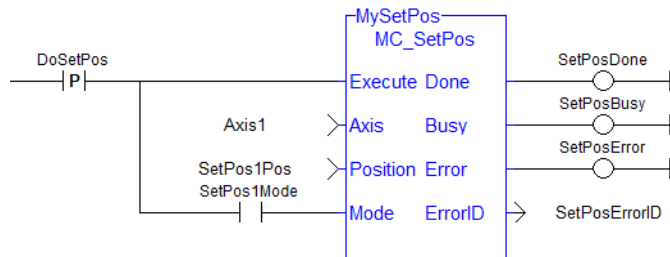
(* MC_SetPos absolute mode example: Set position value to zero.
*)
Inst_MC_SetPos ( Axis1 , 0, 0 );
//Inst_MC_SetPos is an instance of MC_SetPos function

(* MC_SetPos relative mode example: Increase position value by
1000. *)
```



```
Inst_MC_SetPos ( Axis1 , 1000 , 1 );
Inst_MC_SetPos is an instance of MC_SetPos function
```

### 2.2.6.18.7.2 Ladder Diagram



### 2.2.6.19 MC\_SetPosition

#### 2.2.6.20.1 Description

This function has been deprecated by the "MC\_SetPos" (p. 527) function block.

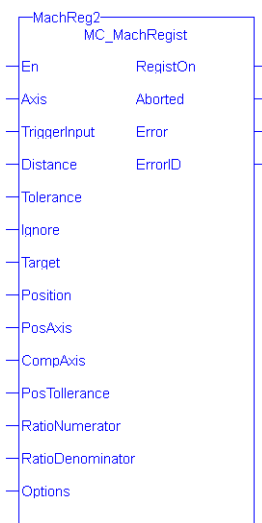
### 2.2.7 Registration Function Blocks

This set of function blocks allow for Mark-to-Mark or Mark-to-Machine registration. See [Registration](#) for techniques on setting up and using the registration function blocks.

#### 2.2.7.1 MC\_MachRegist PLCopen

##### 2.2.7.2 Description

This function block enables Mark-to-Machine registration and can be used on any servo or digitizing axis and with any move type. It is most frequently used in master/slave applications.



Used with	Effect
...	
Non-slave moves	Resets the axis position when a good mark is captured by the fast input.
Slave moves	In addition to resetting the axis position, applies a compensation offset to correct for the difference between the target position and the measured position. This provides the ability to compensate for product or process inconsistencies providing a system that remains synchronized with no accumulated error and maintaining repeatable accuracy throughout the process.

- A positive transition of the **En** input will start registration. The application may change the registration parameters while registration is active by changing the input values and causing another positive transition of the En input. The function block will then read and apply the new values.
- The axis number at the **Axis** input indicates the axis whose position, at the fast input, is used to determine if the mark is a good mark.
- The **Distance**, **Tolerance**, and **Ignore** inputs are used to determine whether or not the registration mark is good. For a mark to be recognized as good, it must be outside of the Ignore distance and the correct Distance from the previous mark +/- the Tolerance window. A mark is considered bad if it occurs outside of the "good tolerance band" and is not ignored. Both good marks and bad marks are recognized as marks, ignored marks are not recognized. If all marks are to be recognized as good marks, enter 0 at both **Distance** and **Tolerance**.

- The **Distance** value defines the distance between good marks. In Clear Lane and Product registration the Distance input value typically is the same as the Target input value. However in Print registration the Distance is typically not the same as Target.
- The **Tolerance** value is the distance, plus and minus, about Distance. Marks that are detected within this window are considered good marks and registration will occur. Marks that are detected outside this window and outside the Ignore band, are considered bad marks and registration will not occur. This window should be large enough to allow for the worst case error in the distance between the previous mark and the current mark.
- The **Ignore** value defines the distance from the previous mark where all marks detected by the fast input will be ignored. This is crucial when registering products that do not have Clear Lane registration marks.
- The **Target** input is the expected target position that is used to calculate how much registration compensation is to be applied when a registration mark is considered good. When a good mark is detected, the position of the **PosAxis** is compared to the Target position to calculate a correction. The registration correction will only be applied with master/slave move types.
- The **Position** input is the position value that the registration Axis position will be reset to when a good registration mark is detected.
- When a good mark occurs the position of the **PosAxis** is compared to the Target position and used to calculate the amount of registration compensation to apply to the CompAxis.
- Registration compensation is applied to the axis specified at the **CompAxis** input under the following conditions. If CompAxis is executing a slave move (i.e. MC\_GearIn or MC\_CamIn), the compensation is applied directly to the axis. If CompAxis is a master axis, the compensation is applied to the master offsets of all its slaves. This shifts the master's position as seen by its slaves.
- The **PosTolerance** input is the distance, plus and minus, about the Target position used to determine if compensation will be applied. When a good mark occurs, the position of the **PosAxis** axis is checked to see if it lies within the window defined by PosTolerance. If it is in the window, compensation will be applied. If it is outside the window, compensation will not be applied even though a good mark was found.
- If **PosAxis** and **CompAxis** are different axes, the **RatioNumerator** and **RatioDenominator** inputs define the conversion factor for calculating the compensation value. This is needed because the amount of error between actual and target positions is determined by PosAxis's position and the compensation is applied to the CompAxis. The RatioNumerator should typically be the number of User Units of CompAxis motion for one registration cycle and the RatioDenominator should typically be the number of User Units of PosAxis motion for one registration cycle. If PosAxis and CompAxis are the same, RatioNumerator and RatioDenominator should be the same value, thus resulting in a 1:1 ratio.
- The **Option** input defines various modes of operation for registration.
  - The first bit, 0001H, selects Absolute or Resetting. This refers to the way in which the second mark and all subsequent marks are determined to be good marks. With both registration schemes, the very first mark detected is the starting point. With Resetting registration, when the next mark is detected, the position of that mark becomes the starting point for the next good mark detection calculation and so on. The starting point is "reset" with each good or bad mark. This feature allows the product to re-synchronize, if necessary, due to process issues like product shift, etc. In contrast, Absolute registration determines all good marks based on the very first mark. The position of the second and each subsequent mark is compared to an integer multiple of Distance from the very first mark. This method insures the product will always register to a known fixed distance.
  - The third bit, 0004H, must always be 0. Mark-to-machine registration requires time-based capture.

### 2.2.7.3.1 Arguments

#### 2.2.7.4.2.1 Input

<b>En</b>	<b>Description</b>	Rising edge of EN enables execution
	<b>Data type</b>	BOOL

	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Axis whose position is used to determine a good mark.
	<b>Data type</b>	Axis_Ref
	<b>Range</b>	The range of .AXIS_NUM is [1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	n/a
<b>TriggerInput</b>	<b>Description</b>	<p>Structure specifying the fast input. The structure elements are:</p> <p><b>InputID</b> INT  0 = Capture Engine 0  1 = Capture Engine 1  Range is [0, 1]  For information on configuring the capture engines, refer to <a href="#">AKD Capture Engine Configuration</a>.</p> <p><b>Direction</b> INT  1 = rising edge, 2 = falling edge, 3 = NA, 4 = toggle between both, falling edge first, 5 = toggle between both, rising edge first, range = [1,5]</p> <p><b>TrigID</b> INT  Axis number of the fast input. Zero indicates this trigger axis is to be the same as the Axis input. range = [0,256]</p>
		<p style="text-align: center;"><b>NOTE</b></p> <p><b>TrigMode</b> INT (TriggerInput.TrigMode) is not presently supported by this function. The TriggerInput.Mode may be supported in a future software version.</p>
	<b>Data type</b>	TRIGGER_REF
	<b>Range</b>	
	<b>Unit</b>	
	<b>Default</b>	
<b>Distance</b>	<b>Description</b>	This is the expected distance between good marks. Along with Tolerance and Ignore, this value is used to determine if the mark detected by the fast input is a good mark.
	<b>Data type</b>	LREAL
	<b>Range</b>	When converted to feedback units, the range is $[-2^{51}, 2^{51}-1]$ . This value must have the same sign as Ignore.
	<b>Unit</b>	user units
	<b>Default</b>	n/a

<b>Tolerance</b>	<b>Description</b>	This value specifies the distance, plus or minus, about Distance to determine if the mark detected by the fast input is a good mark.
	<b>Data type</b>	LREAL
	<b>Range</b>	When converted to feedback units, the range is $[0, 2^{51}-1]$
	<b>Unit</b>	user units
	<b>Default</b>	n/a
<b>Ignore</b>	<b>Description</b>	This value specifies the distance after the previous good mark in which any detected marks are ignored.
	<b>Data type</b>	LREAL
	<b>Range</b>	When converted to feedback units, the range is $[-2^{51}, 2^{51}-1]$ . This value must have the same sign as Distance.
	<b>Unit</b>	user units
	<b>Default</b>	n/a
<b>Target</b>	<b>Description</b>	This is the target position. This position is compared to the actual position captured by the fast input to determine the amount of registration compensation to apply.
	<b>Data type</b>	LREAL
	<b>Range</b>	When converted to feedback units, the range is: <ul style="list-style-type: none"> <li>• <math>[-2^{51}, 2^{51}-1]</math> if PosAxis' rollover value is zero</li> <li>• <math>[0, \text{PosAxis' Rollover Value}]</math> if PosAxis' rollover value is non-zero (i.e., <math>\geq 0 &lt; \text{PosAxis' Rollover Value}</math>)</li> </ul>
	<b>Unit</b>	user units
	<b>Default</b>	n/a
<b>Position</b>	<b>Description</b>	The position the axis is set to when a good registration mark occurs. If the "Inhibit Reference on Good Mark" option is specified for the Option argument (see Options Table below), then this argument is not used and the position of the axis is not changed when a registration mark is encountered.
	<b>Data type</b>	LREAL
	<b>Range</b>	When converted to feedback units, the range is: <ul style="list-style-type: none"> <li>• <math>[-2^{51}, 2^{51}-1]</math> if PosAxis' rollover value is zero</li> <li>• <math>[0, \text{PosAxis' Rollover Value}]</math> if PosAxis' rollover value is non-zero (i.e., <math>\geq 0 &lt; \text{PosAxis' Rollover Value}</math>)</li> </ul>
	<b>Unit</b>	user units

	<b>Default</b>	n/a
<b>PosAxis</b>	<b>Description</b>	The position of this axis at the time the fast input occurs is compared to the Target position to determine the amount of registration compensation to apply.
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	The range of .AXIS_NUM is [1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	n/a
<b>CompAxis</b>	<b>Description</b>	The calculated registration compensation is applied to this axis.
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	The range of .AXIS_NUM is [1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	n/a
<b>PosTolerance</b>	<b>Description</b>	This value specifies the distance, plus or minus, about the Target position to determine if the position will be accepted and compensation value is calculated and applied.
	<b>Data type</b>	LREAL
	<b>Range</b>	When converted to feedback units, the range is $[-2^{51}, 2^{51}-1]$
	<b>Unit</b>	user units
	<b>Default</b>	n/a
<b>RatioNumerator</b>	<b>Description</b>	This value is typically the number of User Units of CompAxis motion for one product cycle. This value is used with RatioDenominator to create a conversion factor for calculating the compensation value when PosAxis and CompAxis are different axes.
	<b>Data type</b>	DINT
	<b>Range</b>	When converted to feedback units, the range is [1,4294967295]
	<b>Unit</b>	user units
	<b>Default</b>	n/a
<b>RatioDenominator</b>	<b>Description</b>	This value is typically the number of User Units of PosAxis motion for one product cycle. This value is used with RatioNumerator to create a conversion factor for calculating the compensation value when PosAxis and CompAxis are different axes.
	<b>Data type</b>	DINT

	<b>Range</b>	When converted to feedback units, the range is [1,4294967295]	
	<b>Unit</b>	user units	
	<b>Default</b>	n/a	
<b>Options</b>	<b>Description</b>	Each bit enables/disables an option. The following table defines the bits. Any bits not defined are reserved. The third bit, 0004H, must be 0.	
	<b>Data type</b>	UINT	
	<b>Range</b>	refer to the following options table	
	<b>Unit</b>	n/a	
	<b>Default</b>	n/a	

Hexadecimal	Decimal	Option	Description
0001 H	1	Absolute/Resetting	0 = Resetting, 1 = Absolute
0002 H	2	Reserved	0
0004 H	4	Time/position based capture	0 = time based capture, 1 = position based capture
0008 H	8	Inhibit Reference on Good Mark	0 = Perform reference, 1 = inhibit reference When this bit is set, the Position function block argument is unused and the axis position is not changed when a registration mark is encountered.
0010H	16	Inhibit Master Compensation	0 = Perform Master Compensation, 1 = Inhibit Master Compensation
0020H	32	Inhibit Slave Compensation	0 = Perform Slave Compensation, 1 = Inhibit Slave Compensation.

Table 1-2: MC\_MachRegist Options Table

### 2.2.7.5.3.2 Outputs

<b>RegistOn</b>	<b>Description</b>	Indicates registration is activated
	<b>Data type</b>	BOOL
<b>Aborted</b>	<b>Description</b>	Indicates registration has been terminated by MC_StopRegist.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified or registration was terminated due to an error
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is TRUE. See table in <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Data type</b>	INT

#### TIP

To use Capture Engine 1 modify the input PDOs that are used and add the Latch Position 1 parameter.

### 2.2.7.6.4 Related Functions

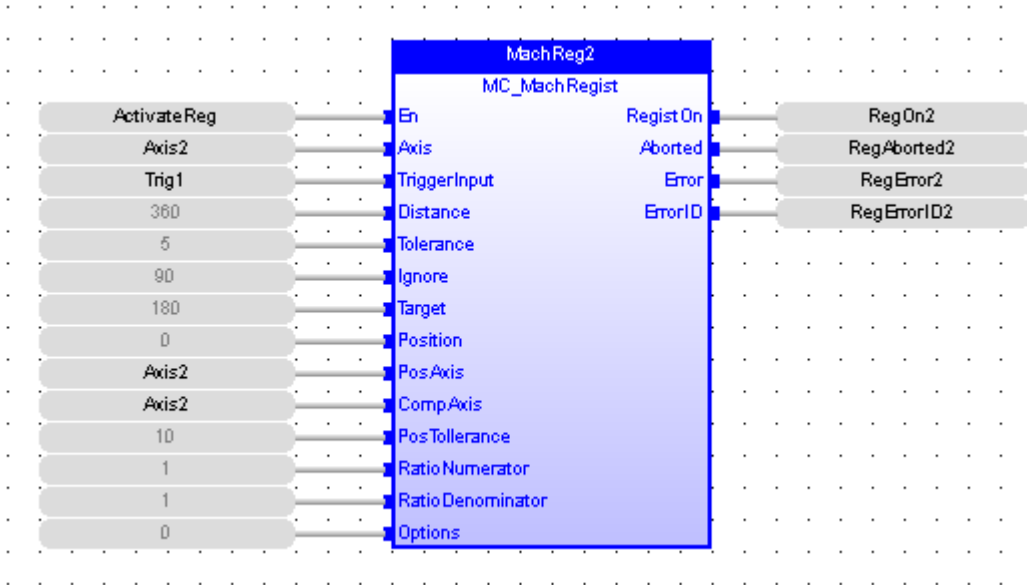
[MC\\_ReadParam](#)

[MC\\_StopRegist](#)

[MC\\_WriteParam](#)

### 2.2.7.7.5 Examples

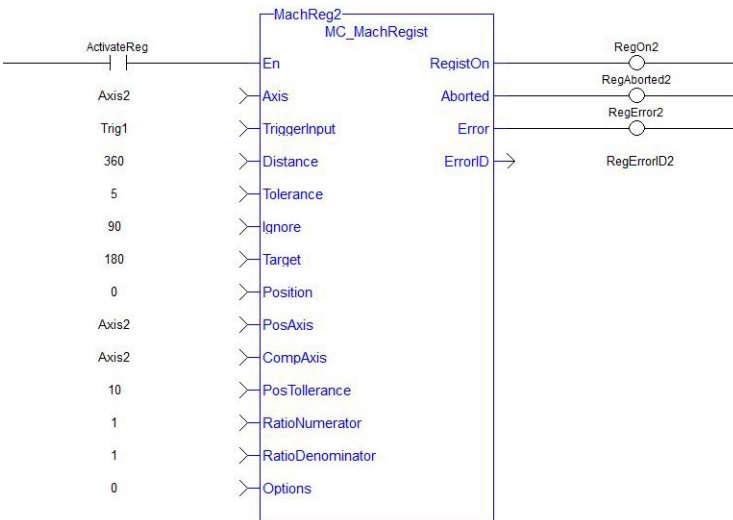
#### 2.2.7.8.6.1 Function Block



#### 2.2.7.9.7.2 Instruction List

```
CAL Inst_MC_MachRegist( ActivateReg, Axis2, Trig1, 360, 5, 90, 180, 0, Axis2, Axis2, 5, 1, 1, 0 )
```

#### 2.2.7.10.8.3 Ladder Diagram



#### 2.2.7.11.9.4 Structured Text

```
Inst_MC_MachRegist( ActivateReg, Axis2, Trig1, 360, 5, 90, 180, 0, Axis2, Axis2, 10, 1, 1, 1, 0 )
```



## 2.2.7.12 MC\_MarkRegist

### 2.2.7.13.1 Description

This function block enables mark-to-mark registration and can be used on any servo or digitizing axis and with any move type. This function block is most frequently used in master/slave applications.



Used with ...	Effect
Non-slave moves	Resets the axis position when a good mark is captured by the fast input.
Slave moves	In addition to resetting the axis position, applies a compensation offset to correct for the difference between the target mark-to-mark distance and the measured mark-to-mark distance. This provides the ability to compensate for product or process inconsistencies providing a system that remains synchronized with no accumulated error and maintaining repeatable accuracy throughout the process.

- A positive transition of the **En** input will start registration. The application may change the registration parameters while registration is active by changing the input values and causing another positive transition of the En input. The function block will then read and apply the new values.
- The axis number at the **Axis** input identifies the axis of registration. If Axis is a master axis for another axis's slave move, Master Registration will be activated. Master Registration calculates a compensation that is added to the master offset of its slaves. This offset shifts the position of the master axis as seen by its slaves. The compensation is not applied to the master axis, but to its slaves. If Axis is a slave axis, Slave Registration will be activated. Slave Registration calculates a compensation that is added to the slave offset of the axis. This compensation value is applied directly to the slave axis.
- The **Distance**, **Tolerance**, and **Ignore** inputs are used to determine whether or not the registration mark is good. For a mark to be recognized as good, it must be outside of the Ignore distance and the correct Distance from the previous mark +/- the Tolerance window. A mark is considered bad if it occurs outside of the "good tolerance band" and is not ignored. Both good marks and bad marks are recognized as marks, ignored marks are not recognized. If all marks are to be recognized as good marks, enter 0 at both **Distance** and **Tolerance**.
- The **Distance** value defines the distance between good marks. In Clear Lane and Product registration the Distance input value typically is the same as the **Target** input value. However in Print registration the Distance is typically not the same as Target.
- The **Tolerance** value is the distance, plus and minus, about **Distance**. Marks that are detected within this window are considered good marks and registration will occur. Marks that are detected outside this window and outside the Ignore band, are considered bad marks and registration will not occur. This window should be large enough to allow for the worst case error in the distance between the previous mark and the current mark.
- The **Ignore** value defines the distance from the previous mark where all marks detected by the fast input will be ignored. This is crucial when registering products that do not have Clear Lane registration marks.
- The **Target** input is the expected distance between good registration marks and is used to calculate how much registration compensation is to be applied when a registration mark is considered good. In many applications this is often equivalent to the product length or the cycle length. When a good mark

is detected, the actual distance between the good mark and the previous mark is determined and compared to the Target distance to calculate a correction. The registration correction will only be applied with master/slave move types and always affects the slave axis.

- The **Position** input is the position value that the registration Axis position will be reset to when a good registration mark is detected.
- The **Option** input defines various modes of operation for registration. The first bit, 0001H, selects Absolute or Resetting. This refers to the way in which the second mark and all subsequent marks are determined to be good marks. With both registration schemes, the very first mark detected is the starting point. With Resetting registration, when the next mark is detected, the position of that mark becomes the starting point for the next good mark detection calculation and so on. The starting point is “reset” with each good or bad mark. This feature allows the product to re-synchronize, if necessary, due to process issues like product shift, etc. In contrast, Absolute registration determines all good marks based on the very first mark. The position of the second and each subsequent mark is compared to an integer multiple of Distance from the very first mark. This method insures the product will always register to a known fixed distance.

#### 2.2.7.14.2 Arguments

##### 2.2.7.15.3.1 Input

<b>En</b>	<b>Description</b>	Rising edge of EN enables execution
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Axis to apply registration to
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	The range of .AXIS_NUM is [1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	n/a

<b>TriggerInput</b>	<b>Description</b>	<p>Structure specifying the fast input.</p> <p>The structure elements are:</p> <p><b>InputID</b> INT  0 = Capture Engine 0  1 = Capture Engine 1  Range is [0, 1]  For information on configuring the capture engines, refer to <a href="#">AKD Capture Engine Configuration</a>.</p> <p><b>Direction</b>INT  1 = rising edge, 2 = falling edge, 3 = NA, 4 = toggle between both, falling edge first, 5 = toggle between both, rising edge first, range = [1,5]</p> <p><b>TrigID</b>INT  Axis number of the fast input. Zero indicates this trigger axis is to be the same as the Axis input. range = [0,256]</p>
	<b>NOTE</b>	
	<p><b>TrigMode</b> INT (TriggerInput.TrigMode) is not presently supported by this function. The TriggerInput.Mode may be supported in a future software version.</p>	
	<b>Data type</b>	TRIGGER_REF
	<b>Range</b>	
<b>Unit</b>		
<b>Default</b>		
<b>Distance</b>	<b>Description</b>	This is the expected distance between good marks. Along with Tolerance and Ignore, this value is used to determine if the mark detected by the fast input is a good mark.
	<b>Data type</b>	LREAL
	<b>Range</b>	When converted to feedback units, the range is $[-2^{51}, 2^{51}-1]$ . This value must have the same sign as Ignore.
	<b>Unit</b>	user units
	<b>Default</b>	n/a
<b>Tolerance</b>	<b>Description</b>	This value specifies the distance, plus or minus, about Distance to determine if the mark detected by the fast input is a good mark.
	<b>Data type</b>	LREAL
	<b>Range</b>	When converted to feedback units, the range is $[0, 2^{51}-1]$
	<b>Unit</b>	user units
	<b>Default</b>	n/a

<b>Ignore</b>	<b>Description</b>	This value specifies the distance after the previous good mark in which any detected marks are ignored.									
	<b>Data type</b>	LREAL									
	<b>Range</b>	When converted to feedback units, the range is $[-2^{51}, 2^{51}-1]$ . This value must have the same sign as Distance.									
	<b>Unit</b>	user units									
	<b>Default</b>	n/a									
<b>Target</b>	<b>Description</b>	This is the target distance between good marks. This distance is compared to the actual distance measured by the fast input to determine the amount of registration compensation to apply.									
	<b>Data type</b>	LREAL									
	<b>Range</b>	When converted to feedback units, the range is $[-2^{51}, 2^{51}-1]$ . This value must have the same sign as Distance.									
	<b>Unit</b>	user units									
	<b>Default</b>	n/a									
<b>Position</b>	<b>Description</b>	The position the axis is set to when a good registration mark occurs. If the "Inhibit Reference on Good Mark" option is specified for the Option argument (see Options Table below), then this argument is not used and the position of the axis is not changed when a registration mark is encountered.s									
	<b>Data type</b>	LREAL									
	<b>Range</b>	When converted to feedback units, the range is: <ul style="list-style-type: none"> <li>• <math>[-2^{51}, 2^{51}-1]</math> if PosAxis' rollover value is zero</li> <li>• <math>[0, \text{PosAxis' Rollover Value}]</math> if PosAxis' rollover value is non-zero (i.e., <math>\geq 0 &lt; \text{PosAxis' Rollover Value}</math>)</li> </ul>									
	<b>Unit</b>	user units									
	<b>Default</b>	n/a									
<b>Options</b>	<b>Description</b>	Each bit enables/disables an option. The following table defines the bits. Any bits not defined are reserved.									
	<b>Data type</b>	UINT									
	<b>Range</b>	Refer to the following options table.									
	<b>Unit</b>	n/a									
	<b>Default</b>	n/a									
<table border="1"> <thead> <tr> <th>Hexadecimal</th> <th>Decimal</th> <th>Option</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0001 H</td> <td>1</td> <td>Absolute/Resetting</td> <td>0 = Resetting, 1 = Absolute</td> </tr> </tbody> </table>				Hexadecimal	Decimal	Option	Description	0001 H	1	Absolute/Resetting	0 = Resetting, 1 = Absolute
Hexadecimal	Decimal	Option	Description								
0001 H	1	Absolute/Resetting	0 = Resetting, 1 = Absolute								

Hexadecimal	Decimal	Option	Description
0002 H	2	Reserved	0
0004 H	4	Time/position based capture	0 = time based capture, 1 = position based capture
0008 H	8	Inhibit Reference on Good Mark	0 = Perform reference, 1 = inhibit reference When this bit is set, the Position function block argument is unused and the axis position is not changed when a registration mark is encountered.
0010H	16	Inhibit Master Compensation	0 = Perform Master Compensation, 1 = Inhibit Master Compensation
0020H	32	Inhibit Slave Compensation	0 = Perform Slave Compensation, 1 = Inhibit Slave Compensation.

Table 1-3: MC\_MarkRegist Options Table.

### TIP

To use Capture Engine 1 modify the input PDOs that are used and add the Latch Position 1 parameter.

#### 2.2.7.16.4.2 Outputs

<b>RegistOn</b>	<b>Description</b>	Indicates that registration is active.
	<b>Data type</b>	BOOL
<b>Aborted</b>	<b>Description</b>	Indicates registration has been terminated by MC_StopRegist.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified or registration was terminated due to an error
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is TRUE. See table in <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Data type</b>	INT

#### 2.2.7.17.5 Related Functions

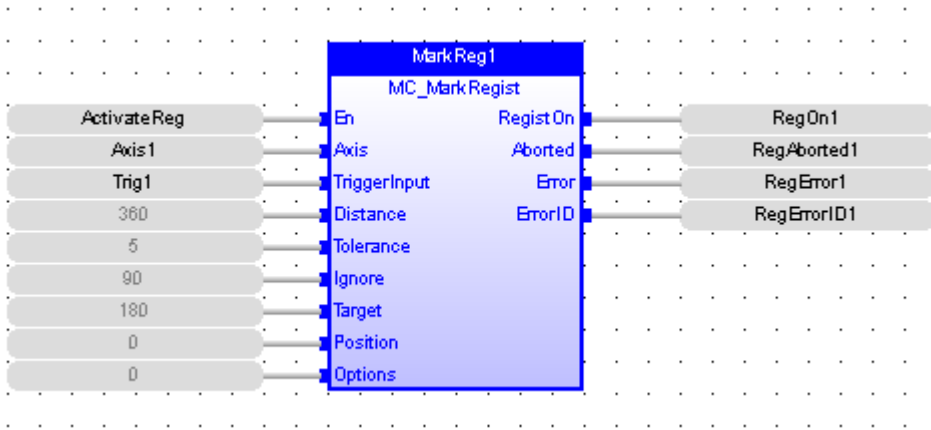
[MC\\_ReadParam](#)

[MC\\_Stop](#)

[MC\\_WriteParam](#)

#### 2.2.7.18.6 Examples

#### 2.2.7.19.7.1 Function Block

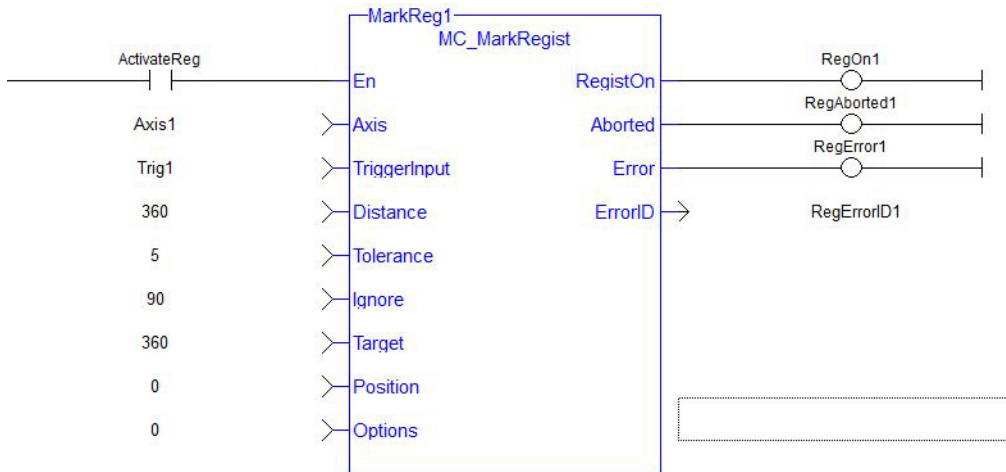


### 2.2.7.20.8.2 Instruction List

```

CAL Inst_MC_MarkRegist( ActivateReg, Axis1, Trig1, 360, 5, 90, 360, 0, 0 )
    
```

### 2.2.7.21.9.3 Ladder Diagram



### 2.2.7.22.10.4 Structured Text

```

Inst_MC_MarkRegist( ActivateReg, Axis1, Trig1, 360, 5, 90, 360, 0, 0 );
    
```

**2.2.7.23 MC\_StopRegist** PLCopen

**2.2.7.24.1 Description**

This function will turn off registration for the specified axis and disarm the specified fast input.



**2.2.7.25.2 Arguments**

**2.2.7.26.3.1 Input**

<b>En</b>	<b>Description</b>	Enables execution
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Axis registration to turn off
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	The range of .AXIS_NUM is [1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	n/a
<b>TriggerInput</b>	<b>Description</b>	<p>Structure specifying the fast input to disarm. The structure elements are:</p> <p><b>InputID</b> INT                      0 = Capture Engine 0                      1 = Capture Engine 1                      Range is [0, 1]                      For information on configuring the capture engines, refer to <a href="#">AKD Capture Engine Configuration</a>.</p> <p><b>Direction</b> INT                      1 = rising edge, 2 = falling edge, 3 = NA, 4 = toggle between both, falling edge first, 5 = toggle between both, rising edge first,                      range = [1,5]</p> <p><b>TrigID</b> INT                      Axis number of the fast input. 0 indicates this trigger axis is to be the same as the Axis input.                      range = [0,256]</p> <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p><b>NOTE</b></p> <p><b>TrigMode</b> INT (TriggerInput.TrigMode) is not presently supported by this function. The TriggerInput.Mode may be supported in a future software version.</p> </div>

<b>Data type</b>	TRIGGER_REF
<b>Range</b>	
<b>Unit</b>	
<b>Default</b>	

#### 2.2.7.27.4.2 Outputs

<b>OK</b>	<b>Description</b>	Indicates function executed successfully
	<b>Data type</b>	BOOL

#### **TIP**

To use Capture Engine 1 modify the input PDOs that are used and add the Latch Position 1 parameter.

#### 2.2.7.28.5 Related Functions

[MC\\_MachRegist](#)

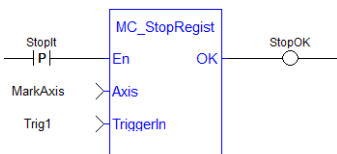
[MC\\_MarkRegist](#)

#### 2.2.7.29.6 Examples

##### 2.2.7.30.7.1 Function Block



##### 2.2.7.31.8.2 Ladder Diagram



##### 2.2.7.32.9.3 Structured Text

```
StopOK := MC_StopRegist( Axis1, Trig1);
```



## 2.2.8 Superimposed Axes

This feature allows the application program to superimpose the moves of multiple axes ("Superimposed Axes") on top of the move of another axis ("Receiving Axis"). This is performed internally by adding the command deltas of the Superimposed Axes to the command delta of the Receiving Axis. Up to four different Superimposed Axes can be superimposed upon a Receiving Axis.

See "MC\_AddSuperAxis" (p. 545), "MC\_RemSuperAxis" (p. 547) and [PLCopen Function Blocks - Overview](#) for more information.

### 2.2.8.1 MC\_AddSuperAxis

This function will add a Superimposed Axis to the Axis's list of assigned superimposed axes. While the Superimposed Axis is on this list, its command deltas will be added to the Axis's command deltas. Up to four different superimposed axes can be on an axis's list. The `Axis` and the `SuperimposedAxis` must have the same update rate. The `OK` output will go high to indicate that the function executed successfully. If the `OK` output does not go high, one of the following errors was detected:

- Axis and SuperimposedAxis do not have the same update rate
- Four different superimposed axes have already been assigned to Axis
- Axis is not a valid axis - Axis is not a servo or virtual axis
- SuperimposedAxis is not a valid axis number
- SuperimposedAxis is not a servo or virtual axis

#### 2.2.8.2.1 Inputs

<b>En</b>	<b>Description</b>	Enables Execution
	<b>Data Type</b>	BOOL
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	n/a
<b>Axis</b>	<b>Description</b>	Axis to receive the additional superimposed axis's command delta
	<b>Data Type</b>	AXIS_REF
	<b>Range</b>	.AXIS_NUM [1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	n/a
<b>SuperimposedAxis</b>	<b>Description</b>	Axis number of the superimposed axis whose command delta will be added to delta of <code>Axis</code>
	<b>Data Type</b>	UINT
	<b>Range</b>	[1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	n/a

#### 2.2.8.3.2 Outputs

<b>OK</b>	<b>Description</b>	Execution successful
	<b>Data Type</b>	BOOL
	<b>Range</b>	n/a

### 2.2.8.4.3 Examples

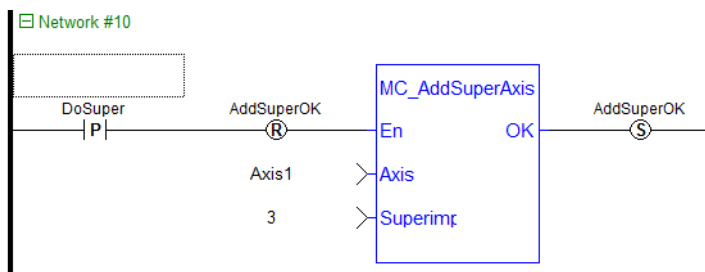
#### 2.2.8.5.4.1 Structured Text

```
AddOKST := MC_AddSuperAxis( Axis1, 3 );|
```

#### 2.2.8.6.5.2 Function Block Diagram



#### 2.2.8.7.6.3 Ladder Diagram



#### 2.2.8.8.7 Related Functions

["MC\\_RemSuperAxis" \(p. 547\)](#)

### 2.2.8.9 MC\_RemSuperAxis PLCopen

This function removes the Superimposed Axis from the Axis's list of assigned superimposed axes. If the value at `SuperimposedAxis` is 0 all the assigned superimposed axes will be removed from Axis's list. The `OK` output will go high to indicate that the function executed successfully. If the `OK` output does not go high, one of the following errors was detected:

- Axis is not a valid axis
- Axis is not a servo or virtual axis

#### 2.2.8.10.1 Inputs

<b>En</b>	<b>Description</b>	Enables Execution
	<b>Data Type</b>	BOOL
	<b>Range</b>	-
	<b>Unit</b>	n/a
	<b>Default</b>	
<b>Axis</b>	<b>Description</b>	Axis whose list of assigned superimposed axes will be updated.
	<b>Data Type</b>	AXIS_REF
	<b>Range</b>	.AXIS_NUM [1,256]
	<b>Unit</b>	n/a
	<b>Default</b>	n/a
<b>SuperimposedAxis</b>	<b>Description</b>	Axis number of the superimposed axis that will be removed from Axis's list of assigned superimposed axes. A value of 0 will remove all superimposed axes from Axis's list.
	<b>Data Type</b>	UINT
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	n/a

#### 2.2.8.11.2 Outputs

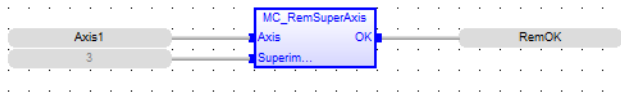
<b>OK</b>	<b>Description</b>	Execution successful
	<b>Data Type</b>	BOOL
	<b>Range</b>	n/a

#### 2.2.8.12.3 Examples

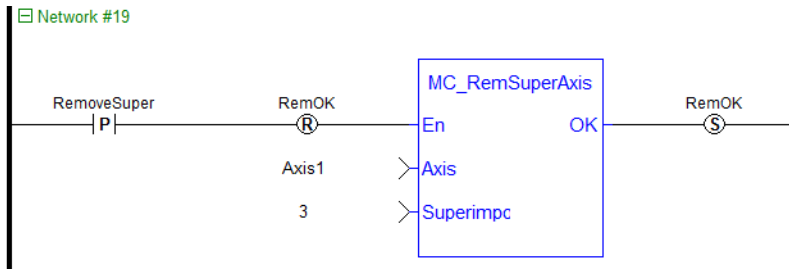
#### 2.2.8.13.4.1 Structured Text

```
RemOK := MC_RemSuperAxis (Axis1, 3);
```

### 2.2.8.14.5.2 Function Block Diagram



### 2.2.8.15.6.3 Ladder Diagram



### 2.2.8.16.7 Related Functions

["MC\\_AddSuperAxis" \(p. 545\)](#)

## 2.3 MotionLibrary- Common

Functions sorted in alphabetical order.

Name	Description	Return type
"MC_ErrorDescription" (p. 549)	Return a text description corresponding to a motion control error ID code	STRING
"MLMotionCycleTime" (p. 569)	Returns the Motion Base Cycle time in Seconds	
"MLMotionInit" (p. 571)	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
"MLMotionRstErr" (p. 573)	Clears motion engine errors, motion bus driver errors, and EtherCAT network errors. MLMotionRstErr will return the motion engine status to the Stopped state, if an error condition was cleared successfully. Returns TRUE if the function succeeded.	BOOL
"MLMotionStart" (p. 574)	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" (p. 576)	Returns the status of the motion engine 0: Not initialized 1: Running 2: Stopped 3: Error	None
"MLMotionStop" (p. 578)	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" (p. 579)	Prints the system time to the log	BOOL
"MLProfileBuild" (p. 552)	Builds a cam profile from application data	See "Output" (p. 555)
"MLProfileCreate" (p. 561)	Creates a new cam profile object	None
"MLProfileInit" (p. 563)	Initializes a previously created cam profile object	BOOL
"MLProfileRelease" (p. 566)	Removes a Profile so the Profile ID may be used by a different or new Profile.	See "Output" (p. 567)

### 2.3.1 Motion Library - Common - Info

Name	Description	Return type
"MC_ErrorDescription" (p. 549)	Converts the PLCopen error IDs into message strings.	String

#### 2.3.1.1 MC\_ErrorDescription

PLCopen ✓

Pipe Network ✓

This function converts the PLCopen error IDs into message strings which can be used for display or logging.

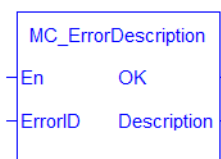


Figure 1-91: MC\_ErrorDescription Function Block

**2.3.1.2.1 Arguments****2.3.1.3.2.1 Inputs**

<b>En</b>	<b>Description</b>	If True, then this function will convert the Error Id into a string message
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ErrorID</b>	<b>Description</b>	Error ID generated from a PLCopen Function Block. See <a href="#">PLCopen Function Block ErrorID Output</a> for output details.
	<b>Data type</b>	INT
	<b>Range</b>	0,69
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.3.1.4.3.2 Outputs**

<b>OK</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Description</b>	<b>Description</b>	String error description
	<b>Data type</b>	STRING
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.3.1.5.4 Examples****2.3.1.6.5.1 Structured Text**

```
Description := MC_ErrorDescription(ErrorID);
```

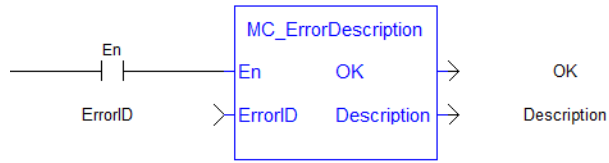
**2.3.1.7.6.2 IL**

Not applicable

**2.3.1.8.7.3 Function Block**



### 2.3.1.9.8.4 Ladder Diagram



## 2.3.2 Motion Library - Common - Profiles

Name	Description	Return type
"MLProfileBuild" (p. 552)	Builds a cam profile from application data	See "Output" (p. 555)
"MLProfileCreate" (p. 561)	Creates a new cam profile object	None
"MLProfileInit" (p. 563)	Initializes a previously created cam profile object	BOOL
"MLProfileRelease" (p. 566)	Removes a Profile so the Profile ID may be used by a different or new Profile.	See "Output" (p. 567)

### 2.3.2.1 MLProfileBuild

#### 2.3.2.2.1 Description

This Function Block allows the application to create a cam profile that may be executed by a cam block in PipeNetwork or PLCopen. This Function Block will take input as cam data (see [Cam Profile Editor's Cam Table](#) for information) and profile properties from application data memory and compile the input data to a form the controller can use to calculate cam positions. The input cam data and profile properties are similar to the cam data entered in the IDE's Cam Editor and the runtime's Cam Profile Properties dialog. MLProfileBuild internally perform two functions:

1. Compile the cam data (like the cam editor performs in the IDE).
2. Puts the compiled profile into the profile object so it can be used by other Profile Function Blocks (provides similar functionality to "MLProfileInit" (p. 563)).

#### NOTE

Prior to using MLProfileBuild you must call "MLProfileCreate" (p. 561) to create the profile object. The ID output of MLProfileCreate is then used as the ProfileID input to MLProfileBuild. MLProfileCreate must be performed in the application *before* the "MLMotionStart" (p. 574) command is executed.

MLProfileBuild will compile the cam profile data specified at the CamData input and write the resulting profile to the CAM Profile object specified at input ProfileID. The created profile can then be used as an input to PLCopen Cam Function Blocks ([MC\\_CamTblSelect](#), [MC\\_CamIn](#), [MC\\_CamOut](#)), or any Pipe network Cam Profile Function/Function Blocks. When the operation is complete, the Done output will go high. If an error is encountered, the Error output will go high and the ErrorID output will return a error code. If the Error can be attributed to a specific profile element in the CamData array, ErrorElem will attempt to indicate the element in error.

#### 2.3.2.3.2.1 CamProps\_Ref Structure

The cam properties structure (CamProps\_Ref) will contain the following data members:

Parameter	Type	Description
InputScale	LREAL	The input amplitude or master axis multiplier applied to the CAM profile
OutputScale	LREAL	The output amplitude or slave axis multiplier applied to the CAM profile
InputOffset	LREAL	input offset or master axis shift applied to the CAM profile
OutputOffset	LREAL	The output offset or slave axis shift applied to the CAM profile

See [Master/Input offset](#) for more information about the parameters which transform the cam profile.

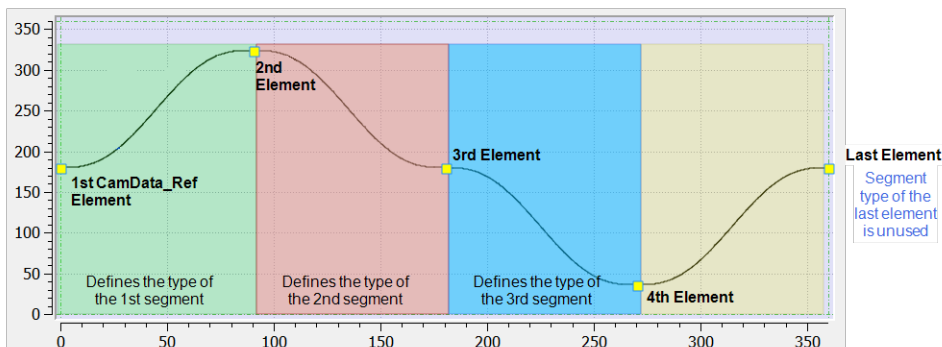
#### 2.3.2.4.3.2 CamData\_Ref Structure

The Cam\_Data function block input will be an array of CamData\_Ref structures. Each element of the structure will contain the following data members:



Parameter	Type	Description
MasterIn	LREAL	master position (in the unit range [0 - InputScale])
SlaveOut	LREAL	slave position (in the unit range [0 - OutputScale])
SegType	UINT	Defines the segment type for the segment following the master positions defined by <b>MasterIn</b> . CAM_SEGMENT_TYPE_POINT = Point 5 <sup>th</sup> order polynomial) CAM_SEGMENT_TYPE_LINE = Line (constant velocity segment) CAM_SEGMENT_TYPE_PARABOLIC = Parabolic (constant acceleration) See <a href="#">Cam Profile Segment Overview</a> for information on the segment types.
Vel	LREAL	Cam velocity at the master position specified by <b>MasterIn</b> . Units: (slave position user units) / (master position user units)
Accel	LREAL	<u>For CAM_SEGMENT_TYPE_POINT:</u> <b>Accel</b> represents the cam acceleration at the master position specified by <b>MasterIn</b> . Units: (slave position user units) / (master position user units) <u>For CAM_SEGMENT_TYPE_LINE and CAM_SEGMENT_TYPE_PARABOLIC:</u> <b>Accel</b> is ignored.

The type of the N<sup>th</sup> cam segment is defined by the N<sup>th</sup> Cam\_Data element. Since the cam will be constructed with one less segment than the Cam\_Data elements, the last element's SegType will not be used.



See [Cam Profile Editor's Cam Table](#) for more information.

### 2.3.2.5.4 Arguments

#### 2.3.2.6.5.1 Input

<b>Enable</b>	<b>Description</b>	Enable execution. Starts on rising edge.
	<b>Data Type</b>	BOOL
	<b>Range</b>	
	<b>Unit</b>	
	<b>Default</b>	

<b>Cam_Props</b>	<b>Description</b>	Structures containing the cam profile properties
	<b>Data Type</b>	CamProps_ref
	<b>Range</b>	
	<b>Unit</b>	
	<b>Default</b>	
<b>Cam_Data</b>	<b>Description</b>	Array of structures containing the cam profile data
	<b>Data Type</b>	CamData_ref
	<b>Range</b>	N=3 to 2000 elements
	<b>Unit</b>	
	<b>Default</b>	3 elements minimum size
<b>CamDataCount</b>	<b>Description</b>	Number of elements in the Cam_Data array to be used.
	<b>Data Type</b>	UINT
	<b>Range</b>	3 - 2000
	<b>Unit</b>	elements
	<b>Default</b>	3
<b>ProfileID</b>	<b>Description</b>	ID number of a created CAM Profile
	<b>Data Type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	--
<b>Cyclic</b>	<b>Description</b>	False: one time through the profile; True: repeating profile
	<b>Data Type</b>	BOOL
	<b>Range</b>	0-1 (FALSE/TRUE)
	<b>Unit</b>	
	<b>Default</b>	

<b>Options</b>	<b>Description</b>	Describes the combinations of segments that may be used to build a cam profile.
	<b>Data Type</b>	UINT
	<b>Range</b>	CAM_PROFILE_OPTION_DEFAULT: Allows the use of point and line segments. CAM_PROFILE_OPTION_PARABOLIC: Allows the use of line and parabolic segments.
	<b>Unit</b>	
	<b>Default</b>	

**NOTE**

The DEFAULT and PARABOLIC options cannot be combined. A cam profile can only use point and line segments, or parabolic and line segments. Point and parabolic segments cannot both be used in the same profile.

#### 2.3.2.7.6.2 Output

<b>Done</b>	<b>Description</b>	Indication of whether or not the profile was successfully compiled and built.
	<b>Data Type</b>	BOOL
	<b>Range</b>	0-1 (FALSE/TRUE)
	<b>Unit</b>	
<b>Busy</b>	<b>Description</b>	Indication that the function block is executing. TRUE if executing. False if not executing.
	<b>Data Type</b>	BOOL
	<b>Range</b>	0-1 (FALSE/TRUE)
	<b>Unit</b>	
<b>Err</b>	<b>Description</b>	Indication that the function did not execute correctly. ErrorID output will be valid and indicate the reason.
	<b>Data Type</b>	BOOL
	<b>Range</b>	0-1 (FALSE/TRUE)
	<b>Unit</b>	
<b>ErrorID</b>	<b>Description</b>	Indication of the reason for the failure to execute properly. See <a href="#">"Error Codes" (p. 556)</a> table.
	<b>Data Type</b>	INT
	<b>Range</b>	
	<b>Unit</b>	

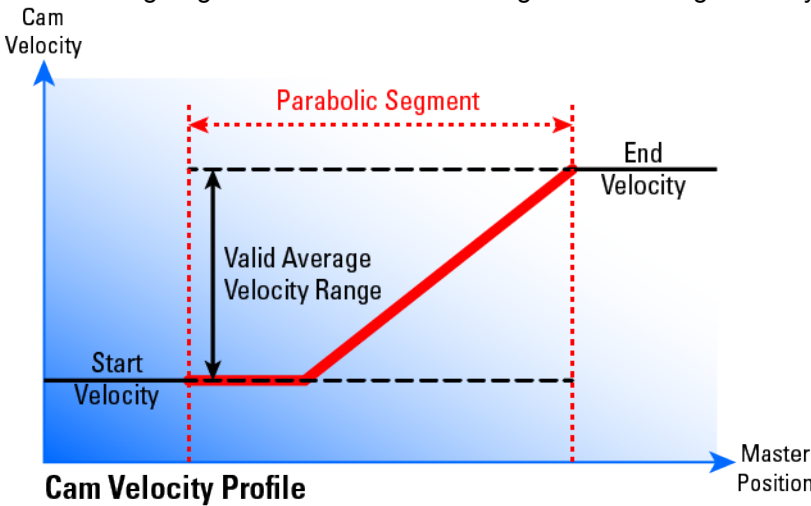
<b>ErrorElem</b>	<b>Description</b>	The array element number of the cam data where an error is detected
	<b>Data Type</b>	UINT
	<b>Range</b>	
	<b>Unit</b>	

### 2.3.2.8.7 Error Codes

#### NOTE

If **Cyclic** is TRUE and the Vel/Accel of the first and last elements do not match, MLProfileBuild will automatically copy the first element's vel/accel to the last element's. A LOG warning message will be posted indicating that this change has occurred.

ErrorID	Description
100	Cam_Data array does not have CamDataCount elements. The Cam_Data array is not large enough to hold the specified number of elements.
101	Invalid master or slave scale. The scale cannot be less than zero.
102	Element master or slave position is outside the range defined by Cam_Props.
103	A segment type was specified that is not supported by the value of the Options argument.
104	Master position of an element is too close to the master position of a previous element. $0.0005 * \text{InputScale}$ (see "CamProps_Ref Structure" on page 552) is the minimum distance allowed.. Each element is compared to all previous elements. If the master distance between any two elements in the list are too close, this error will be generated.
106	Invalid profile ID. This can occur if the profile ID: <ul style="list-style-type: none"> <li>1. does not exist</li> <li>2. has not been created yet</li> <li>3. profile ID is not a profile</li> </ul>
107	CamDataCount exceeds maximum array size of 2000.
108	Profile is currently in use.
109	Attempting to build a profile already containing elements. Profile needs to be released first using MLProfileRelease.
110	The controller is running low on memory and could not allocate memory to hold the cam table data.
111	CamDataCount is not large enough. The minimum allowed value is 3.
112	For CAM_PROFILE_OPTION_PARABOLIC: Elements are not sorted in increasing order by master position. After the first element, each element must have its master position be greater than the master position of the previous element.

ErrorID	Description
113	<p>For CAM_SEGMENT_TYPE_PARABOLIC: The average velocity for the segment is outside of range defined by start and end element velocities.</p> <p>The following diagram illustrates the valid range for the average velocity.</p>  <p><b>Cam Velocity Profile</b></p>
200	First element's <code>MasterIn</code> value not equal to zero.
201	Last element's <code>MasterIn</code> value does not equal value of X-amplitude.
202	Cannot modify the first element in the cam element table. <code>SlaveOut</code> value is outside the output range specified by <code>Cam_Props</code> .
203	Cannot modify the last element in the cam element table. <code>SlaveOut</code> value is outside the output range specified by <code>Cam_Props</code> .

### 2.3.2.9.8 Related Functions

- ["MLCamInit"](#) (p. 210)
- ["MLCamSwitch"](#) (p. 212)
- ["MLProfileCreate"](#) (p. 561)
- ["MLProfileInit"](#) (p. 563)
- ["MLProfileRelease"](#) (p. 566)
- ["MC\\_CamIn"](#) (p. 479)
- ["MC\\_CamOut"](#) (p. 486)
- [MC\\_CamTblSelect](#)

### 2.3.2.10.9.1 See Also

- [Cam Profile Segment Overview](#)

### 2.3.2.11.10 Example of How to Use MLProfileBuild

Prior to using `MLProfileBuild` you must first create a profile. This must be done prior to `MLMotionStart`.

```
// Allocate space for a profile that will be built later
profileID := MLProfileCreate('ProfileName');
```

Next you need to define your profile data. This is done by creating an array of `CamData_Ref` structures in the data dictionary and then entering each of your elements into that newly created structure. In this example `ProfileData` is the name of the `CamData_Ref` structure.

```
// Define the profile data
ProfileData[0].MasterIn := 0.0;
ProfileData[0].SlaveOut := 180.0;
ProfileData[0].SegType := CAM_SEGMENT_TYPE_POINT;
ProfileData[0].Velocity := 0.0;
ProfileData[0].Acceleration := 0.0;

ProfileData[1].MasterIn := 180.0;
ProfileData[1].SlaveOut := 324.0;
ProfileData[1].SegType := CAM_SEGMENT_TYPE_POINT;
ProfileData[1].Velocity := 0.5;
ProfileData[1].Acceleration := -0.025;

ProfileData[2].MasterIn := 360.0;
ProfileData[2].SlaveOut := 240.0;
ProfileData[2].SegType := CAM_SEGMENT_TYPE_POINT;
ProfileData[2].Velocity := 0.0;
ProfileData[2].Acceleration := 0.0;
```

Now you need to define your profile properties. This is done by creating a CamProps\_Ref structure in the data dictionary and then entering each of the properties into the newly created structure. In this example ProfileProps is the name of the CamProps\_Ref structure.

```
// Define the profile properties
ProfileProps.InputScale := 360.0; // Must be Positive!
ProfileProps.OutputScale := 360.0; // Must be Positive!
ProfileProps.InputOffset := 0.0;
ProfileProps.OutputOffset := 0.0;
```

Next call the MLProfileBuild Function Block in the IEC language of choice. As part of this call it is recommended that you validate the Done and Error output before proceeding.

```
// Build the profile
Inst_MLProfileBuild( TRUE, ProfileProps, ProfileData, 3,
ProfileID, TRUE, CAM_PROFILE_OPTION_DEFAULT);
```

Finally, after verifying that MLProfileBuild is Done and there are no errors, you can proceed and use the newly generated profile. The next step depends on the motion engine in use.

- PLCopen: call "MC\_CamTblSelect" (p. 497)
- Pipe Network: call either "MLCamInit" (p. 210) or "MLCamSwitch" (p. 212)

#### NOTE

**Pipe Network:** In order to correctly set the cam scales and offsets (defined by the Cam\_Props argument) "MLPrfWriteIScale" (p. 224), "MLPrfWriteOScale" (p. 228), "MLPrfWriteIOffset" (p. 222) and "MLPrfWriteOOffset" (p. 226) must be called before calling "MLCamSwitch" (p. 212),

```
// Switch Pipe Network Profile
MLPrfWriteIScale(profileID, ProfileProps.InputScale);
MLPrfWriteOScale(profileID, ProfileProps.OutputScale);
MLPrfWriteIOffset(profileID, ProfileProps.InputOffset);
```

```

.PrfWriteOffset(profileID, ProfileProps.OutputOffset);
.CamSwitch(PipeNetwork.CAM, profileID);

```

### 2.3.2.12.11 Example of Building a Parabolic Cam Profile

In order to build a parabolic cam profile, your cam data elements must use `CAM_SEGMENT_TYPE_PARABOLIC` or `CAM_SEGMENT_TYPE_LINE` when defining the cam data array:

```

// Define the profile data
ProfileData[0].MasterIn      := 0.0;
ProfileData[0].SlaveOut     := 0.0;
ProfileData[0].SegType      := CAM_SEGMENT_TYPE_PARABOLIC;
ProfileData[0].Velocity     := 0.0;
ProfileData[0].Acceleration := 0.5;

ProfileData[1].MasterIn      := 50.0;
ProfileData[1].SlaveOut     := 150.0;
ProfileData[1].SegType      := CAM_SEGMENT_TYPE_LINE;
ProfileData[1].Velocity     := 5.0;
ProfileData[1].Acceleration := 0.0;           // Not used

ProfileData[2].MasterIn      := 55.0;
ProfileData[2].SlaveOut     := 175.0;
ProfileData[2].SegType      := CAM_SEGMENT_TYPE_PARABOLIC;
ProfileData[2].Velocity     := 5.0;
ProfileData[2].Acceleration := 0.0;           // No limit to the
acceleration rate of the segment.

ProfileData[3].MasterIn      := 105.0;
ProfileData[3].SlaveOut     := 250.0;
ProfileData[3].SegType      := CAM_SEGMENT_TYPE_PARABOLIC;
ProfileData[3].Velocity     := 0.0;
ProfileData[3].Acceleration := 0.5;

ProfileData[4].MasterIn      := 225.0;
ProfileData[4].SlaveOut     := 125.0;
ProfileData[4].SegType      := CAM_SEGMENT_TYPE_PARABOLIC;
ProfileData[4].Velocity     := -10.0;
ProfileData[4].Acceleration := 0.5;

ProfileData[5].MasterIn      := 360.0;
ProfileData[5].SlaveOut     := 0.0;
ProfileData[5].SegType      := CAM_SEGMENT_TYPE_PARABOLIC;           //
Not used
ProfileData[5].Velocity     := 0.0;
ProfileData[5].Acceleration := 0.0;           // Not used

```

When calling the `MLProfileBuild` function block, make sure `CAM_PROFILE_OPTION_PARABOLIC` is specified for the `Option` argument: in the IEC language of choice. As part of this call it is recommended that you validate the **Done** and **Error** output before proceeding.

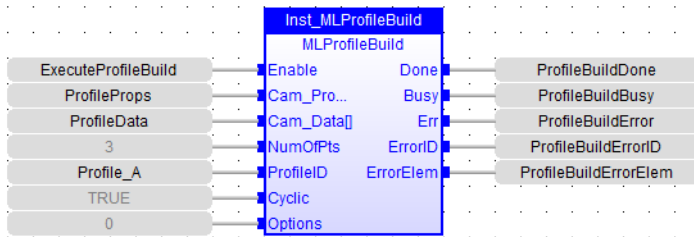
```
// Build the profile
Inst_MLProfileBuild( TRUE, ProfileProps, ProfileData, 3,
ProfileID, TRUE, CAM_PROFILE_OPTION_PARABOLIC);
```

### 2.3.2.13.12 Code Examples

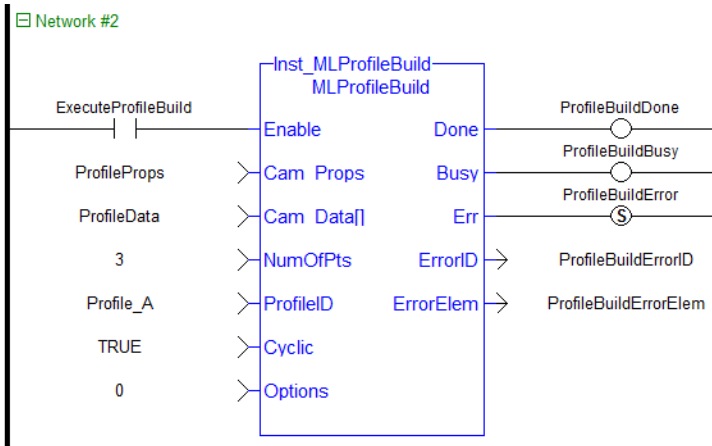
#### Structured Text:

```
// Build the profile
Inst_MLProfileBuild( TRUE, ProfileProps, ProfileData, 3,
ProfileID, TRUE, CAM_PROFILE_OPTION_DEFAULT);
```

#### Function Block Diagram:



#### Ladder Diagram:





### 2.3.2.14 MLProfileCreate

#### 2.3.2.15.1 Description

Creates a new Profile Object for use in a PLC Program or Pipe Network CAM block. This function block is automatically called if a Profile is created in the Project Explorer, with user-defined settings then entered in the CAM Profile Properties screen.

Profiles are created and initiated separately and the shape is modified with the CAM Editor. With the Editor profiles can be changed graphically or by manually changing values in a numeric table relating input and output values with specific slopes. The Cam Editor software tool provides the capability to visualize, analyze, edit, and smooth profiles.

Profile switching can be done on the fly, without losing synchronization and without dead time. In addition, the offsets and ratios of CAM Profiles can be changed on the fly.

#### NOTE

Profile objects are normally created in the Project Explorer. Then you do not have to add MLCamInit function blocks to their programs. By right clicking the Profiles folder under the PLC->Motion Tree, you can select Add new profile. Parameters are entered directly in a pop-up window, and the code is then automatically added to the current project.

#### TIP

This function must be called or executed before "[MLMotionStart](#)" (p. 574) is called.

#### 2.3.2.16.2 Arguments

##### 2.3.2.17.3.1 Input

<b>Name</b>	<b>Description</b>	Name of initialized CAM Profile
	<b>Data type</b>	STRING
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

##### 2.3.2.18.4.2 Output

<b>OK</b>	<b>Description</b>	Indicates the profile has been created
	<b>Data type</b>	BOOL
<b>ID</b>	<b>Description</b>	Returns the ID number of the created CAM Profile
	<b>Data type</b>	DINT
	<b>Unit</b>	n/a

#### 2.3.2.19.5 Related Functions

["MLProfileInit"](#) (p. 563)

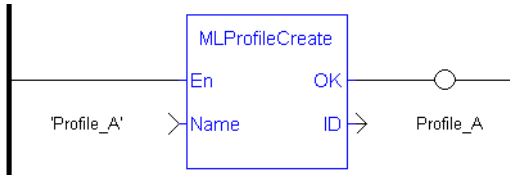
["MLCamInit"](#) (p. 210)

#### 2.3.2.20.6 Example

##### 2.3.2.21.7.1 Structured Text

```
//Create a new Profile  
Profile_A := MLProfileCreate( 'Profile_A' );
```

### 2.3.2.22.8.2 Ladder Diagram



### 2.3.2.23.9.3 Function Block Diagram



### 2.3.2.24 MLProfileInit

#### 2.3.2.25.1 Description

Initializes a previously created CAM Profile object for use in a PLC Program or Pipe Network CAM block. This function block is automatically called if a Profile is created in the Project Explorer, with user-defined settings then entered in the CAM Profile Properties screen.

Profiles are created and initiated separately and the shape is modified with the CAM Editor. With the Editor profiles can be changed graphically or by manually changing values in a numeric table relating input and output values with specific slopes. The Cam Editor software tool provides the capability to visualize, analyze, edit, and smooth profiles.

Profile switching can be done on the fly, without losing synchronization and without dead time. In addition, the offsets and ratios of CAM Profiles can be changed on the fly.

#### NOTE

Profile objects are normally initiated in the Project Explorer. Then you do not have to add MLCamInit function blocks to their programs. By right clicking the Profiles folder under the PLC->Motion Tree, you can select Add new profile. Parameters are entered directly in a pop-up window, and the code is then automatically added to the current project.

#### TIP

Loading a Profile Editor-generated profile into a ProfileID released by MLProfileRelease should be done with care. The MLProfileInit () function call can take in excess of 4 milliseconds to execute. Application execution is suspended during this time until the function call is completed.

#### 2.3.2.26.2 Arguments

##### 2.3.2.27.3.1 Input

<b>ProfileID</b>	<b>Description</b>	ID number of a created CAM Profile
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>FileName</b>	<b>Description</b>	Filename used to save Profile on the computer's hard disk
	<b>Data type</b>	STRING
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>InputScale</b>	<b>Description</b>	The input amplitude or x-axis multiplier applied to the CAM Profile
	<b>Data type</b>	LREAL
	<b>Range</b>	Positive
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>OutputScale</b>	<b>Description</b>	The output amplitude or y-axis multiplier applied to the CAM Profile
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>InputOffset</b>	<b>Description</b>	The input offset or x-axis shift applied to the CAM Profile.
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>OutputOffset</b>	<b>Description</b>	The output offset or y-axis shift applied to the CAM Profile
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 2.3.2.28.4.2 Output

<b>Default (.Q)</b>	<b>Description</b>	Returns TRUE if a new CAM Profile is initialized
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

#### 2.3.2.29.5.3 Return Type

BOOL

#### 2.3.2.30.6 Related Functions

["MLProfileCreate" \(p. 561\)](#)

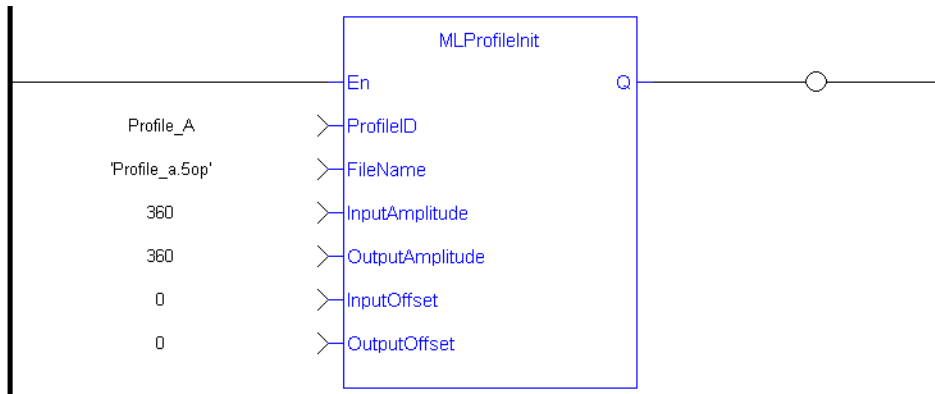
["MLCamInit" \(p. 210\)](#)

### 2.3.2.31.7 Example

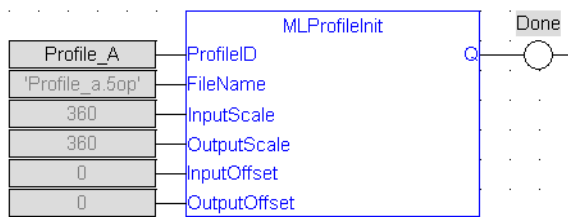
#### 2.3.2.32.8.1 Structured Text

```
//Initialize a previously created CAM Profile
MLProfileCreate( Profile_A , 'Profile_A.5op' , 360, 360, 0, 0
);
```

#### 2.3.2.33.9.2 Ladder Diagram



#### 2.3.2.34.10.3 Function Block Diagram



**2.3.2.35 MLProfileRelease** PLCopen ✓ Pipe Network ✓

An application program is limited to 256 Profile ID's. This FB releases an existing profile ID definition so that the profile ID can be used for a different/new Profile (minimizing the risk of reaching 256 Profile ID's). Once the existing Profile ID definition has been successfully released, the Profile ID can then be used by either "MLProfileInit" (p. 563) or "MLProfileBuild" (p. 552) to create a new Profile.

The Profile ID selected by the input parameter must not be in-use by a motion engine. In-use is defined as:

- For Pipe Network – it must not be currently selected for use by an active CAM block in an active pipe. Pipe has been activated by "MLCamSwitch" (p. 212).
- For PLCOpen – selected for use by "MC\_CamIn" (p. 479) and has an active move.

There are a number of ways to change an in-use profile to one that is not in-use (deactivated):

- For Pipe Network – Perform a "MLCamSwitch" (p. 212) on an active Pipe to a different Profile or deactivate the pipe.
- For PLCOpen – whenever the active profile move is halted or aborted, the profile is no longer in use. "MC\_CamOut" (p. 486) is one way of aborting the profile move. Actually, any PLCOpen motion command that aborts a profile move will also deactivate a profile.

**NOTE**

Any profile ID created by [MC\\_CamTblSelect](#) from the specified ProfileID will be destroyed and need to be recreated upon completion of this FB. This means that all derived profile ID's created by MC\_CamTblSelect FB must also not be in use by the PLCOpen motion engine in order for this function to succeed.

**TIP**

Loading a Profile Editor-generated profile into a ProfileID released by MLProfileRelease should be done with care. The MLProfileInit () function call can take in excess of 4 milliseconds to execute. Application execution is suspended during this time until the function call is completed.

**2.3.2.36.1 Arguments**

For more information on how Arguments work, refer to [PLCopen function blocks - General rules](#) .

**2.3.2.37.2.1 Input**

<b>Enable</b>	<b>Description</b>	Enable execution of the function block. Successful completion will result in a profile ID that is no longer assigned to a specific profile and can be reused for a different/new Profile. Prior to reusing this Profile ID it will need to be re-initialized by either an MLProfileInit Function call or by calling MLProfileBuild.
	<b>Data Type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	
	<b>Default</b>	0

<b>ProfileID</b>	<b>Description</b>	Specify a Profile ID that has been created by MLProfileCreate. This is the profile ID that will be released so it can be reused for different/new Profiles. This Profile ID must not be in use by a motion engine.
	<b>Data Type</b>	DINT
	<b>Range</b>	1 to 256
	<b>Unit</b>	
	<b>Default</b>	0

### 2.3.2.38.3.2 Output

<b>Done</b>	<b>Description</b>	If high, Successful completion. The Profile can now be reused.
	<b>Data Type</b>	BOOL
	<b>Range</b>	0, 1
<b>Err</b>	<b>Description</b>	If high, the Function Block did not complete successfully. Reason is given in Error ID.
	<b>Data Type</b>	BOOL
	<b>Range</b>	0, 1
<b>ErrorID</b>	<b>Description</b>	Indicates the reason for the failure. See " <a href="#">Error Codes</a> " (p. 567) table for possible reasons.
	<b>Data Type</b>	INT
	<b>Range</b>	

### 2.3.2.39.4 Error Codes

ErrorID	Description
106	Invalid profile ID. Profile ID: <ol style="list-style-type: none"> <li>1. does not exist</li> <li>2. has not been created yet</li> <li>3. profile ID is not a profile</li> </ol>
108	Profile cannot be released because it is in use by the motion engine or currently selected by an active CAM block.

### 2.3.2.40.5 Related Functions

["MLProfileCreate"](#) (p. 561)

["MLProfileInit"](#) (p. 563)

["MLProfileBuild"](#) (p. 552)

["MLCamInit"](#) (p. 210)

["MC\\_CamTblSelect"](#) (p. 497)

["MC\\_CamIn"](#) (p. 479)

["MC\\_CamOut"](#) (p. 486)

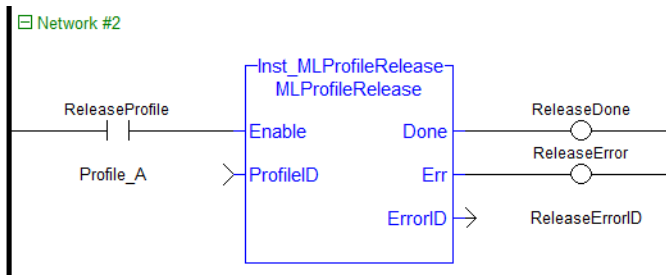
### 2.3.2.41.6 Example

#### 2.3.2.42.7.1 Structured Text

```
//Release a Cam Profile
Inst_MLProfileRelease( Profile_A , 'Profile_A.5op');

If Inst_MLProfileRelease.Done THEN
    // Do Something
ELSIF Inst_MLProfileRelease.Err THEN
    // Handle Error
END_IF;
```

#### 2.3.2.43.8.2 Ladder Diagram



#### 2.3.2.44.9.3 Function Block Diagram





### 2.3.3 Motion Library

Name	Description	Return type
"MLMotionInit" (p. 571)	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" (p. 574)	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" (p. 576)	Returns the status of the motion engine 0: Not initialized 1: Running 2: Stopped 3: Error	None
"MLMotionStop" (p. 578)	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" (p. 579)	Prints the system time to the log	BOOL
"MLMotionCycleTime" (p. 569)	Returns the Motion Base Cycle time in seconds.	

#### 2.3.3.1 State Machine

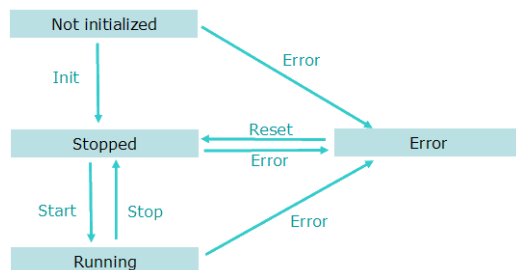


Figure 1-92: Motion State Machine

The Motion State Machine is driven by the IEC 61131-3 application with the help of the "Motion Library" (p. 569) function blocks.

Each arrow represents a transition from one State to another one.

#### 2.3.3.2 MLMotionCycleTime PLCopen ✓ Pipe Network ✓

Returns the Motion Base Cycle time in seconds.

##### 2.3.3.3.1 Arguments

###### 2.3.3.4.2.1 Input

Enable	Description
	Data type
	Range
	Unit
	Default

### 2.3.3.5.3.2 Output

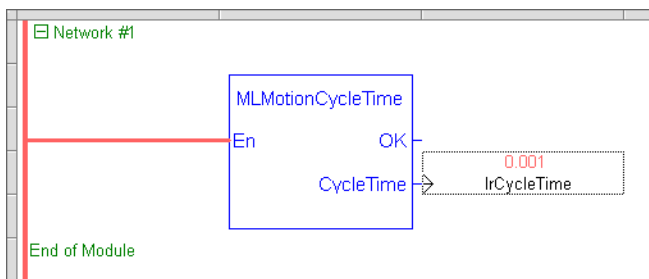
<b>OK</b>	<b>Description</b>
	<b>Data type</b>
<b>CycleTime</b>	<b>Description</b> Cycle time in seconds
	<b>Data type</b>
	<b>Unit</b>

### 2.3.3.6.4 Example

#### 2.3.3.7.5.1 Structured Text

```
//Read EtherCAT cycle rate in ms
lrCycleTime:= MLMotionCycleTime();
```

### 2.3.3.8.6.2 Ladder Diagram



### 2.3.3.9.7.3 Function Block Diagram



**2.3.3.10 MLMotionInit**  **2.3.3.11.1 Description**

Initializes the motion library. Must be called before any other Motion Library function. Returns TRUE if the function succeeded.

**NOTE**

The BasePeriod argument establishes the base cycle time (in microseconds) for the Motion Engine when running simulations without the EtherCAT Motion Bus. When the EtherCAT Motion Bus is present, the EtherCAT cycle time overrides the BasePeriod argument (the cycle time is defined in the [Master](#) tab). The EtherCAT cycle time then becomes the base cycle time for the Motion Engine.

**2.3.3.12.2 Parameter**

**BasePeriod** : LREAL (input)

**2.3.3.13.3 Return Type**

BOOL

**2.3.3.14.4 Example****2.3.3.15.5.1 ST**

```
//Initialization code to start EtherCAT network.
//First initialize network with MLMotionInit command
//Then wait for command to finish by monitoring MLMotionStatus
output
//Once initialized, create any cam profiles and PLCopen or
Pipenetwork devices
//Then call MLMotionStart and monitor MLMotionStatus again
before beginning rest of program
FirstCycle := TRUE;

On FirstCycle DO //Initialize the motion engine
  MLMotionInit( 1000);
END_DO;

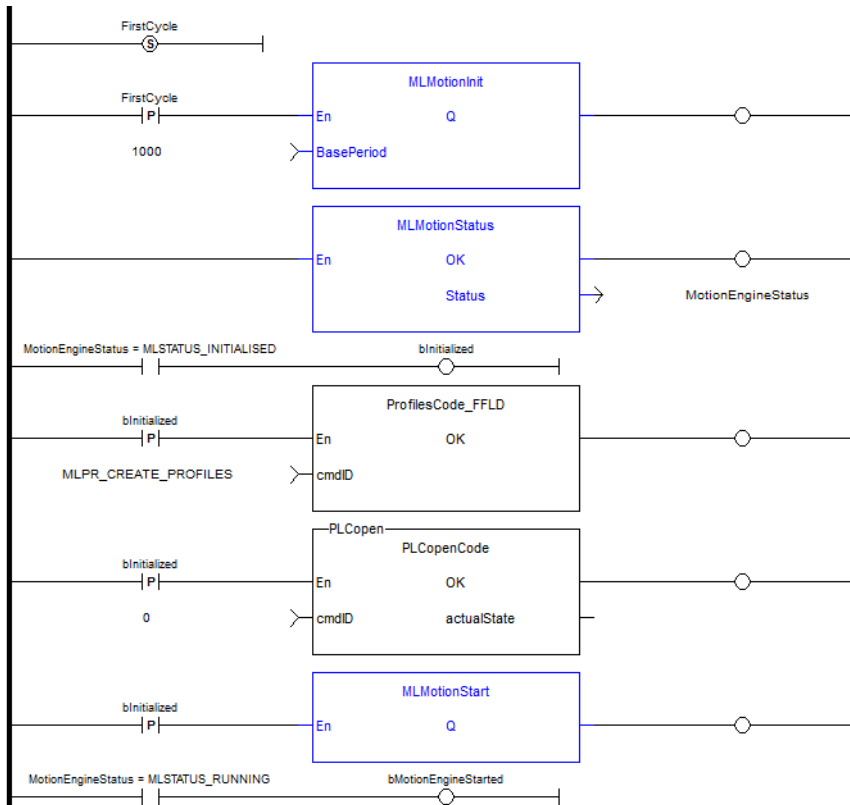
MotionEngineStatus := MLMotionStatus();//Check the current
status of the motion engine
//Once motion engine is initialized, create CAM profiles and
defined Axis, then start the motion engine
ON MotionEngineStatus = MLSTATUS_INITIALISED DO
  Profiles( MLPR_CREATE_PROFILES );
  PLCopen( 0 );
  MLMotionStart();
END_DO;

IF MotionEngineStatus = MLSTATUS_RUNNING THEN
  bMotionEngineStarted := TRUE;
ELSE
  bMotionEngineStarted := FALSE;
END_IF;
```

**2.3.3.16.6.2 FBD**



### 2.3.3.17.7.3 FFLD



**2.3.3.18 MLMotionRstErr** PLCopen ✓ Pipe Network ✓

**2.3.3.19.1 Description**

Clears motion engine errors, motion bus driver errors, and EtherCAT network errors. MLMotionRstErr will return the motion engine status to the Stopped state. Returns TRUE if the function succeeded.

See also: "[MLMotionStatus](#)" (p. 576), "[MLMotionStop](#)" (p. 578), "[MLMotionStart](#)" (p. 574)

**2.3.3.20.2 Return Type**

BOOL

**2.3.3.21.3 Example**

**2.3.3.22.4.1 ST**

```

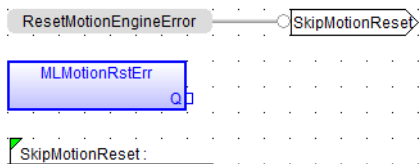
//Reset and restart motion engine
//Done to restart ethercat after controller error such as
//E30 or E33 that stops network communication
//First have to reset error, then start network again

ON ResetMotionEngineError DO
  MLMotionRstErr();
END_DO;

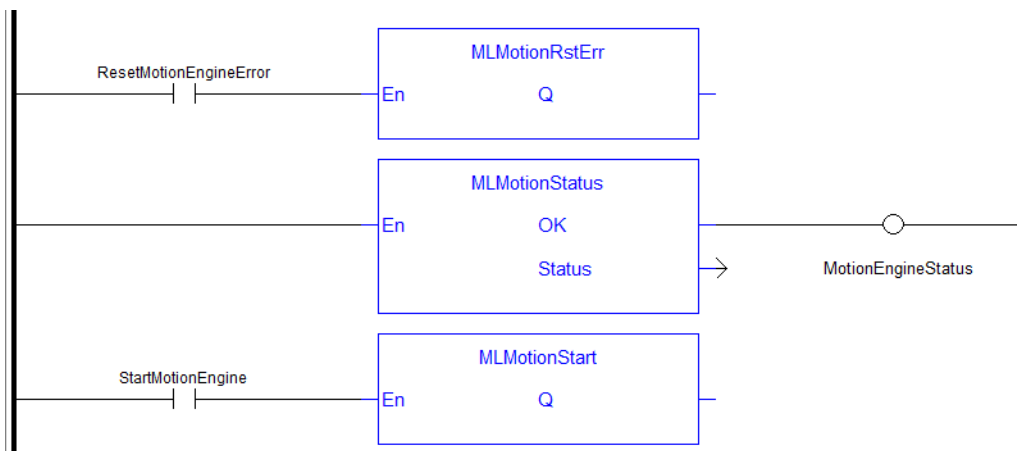
MotionEngineStatus:= MLMotionStatus();

ON StartMotionEngine DO
  MLMotionStart();
END_DO;
```

**2.3.3.23.5.2 FBD**



**2.3.3.24.6.3 FFLD**



**2.3.3.25 MLMotionStart**  **2.3.3.26.1 Description**

Starts the motion engine, motion bus driver, clears the EtherCAT diagnostic registers of all nodes, and initializes EtherCAT network to operational mode. Applicable to PLCopen and PipesNetwork motion engines. MLMotionStart does not clear any pre-existing error conditions. Returns TRUE if the function succeeded. If motion engine is in the Error state, MLMotionStart will return FALSE.

See also: "[MLMotionStop](#)" (p. 578), "[MLMotionRstErr](#)" (p. 573), "[MLMotionStatus](#)" (p. 576)

**2.3.3.27.2 Return Type**

BOOL

**2.3.3.28.3 Example****2.3.3.29.4.1 ST**

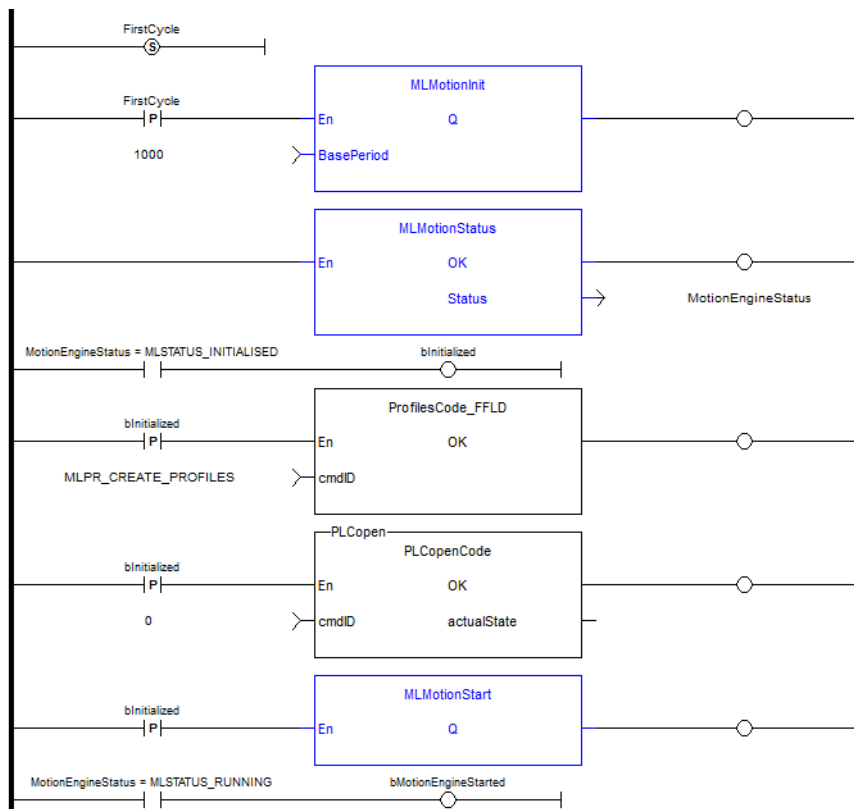
```
//Initialization code to start EtherCAT network.
//First initialize network with MLMotionInit command
//Then wait for command to finish by monitoring MLMotionStatus
output
//Once initialized, create any cam profiles and PLCopen or
Pipenetwork devices
//Then call MLMotionStart and monitor MLMotionStatus again
before beginning rest of program
FirstCycle := TRUE;

On FirstCycle DO //Initialize the motion engine
    MLMotionInit( 1000);
END_DO;

MotionEngineStatus := MLMotionStatus();//Check the current
status of the motion engine
//Once motion engine is initialized, create CAM profiles and
defined Axis, then start the motion engine
ON MotionEngineStatus = MLSTATUS_INITIALISED DO
    Profiles( MLPR_CREATE_PROFILES );
    PLCopen( 0 );
    MLMotionStart();
END_DO;

IF MotionEngineStatus = MLSTATUS_RUNNING THEN
    bMotionEngineStarted := TRUE;
ELSE
    bMotionEngineStarted := FALSE;
END_IF;
```

**2.3.3.30.5.2 FBD****2.3.3.31.6.3 FFLD**



**2.3.3.32 MLMotionStatus**  **2.3.3.33.1 Description**

Returns the status of the motion engine. Based on the [Internal Defines](#), the status will be one of the following.

```
#define MLSTATUS_NOT_INITIALISED 0 (*Motion not initialised*)
#define MLSTATUS_RUNNING 1 (*Motion is running*)
#define MLSTATUS_STOPPED 2 (*Motion is stopped*)
#define MLSTATUS_ERROR 3 (*Motion is in error*)
```

**2.3.3.34.2 Parameter**

**Status : DINT (output)**

**2.3.3.35.3 Return Type**

None

**2.3.3.36.4 Example****2.3.3.37.5.1 ST**

```
//Initialization code to start EtherCAT network.
//First initialize network with MLMotionInit command
//Then wait for command to finish by monitoring MLMotionStatus
output
//Once initialized, create any cam profiles and PLCopen or
Pipenetwork devices
//Then call MLMotionStart and monitor MLMotionStatus again
before beginning rest of program
FirstCycle := TRUE;

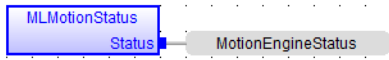
On FirstCycle DO //Initialize the motion engine
MLMotionInit( 1000);
END_DO;

MotionEngineStatus := MLMotionStatus();//Check the current
status of the motion engine
//Once motion engine is initialized, create CAM profiles and
defined Axis, then start the motion engine
ON MotionEngineStatus = MLSTATUS_INITIALISED DO
Profiles( MLPR_CREATE_PROFILES );
PLCopen( 0 );
MLMotionStart();
END_DO;

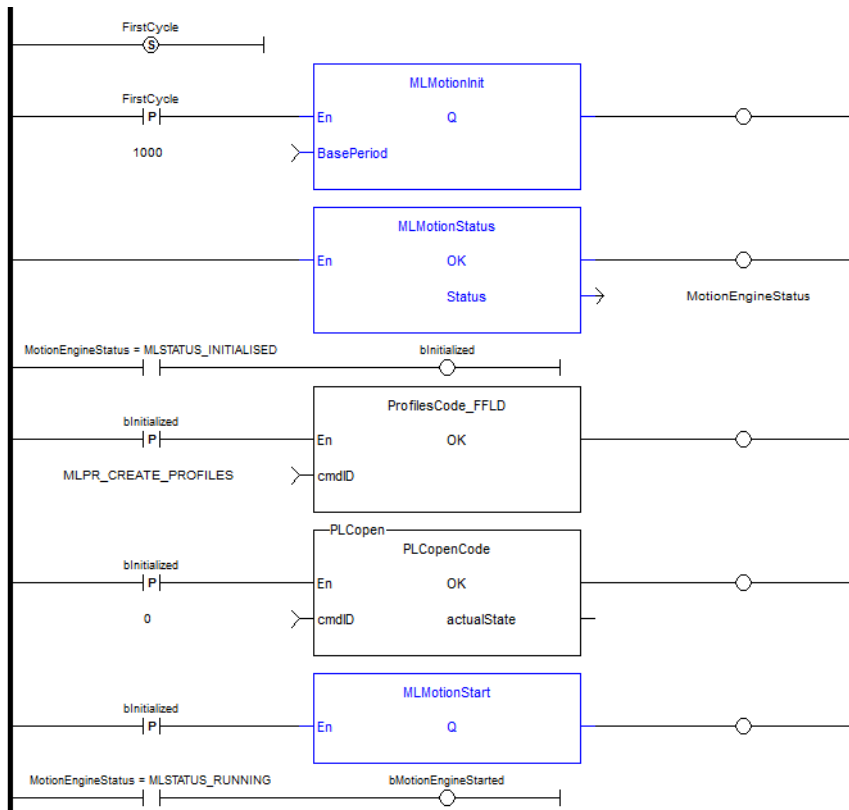
IF MotionEngineStatus = MLSTATUS_RUNNING THEN
bMotionEngineStarted := TRUE;
ELSE
bMotionEngineStarted := FALSE;
END_IF;
```

**2.3.3.38.6.2 FBD**





### 2.3.3.39.7.3 FFLD



**2.3.3.40 MLMotionStop** PLCopen ✓ Pipe Network ✓

**2.3.3.41.1 Description**

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.

See also: "[MLMotionStart](#)" (p. 574), "[MLMotionRstErr](#)" (p. 573), "[MLMotionStatus](#)" (p. 576)

**2.3.3.42.2 Return Type**

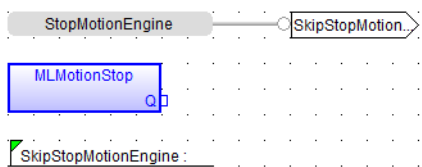
BOOL

**2.3.3.43.3 Example**

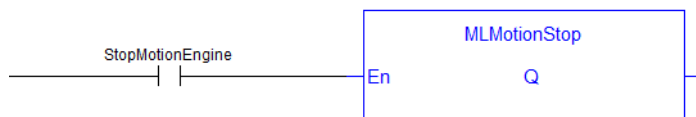
**2.3.3.44.4.1 ST**

```
//Stop the EtherCAT network
ON StopMotionEngine DO
    MLMotionStop();
END_DO;
```

**2.3.3.45.5.2 FBD**



**2.3.3.46.6.3 FFLD**



**2.3.3.47 MLMotionSysTime** PLCopen ✓ Pipe Network ✓**2.3.3.48.1 Description**

Prints the system time to the log. Returns always TRUE.

**2.3.3.49.2 Return Type**

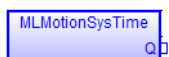
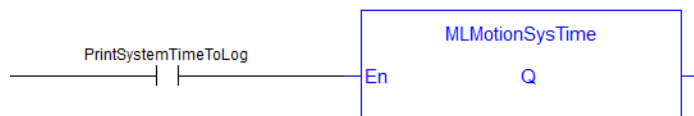
BOOL

**2.3.3.50.3 Units**

milliseconds

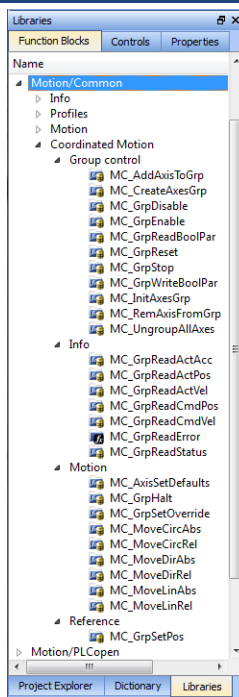
**2.3.3.51.4 Example****2.3.3.52.5.1 ST**

```
//Write the current system time to controller log message
ON PrintSystemTimeToLog DO
  MLMotionSysTime();
END_DO;
```

**2.3.3.53.6.2 FBD****2.3.3.54.7.3 FFLD**

### 2.3.4 Coordinated Motion Function Blocks

This section contains a table with an alphabetical list of the Coordinated Motion function blocks. The table includes where the function block can be found in the KAS IDE library, starting from *Motion/Common* > *Coordinated Motion* > (*Grouping*).

Name	Grouping	Description	Location in the Dictionary
MC_AddAxisToGrp	Group Control	Adds an axis to an axes group.	
MC_AxisSetDefaults	Motion	Sets the default kinematic parameters for an axis.	
MC_CreateAxesGrp	Group Control	Create an axis group for coordinated motion.	
MC_ErrorDescription	Motion/Common > Info	Converts the PLCopen error IDs into message strings	
MC_GrpReset	Group Control	Resets all the axes in an axes group.	
MC_GrpDisable	Group Control	Changes the state of a group to GroupDisabled.	
MC_GrpEnable	Group Control	Changes the state of a group from GroupDisabled to GroupStandby.	
MC_GrpHalt	Motion	Performs a controlled motion stop of all the axes in the group	
MC_GrpReadActAcc	Info	Reads the actual acceleration of the group and the axes in the group.	
MC_GrpReadActPos	Info	Reads the actual position of the axes in the group.	
MC_GrpReadActVel	Info	Reads the actual velocity of the group and the axes in the group.	
MC_GrpReadBoolPar	Group Control	Reads a value from the specified boolean group parameter	
MC_GrpReadParam	Group Control	Reads a value from the specified group parameter.	
MC_GrpReadCmdPos	Info	Reads the command position of the axes in the group.	
MC_GrpReadCmdVel	Info	Reads the command velocity of the axes in the group and the path velocity.	
MC_GrpReadError	Info	Reads the Group ErrorID in State ERRORSTOP.	
MC_GrpReadStatus	Info	Returns the status of an axes group.	

Name	Grouping	Description	Location in the Dictionary
MC_GrpReset	Group Control	Makes the transition from the state GroupErrorStop to GroupStandby by resetting all internal group-related errors. Also resets axis errors and drive faults for each axis in the group.	
MC_GrpSetOverride	Motion	Sets the velocity factor that is multiplied to the commanded velocity of all axes in the group.	
MC_GrpSetPos	Reference	Sets the axis position for all of the axes in an axes group to the positions specified in the Position input.	
MC_GrpStop	Group Control	Performs a non-aborted, controlled motion stop on all axes in an AxesGroup.	
MC_GrpWriteBoolPar	Group Control	Writes a value to the specified boolean group parameter.	
MC_GrpWriteParam	Group Control	Writes a value to the specified group parameter.	
MC_InitAxesGrp	Group Control	Initializes the kinematic limits for the axis group.	
MC_MoveCircAbs	Motion	Commands interpolated circular movement on an axes group to the specified absolute positions.	
MC_MoveCircRel	Motion	Commands interpolated circular movement on an axes group to the specified relative positions.	
MC_MoveDirAbs	Motion	Commands movement of an axes group to an absolute position regardless of path.	
MC_MoveDirRel	Motion	Commands movement of an axes group to a relative position regardless of path.	
MC_MoveLinAbs	Motion	Commands interpolated linear movement on an axes group to the specified absolute positions.	
MC_MoveLinRel	Motion	Commands interpolated linear movement on an axes group to the specified relative positions.	
MC_RemAxisFromGrp	Group Control	Removes an individual axis from an axis group.	
MC_UngroupAllAxes	Group Control	Removes all axes from an axes group.	

#### 2.3.4.1 Coordinated Motion Group Control Library

Function	Description
<a href="#">"MC_AddAxisToGrp" (p. 582)</a>	Adds an axis to an axes group.
<a href="#">"MC_CreateAxesGrp" (p. 585)</a>	Create an axis group for coordinated motion.

Function	Description
"MC_GrpDisable" (p. 588)	Changes the state of a group to GroupDisabled.
"MC_GrpEnable" (p. 590)	Changes the state of a group from GroupDisabled to GroupStandby.
"MC_GrpReadBoolPar" (p. 592)	Reads a value from the specified Boolean group parameter
"MC_GrpReadParam" (p. 595)	Reads a value from the specified group parameter.
"MC_GrpReset" (p. 597)	Makes the transition from the state GroupErrorStop to GroupStandby by resetting all internal group-related errors. Also resets axis errors and drive faults for each axis in the group.
"MC_GrpStop" (p. 599)	Performs a non-aborted, controlled motion stop on all axes in an AxesGroup.
"MC_GrpWriteBoolPar" (p. 602)	Writes a value to the specified Boolean group parameter.
"MC_GrpWriteParam" (p. 605)	Writes a value to the specified group parameter.
"MC_InitAxesGrp" (p. 608)	Initializes the kinematic limits for the axis group.
"MC_RemAxisFromGrp" (p. 611)	Removes an individual axis from an axis group.
"MC_UngroupAllAxes" (p. 614)	Removes all axes from an axes group.

**2.3.4.2.1 MC\_AddAxisToGrp** PLCopen ✓ Pipe Network ✓

**2.3.4.3.2.1 Description**

This function block adds an axis to an axes group. Both the axis and the axes group must be created prior to calling this function block. See "MC\_CreateAxesGrp" (p. 585) and [Create PLCopen Axis](#).

The IdentInGroup input specifies the index of the axis in the group. Axes do not need to be added in sequential order and gaps are acceptable. Gaps are ignored when the group is used.

The group must be in either the "GroupStandby" or "GroupDisabled" state when the axis is added. The state of the group can be read with "MC\_GrpReadStatus" (p. 634). This implies that the group cannot be moving when the axis is added.

This function block does not cause motion.

**TIP**

- An axes group cannot contain more than one instance of an axis.
- Two active groups cannot contain the same axis. An "active" group is one in any state other than GroupDisabled.

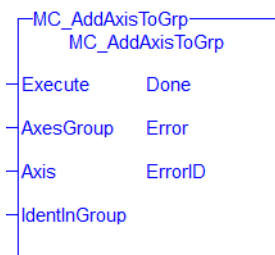


Figure 1-93: MC\_AddAxisToGrp

**2.3.5 Related Functions**

"MC\_CreateAxesGrp" (p. 585), "MC\_GrpReadStatus" (p. 634), "MC\_RemAxisFromGrp" (p. 611), "MC\_UngroupAllAxes" (p. 614), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

**2.3.5.1.1.1 Arguments**

### 2.3.6 Input

<b>Execute</b>	<b>Description</b>	On the rising edge the axis is added to the group.
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	Reference to an axes group
	<b>Data type</b>	AXES_GROUP_REF
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Reference to the axis to be added. An axes group cannot contain more than one instance of an axis.
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>IdentInGroup</b>	<b>Description</b>	<p>The zero-based index of the axis in the group.</p> <ul style="list-style-type: none"> <li>The axis slot in the group cannot be occupied by another axis.</li> <li>The index must be less than the maximum number of axes the group can contain. <code>MaxNumberOfAxes</code> is a property of the axes group and is set when the group is created.</li> </ul> <p>To remove an axis from a group see "<a href="#">MC_RemAxisFromGrp</a>" (p. 611).</p>
	<b>Data type</b>	UINT
	<b>Range</b>	[0, MaxNumberOfAxes - 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 2.3.7 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data type</b>	BOOL

<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error identifier if Error output is set to True. See the table in <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Data type</b>	INT

### 2.3.7.1.1.1 Example

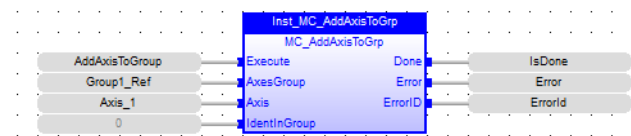
## 2.3.8 Structured Text

```
(*MC_AddAxisToGrp ST example *)
Inst_MC_AddAxisToGrp (AddAxisToGrp, Group1_ref, Axis_1, 0);
```

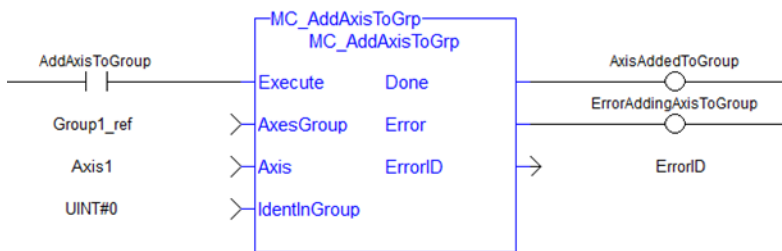
## 2.3.9 IL

```
BEGIN_IL
    CAL Inst_MC_AddAxisToGrp( AddAxisToGrp, Group1_ref, Axis_1,
0 )
END_IL
```

## 2.3.10 FBD



## 2.3.11 Ladder Diagram





### 2.3.11.1.1 MC\_CreateAxesGrp

#### 2.3.11.2.2.1 Description

MC\_CreateAxesGrp creates an axes group for coordinated motion. More than one axes group may be created and be active at the same time but each axis can only be a part of one group at a time.

#### Example of a valid setup:

```

AxesGroup1: Axis0, Axis1, Axis2
AxesGroup2: Axis3, Axis4

```

#### Example of an invalid setup:

```

AxesGroup1: Axis0, Axis1, Axis2
AxesGroup2: Axis2, Axis3, Axis4

```

The invalid setup is not allowed because Axis2 would be a part of two axes groups at the same time.

If an axis needs to be in more than one group, it can be removed from one and then added to another group. This is done using "MC\_RemAxisFromGrp" (p. 611) and "MC\_AddAxisToGrp" (p. 582).

#### IMPORTANT

MC\_CreateAxesGrp must be called between "MLMotionInit" (p. 571) and "MLMotionStart" (p. 574).

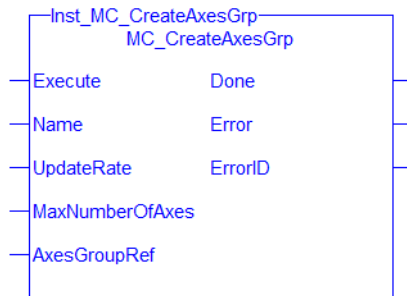


Figure 1-94: MC\_CreateAxesGrp

### 2.3.12 Related Function Blocks

"MC\_InitAxesGrp" (p. 608), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

#### 2.3.12.1.1.1 Arguments

For more detail on how inputs and outputs work, refer to [PLCopen Function Blocks - General Rules](#).

#### 2.3.12.2.2.2 Input

<b>Execute</b>	<b>Description</b>	On the rising edge, this function block will create a coordinated motion axes group
	<b>Data Type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Name</b>	<b>Description</b>	Axes Group Name

	<b>Data Type</b>	STRING
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>UpdateRate</b>	<b>Description</b>	Update rate of the axes group. The group update rate will be the same as the <b>Base Period</b> specified in "MLMotionInit" (p. 571). The update rate will run at the Base Period if it is a smaller time than the Base Period.  (0, 1, and 2 are reserved for future enhancements) 3 = 125 µsec 4 = 250 µsec 5 = 500 µsec 6 = 1 msec 7 = 2 msec 8 = 4 msec 9 = 8 msec
	<b>Data Type</b>	UINT
	<b>Range</b>	[3,9]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>MaxNumberOfAxes</b>	<b>Description</b>	The maximum number of axes that can be controlled by the group.
	<b>Data Type</b>	UINT
	<b>Range</b>	[2,256]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroupRef</b>	<b>Description</b>	The axes group reference variable to be initialized with a reference to the new axes group.
	<b>Data Type</b>	AXES_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 2.3.12.3.3.3 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Date Type</b>	BOOL

<b>Error</b>	<b>Description</b>	If True, then an error has occurred.
	<b>Date Type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See the table in <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Date Type</b>	INT

#### 2.3.12.4.4 Example

Calls to this function block are automatically generated when the application is compiled. Users should not manually call this function block.

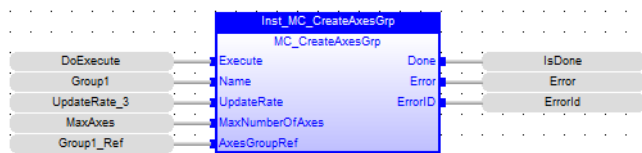
#### 2.3.13 Structured Text

```
Inst_MC_CreateAxesGrp( DoExecute, 'Group1', UpdateRate_3,
MaxAxes, Group1_Ref);
```

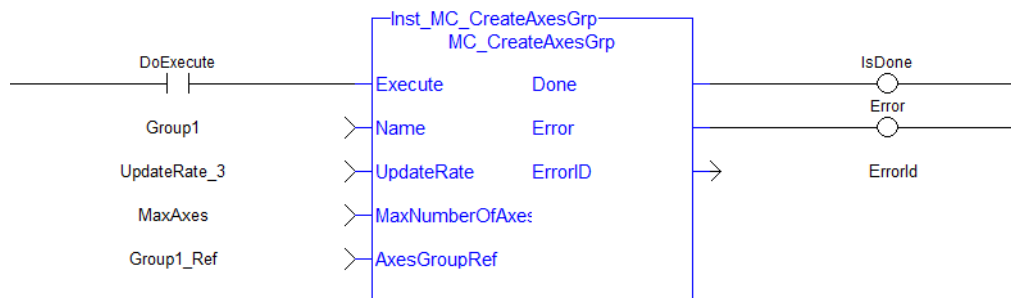
#### 2.3.14 Instruction List

```
BEGIN_IL
    CAL Inst_MC_CreateAxesGrp1(DoExecute, 'Group1', UpdateRate_
3, MaxAxes, Group1_Ref)
END_IL
```

#### 2.3.15 Function Block Diagram



#### 2.3.16 Ladder Diagram



**2.3.16.1.1 MC\_GrpDisable** PLCopen ✓ Pipe Network ✓

**2.3.16.2.2.1 Description**

MC\_GrpDisable changes the state for a group to GroupDisabled. If the group is already in GroupDisabled, then MC\_GrpDisable will do nothing. This function block can be issued in the group states: (GroupDisabled, GroupStandby, or GroupErrorStop).

**NOTE**

MC\_GrpDisable will fail if the group is in any state other than GroupStandby or GroupDisabled.

Refer to [Group State Diagrams](#) for details.

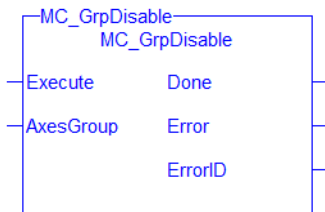


Figure 1-95: MC\_GrpDisable

**2.3.17 Related Functions**

"MC\_GrpEnable" (p. 590), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

**2.3.17.1.1.1 Arguments**

**2.3.18 Input**

<b>Execute</b>	<b>Description</b>	On the rising edge, request to disable the axis group.
	<b>Data Type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	The axis group to be disabled
	<b>Data Type</b>	AXIS_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.3.19 Output**

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
-------------	--------------------	---

	<b>Data Type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, then an error has occurred.
	<b>Data Type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See the table in <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data Type</b>	INT

### 2.3.19.1.1 Example

### 2.3.20 ST

```
(* Inst_MC_GrpDisableST example *)
Inst_MC_GrpDisable( DisableGroup, Group1_Ref );
```

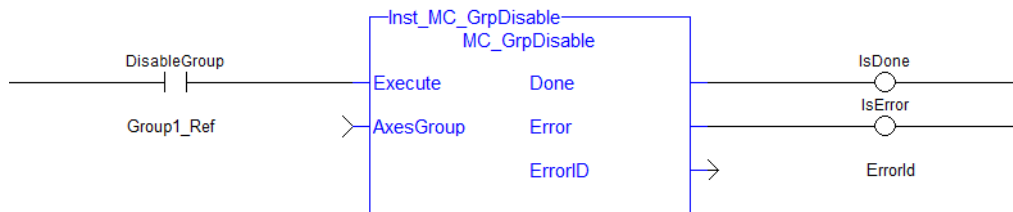
### 2.3.21 IL

```
BEGIN_IL
    CAL Inst_MC_GrpDisable( DisableGroup, Group1_Ref )
END_IL
```

### 2.3.22 FBD



### 2.3.23 FFLD



**2.3.23.1.1 MC\_GrpEnable** PLCopen ✓ Pipe Network ✓

**2.3.23.2.2.1 Description**

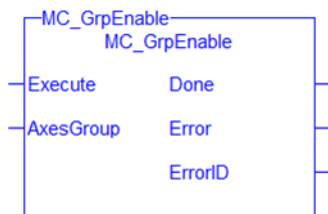
MC\_GrpEnable changes the state of a group from GroupDisabled to GroupStandby. If the group is already in GroupStandby, then MC\_GrpEnable will do nothing.

**NOTE**  
The group must be in GroupStandby in order to perform motion.

MC\_GrpEnable will fail under the following conditions.

- It contains no axes
- The group is not in GroupDisabled or GroupStandby
- One or more axes in the group are in another group that is not in GroupDisabled.

Refer to [Group State Diagrams](#) for more details.



**Figure 1-96:** MC\_GrpEnable

**2.3.24 Related Functions**

"MC\_GrpDisable" (p. 588), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

**2.3.24.1.1.1 Arguments**

**2.3.25 Input**

<b>Execute</b>	<b>Description</b>	On the rising edge, request to enable the axis group
	<b>Data Type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	The axis group to be enabled
	<b>Data Type</b>	AXIS_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.3.26 Output**

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data Type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data Type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See the table in <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data Type</b>	INT

### 2.3.26.1.1.1 Example

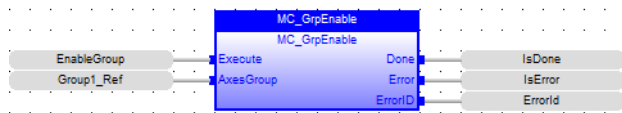
### 2.3.27 Structured Text

```
(* Inst_MC_GrpEnableST example *)
Inst_MC_GrpEnable( EnableGroup, Group1_Ref );
```

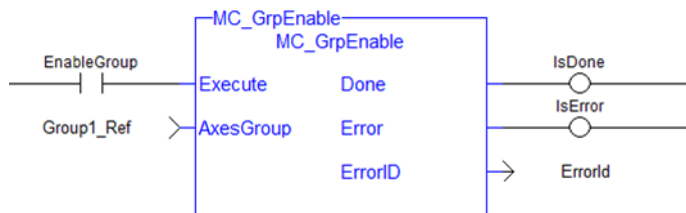
### 2.3.28 IL

```
BEGIN_IL
    CAL Inst_MC_GrpEnable( EnableGroup, Group1_Ref )
END_IL
```

### 2.3.29 FBD



### 2.3.30 FFLD



**2.3.30.1.1 MC\_GrpReadBoolPar** PLCopen ✓ Pipe Network ✓

**2.3.30.2.2.1 Description**

This function block reads a value from the specified Boolean group parameter. See [Recovery of the System State After an Axis Error](#) for more information.

MC\_GrpReadBoolPar(Axesgroup\_Ref GroupID, Uint BoolID) where BoolID can be one of the following 2 currently defined Booleans:

IGNORE\_AXIS\_ESTOP: ID = 1000 : The value read will be either TRUE or False as set by the MC\_GrpWriteBoolPar function block.

AXIS\_ESTOP\_ACTIVE: ID = 1001 : This Read-only parameter will be asserted TRUE whenever an axis in the group is experiencing an Axis Estop Error. When there are no Axis Estop Errors present on the axes in a group, this parameter will be set to FALSE.

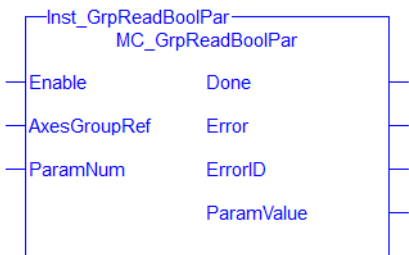


Figure 1-97: MC\_GrpReadBoolPar

**2.3.31 Related Function Blocks**

"MC\_GrpWriteBoolPar" (p. 602), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

**2.3.31.1.1.1 Arguments**

For more details on how inputs and outputs work, refer to [PLCopen Function Blocks - General Rules](#).

**2.3.32 Input**

<b>Enable</b>	<b>Description</b>	If True, then request to read a value from the specified Boolean group parameter.
	<b>Data Type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroupRef</b>	<b>Description</b>	The axis group that the Boolean parameter value will be read from.
	<b>Data Type</b>	AXES_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a



	<b>Default</b>	—
<b>ParamNum</b>	<b>Description</b>	<p>ParamNum can be one of the following two currently defined Booleans:</p> <ul style="list-style-type: none"> <li>• IGNORE_AXIS_ESTOP: ID = 1000 : The value read will be either TRUE or False as set by the MC_GrpWriteBoolPar function block.</li> <li>• AXIS_ESTOP_ACTIVE: ID = 1001 : This Read-only parameter will be asserted TRUE whenever an axis in the group is experiencing an Axis Estop Error. When there are no Axis Estop Errors present on the axes in a group, this parameter will be set to FALSE.</li> </ul>
	<b>Data Type</b>	UINT
	<b>Range</b>	1000, 1001
	<b>Unit</b>	UINT
	<b>Default</b>	—

### 2.3.33 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data Type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data Type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See the table in <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Data Type</b>	INT
<b>ParamValue</b>	<b>Description</b>	True or False
	<b>Data Type</b>	BOOL

#### 2.3.33.1.1 Example

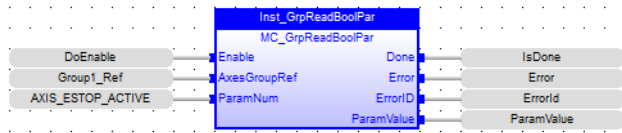
### 2.3.34 ST

```
Inst_GrpReadBoolPar( DoEnable, Group1_Ref, AXIS_ESTOP_ACTIVE );
```

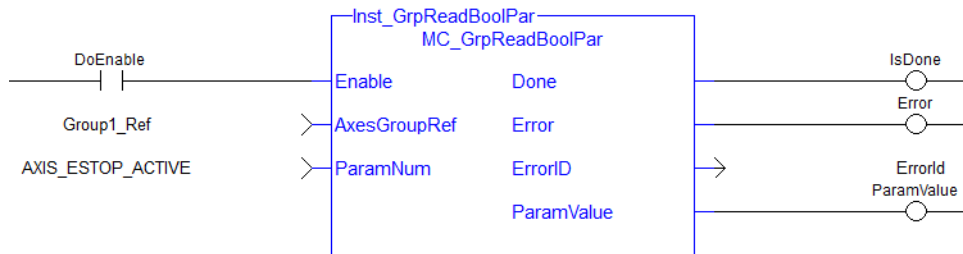
### 2.3.35 IL

```
BEGIN_IL
    Cal Inst_GrpReadBoolPar( DoEnable, Group1_Ref, AXIS_ESTOP_
ACTIVE )
END_IL
```

### 2.3.36 FBD



### 2.3.37 FFLD



### 2.3.37.1.1 MC\_GrpReadParam PLCopen ✓ Pipe Network ✓

This function block reads the value of the specified group parameter.

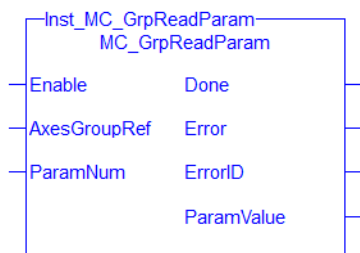


Figure 1-98: MC\_GrpReadParam

#### Related Function Blocks

"MC\_GrpWriteParam" (p. 605), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

#### 2.3.37.2.2.1 Arguments

For more details on how inputs and outputs work, refer to PLCopen Function Blocks - General Rules.

#### 2.3.38 Input

<b>Enable</b>	<b>Description</b>	If True, then request to read a value from the specified group parameter.
	<b>Data Type</b>	BOOL
	<b>Range</b>	0,1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroupRef</b>	<b>Description</b>	The axis group that the parameter value will be read from.
	<b>Data Type</b>	AXIS_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ParamNum</b>	<b>Description</b>	Currently, only one parameter is supported: MC_GRP_PARAM_CIRCLE_TOLERANCE: (ID = 2000): The value read will be the axes group circle construction tolerance. See <a href="#">Precision Requirements for Circular Move Input Parameters</a> for more information.
	<b>Data Type</b>	LREAL
	<b>Range</b>	See <a href="#">Axes Group Parameters</a>
	<b>Unit</b>	LREAL
	<b>Default</b>	—

### 2.3.39 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data Type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data Type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See the table in PLCopen Function Block ErrorID Output.
	<b>Data Type</b>	INT
<b>ParamValue</b>	<b>Description</b>	The value of the group parameter.
	<b>Data Type</b>	LREAL

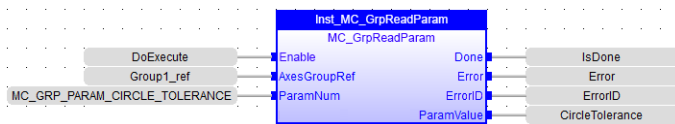
#### 2.3.39.1.1 Examples

### 2.3.40 ST

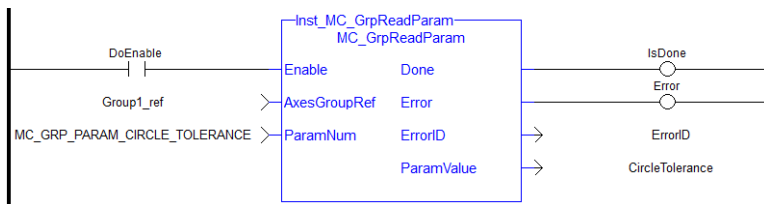
```

Inst_MC_GrpReadParam( DoEnable, Group1_ref, MC_GRP_PARAM_CIRCLE_
TOLERANCE );
CircleTolerance := Inst_MC_GrpReadParam.ParamValue;
    
```

### 2.3.41 FBD



### 2.3.42 FFLD



### 2.3.42.1.1 MC\_GrpReset

#### 2.3.42.2.2.1 Description

This function block makes the transition from the state GroupErrorStop to GroupStandby by resetting all internal group-related errors – it does not affect the output of the FB instances. This function block also resets axis errors and drive faults for each axis in the group. This function block does not cause any motion.

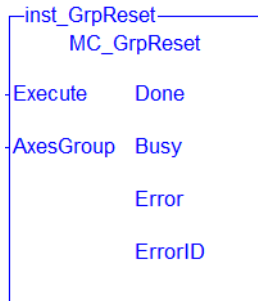


Figure 1-99: MC\_GrpReset

### 2.3.43 Related Functions

"MC\_GrpReadError" (p. 632), "MC\_GrpReadStatus" (p. 634), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

#### 2.3.43.1.1.1 Arguments

For more details on how inputs and outputs work, refer to [PLCopen Function Blocks - General Rules](#).

#### 2.3.44 Input

<b>Execute</b>	<b>Description</b>	On the rising edge, this FB resets group-related errors and all of the axes in the group.
	<b>Data Type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	The axes group in which the axes will be reset.
	<b>Data Type</b>	AXES_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 2.3.45 Output

<b>Done</b>	<b>Description</b>	If True, then the reset completed successfully.
-------------	--------------------	---

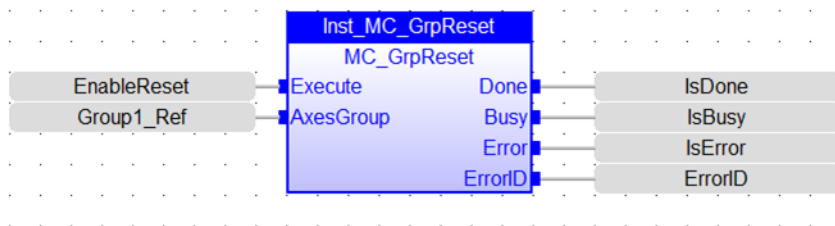
	<b>Data Type</b>	BOOL
<b>Busy</b>	<b>Description</b>	If True, then the FB is executing.
	<b>Data Type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data Type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error identifier if Error output is set to TRUE. See table in <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Data Type</b>	INT

### 2.3.45.1.1.1 Example

### 2.3.46 ST

```
Inst_MC_GrpReset ( EnableReset, Group1_Ref );
```

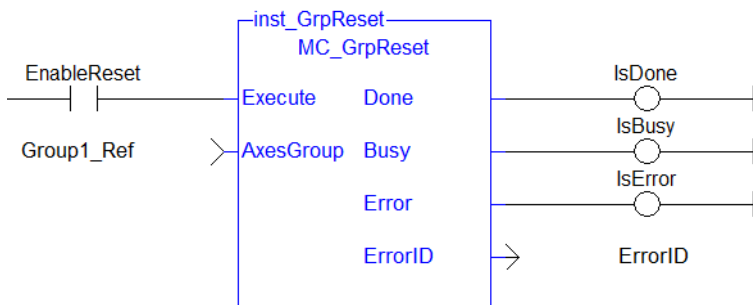
### 2.3.47 FBD



### 2.3.48 IL

```
BEGIN_IL
    CAL Inst_MC_GrpReset ( EnableReset, Group1_Ref )
END_IL
```

### 2.3.49 FFLD



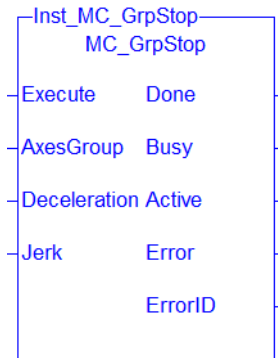
### 2.3.49.1.1 MC\_GrpStop

#### 2.3.49.2.2.1 Description

MC\_GrpStop performs a controlled motion stop of all axes in the group. When the path velocity reaches zero any queued moves are flushed from the buffer and the Done output is set. When both the Done output is true and the application has cleared the Execute input the state transitions to GroupStandby. MC\_GrpStop *can not be aborted*.

#### NOTE

MC\_GrpStop does NOT prevent a single axis from executing nor does it prevent other Coordinated Motion moves from executing once MC\_GrpStop has completed.



### 2.3.50 Related Functions

"MC\_GrpHalt" (p. 642), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

#### 2.3.50.1.1.1 Arguments

For more details on how inputs and outputs work, refer to [PLCopen Function Blocks - General Rules](#).

#### 2.3.51 Input

<b>Execute</b>	<b>Description</b>	On the rising edge the command to stop all of the axes in the group is initiated.
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	The axes group in which the axes will be stopped.
	<b>Data type</b>	AXES_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>Deceleration</b>	<b>Description</b>	The path deceleration rate for all of the axes in the group
	<b>Data type</b>	LREAL
	<b>Range</b>	0 < Deceleration See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>JerK</b>	<b>Description</b>	Not supported
	<b>Data type</b>	LREAL
	<b>Range</b>	0 ≤ Jerk See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 2.3.52 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	TRUE from the moment the EXECUTE input is TRUE until the stop is complete.
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	If True, then the stop is still executing.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error identifier if Error output is set to TRUE. See table in <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Data type</b>	INT

#### 2.3.52.1.1.1 Example

### 2.3.53 Structured Text

```
Inst_MC_GrpStop ( EnableStop, Group1_Ref, Deceleration, Jerk );
```

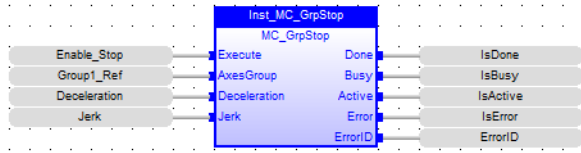
### 2.3.54 IL



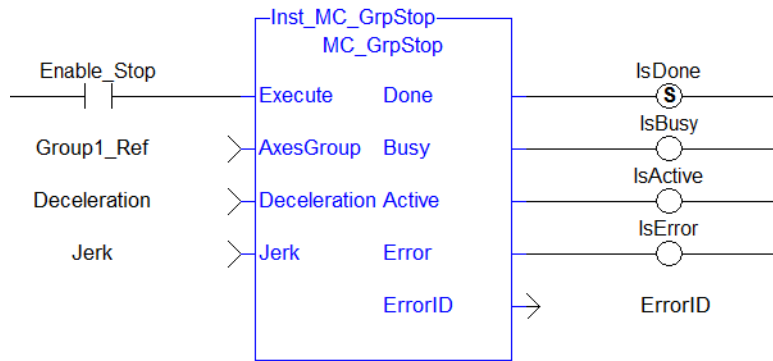
```

BEGIN_IL
  CAL Inst_MC_GrpStop ( EnableStop, Group1_Ref, Deceleration,
  Jerk )
END_IL
    
```

### 2.3.55 FBD



### 2.3.56 FFLD



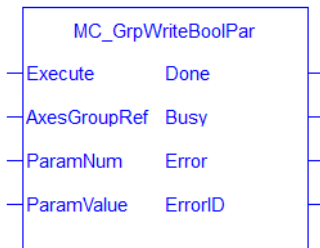
**2.3.56.1.1 MC\_GrpWriteBoolPar** PLCopen ✓ Pipe Network ✓

**2.3.56.2.2.1 Description**

This function block writes a value to the specified Boolean group parameter. See [Recovery of the System State After an Axis Error](#) for more information.

IGNORE\_AXIS\_ESTOP (BoolID = 1000), and the Value can be either TRUE or FALSE.

- Setting this Boolean Parameter to TRUE will result in the Coordinated Motion Engine NOT stopping all axes in a group when one of them is stopped due to an Axis Estop Error. Only the axis experiencing the error will stop when this Parameter is set to TRUE.
- When this parameter is FALSE (Default), all axes in a group will be stopped and the power off request is asserted for each axis.



**Figure 1-100:** MC\_GrpWriteBoolPar

**2.3.57 Related Function Blocks**

"MC\_GrpReadBoolPar" (p. 592), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

**2.3.57.1.1.1 Arguments**

For more details on how inputs and outputs work, refer to [PLCopen Function Blocks - General Rules](#).

**2.3.58 Input**

<b>Execute</b>	<b>Description</b>	On the rising edge, request to write a value to the specified Boolean group parameter.
	<b>Data Type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroupRef</b>	<b>Description</b>	The axis group that the Boolean parameter value will be written to.
	<b>Data Type</b>	AXES_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>ParamNum</b>	<b>Description</b>	The ID number of the Boolean parameter that is to be written IGNORE_AXIS_ESTOP (BoolID = 1000)
	<b>Data Type</b>	UINT
	<b>Range</b>	
	<b>Unit</b>	
	<b>Default</b>	
<b>ParamValue</b>	<b>Description</b>	True or false
	<b>Data Type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 2.3.59 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data Type</b>	BOOL
<b>Busy</b>	<b>Description</b>	If True, then the function block is executing.
	<b>Data Type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data Type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See the table in <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Data Type</b>	INT

#### 2.3.59.1.1.1 Example

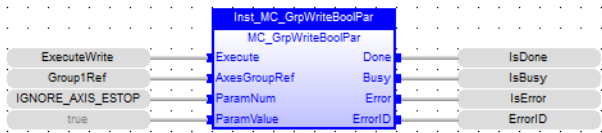
### 2.3.60 ST

```
Inst_MC_GrpWriteBoolPar( ExecuteWrite, Group1Ref, IGNORE_AXIS_
ESTOP, true );
```

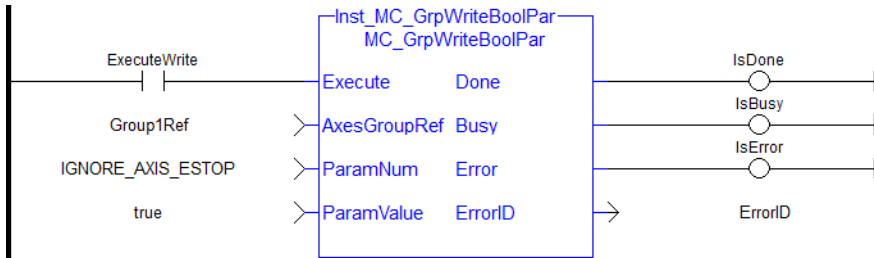
### 2.3.61 IL

```
BEGIN_IL
    Cal Inst_MC_GrpWriteBoolPar( ExecuteWrite, Group1Ref,
IGNORE_AXIS_ESTOP, true )
END_IL
```

### 2.3.62 FBD



### 2.3.63 FFLD



### 2.3.63.1.1 MC\_GrpWriteParam PLCopen ✓ Pipe Network ✓

This function block writes a value to the specified group parameter.

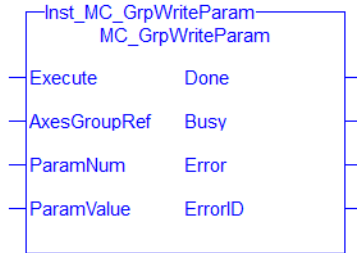


Figure 1-101: MC\_GrpWriteParam

#### Related Function Blocks

"MC\_GrpReadParam" (p. 595), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

#### 2.3.63.2.2.1 Arguments

For more details on how inputs and outputs work, refer to PLCopen Function Blocks - General Rules.

#### 2.3.64 Input

<b>Execute</b>	<b>Description</b>	On the rising edge, request to write a value to the specified group parameter.
	<b>Data Type</b>	BOOL
	<b>Range</b>	0,1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroupRef</b>	<b>Description</b>	The axis group that the parameter value will be written to.
	<b>Data Type</b>	AXIS_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>ParamNum</b>	<b>Description</b>	Currently, only one parameter is supported: MC_GRP_PARAM_CIRCLE_TOLERANCE: (ID = 2000): The value read will be the axes group circle construction tolerance. See <a href="#">Precision Requirements for Circular Move Input Parameters</a> for more information.
	<b>Data Type</b>	LREAL
	<b>Range</b>	See <a href="#">Axes Group Parameters</a>
	<b>Unit</b>	LREAL

	<b>Default</b>	—
<b>ParamValue</b>	<b>Description</b>	The new value for the group parameter
	<b>Data Type</b>	LREAL
	<b>Range</b>	parameter dependent
	<b>Unit</b>	parameter dependent
	<b>Default</b>	—

### 2.3.65 Output

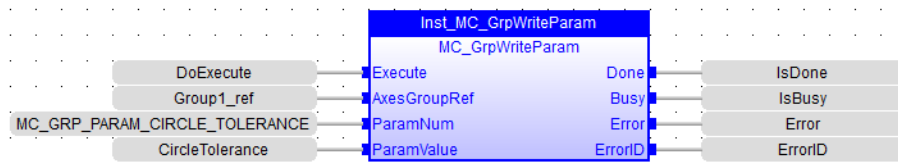
<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data Type</b>	BOOL
<b>Busy</b>	<b>Description</b>	If True, then the function block is executing.
	<b>Data Type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data Type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See the table in PLCOpen Function Block ErrorID Output.
	<b>Data Type</b>	INT

#### 2.3.65.1.1.1 Examples

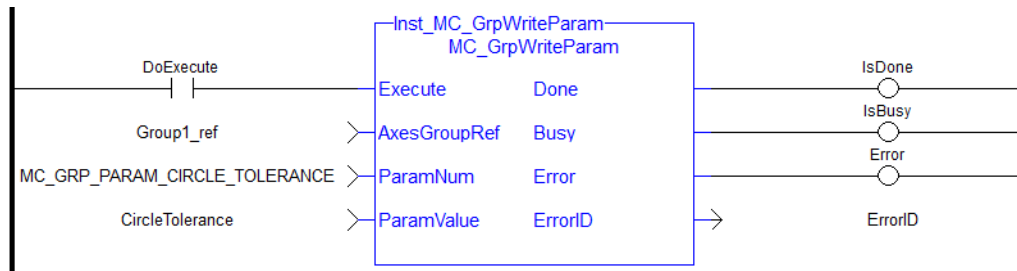
### 2.3.66 ST.

```
Inst_MC_GrpWriteParam( DoExecute, Group1_ref, MC_GRP_PARAM_
CIRCLE_TOLERANCE, CircleTolerance );
```

### 2.3.67 FBD



### 2.3.68 FFLD



**2.3.68.1.1 MC\_InitAxesGrp** PLCopen ✓ Pipe Network ✓

**2.3.68.2.2.1 Description**

MC\_InitAxesGrp initializes the kinematic limits for the axis group. During a move, the motion engine verifies that the limits are not exceeded.

**NOTE**

The function block returns an error if the group state is not GroupStandby or GroupDisabled.

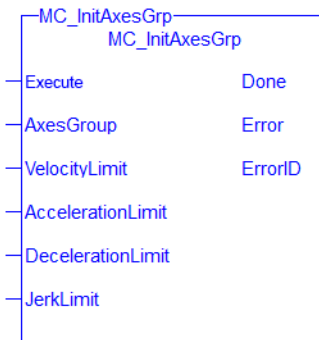


Figure 1-102: MC\_InitAxesGrp

**2.3.69 Related Function Blocks**

"MC\_CreateAxesGrp" (p. 585), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

**2.3.69.1.1.1 Arguments**

**2.3.70 Inputs**

<b>Execute</b>	<b>Description</b>	On the rising edge, this function block will initialize the axis group.
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	The axis group to be initialized
	<b>Data type</b>	AXIS_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>VelocityLimit</b>	<b>Description</b>	Velocity limit
	<b>Data type</b>	LREAL



	<b>Range</b>	$0 < \text{Velocity} < ( 20 * \text{Acceleration} )$ and $0 < \text{Velocity} < ( 20 * \text{Deceleration} )$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	user units per second
	<b>Default</b>	—
<b>AccelerationLimit</b>	<b>Description</b>	Acceleration limit
	<b>Data type</b>	LREAL
	<b>Range</b>	$0 < \text{Velocity} < ( 20 * \text{Acceleration} )$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	user units per second <sup>2</sup>
	<b>Default</b>	—
<b>DecelerationLimit</b>	<b>Description</b>	Deceleration limit
	<b>Data type</b>	LREAL
	<b>Range</b>	$0 < \text{Velocity} < ( 20 * \text{Deceleration} )$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	User units per second <sup>2</sup>
	<b>Default</b>	—
<b>JerkLimit</b>	<b>Description</b>	Jerk limit
	<b>Data type</b>	LREAL
	<b>Range</b>	$( \text{Velocity} / 20 ) < \text{Acceleration} < ( 2 * \text{Jerk} )$ and $( \text{Velocity} / 20 ) < \text{Deceleration} < ( 2 * \text{Jerk} )$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	User units per second <sup>3</sup>
	<b>Default</b>	—

### 2.3.71 Outputs

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, then an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See the table in <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data type</b>	INT

### 2.3.71.1.1.1 Example

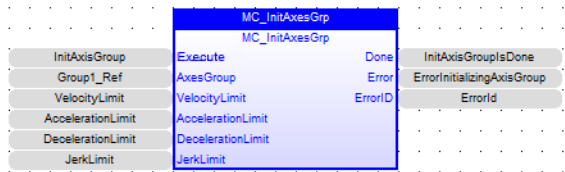
### 2.3.72 Structured Text

```
(* Inst_MC_InitAxesGrpST example *)
Inst_MC_InitAxesGrp( initAxesGrp, grp, velLim, accelLim,
decelLim, jerkLim );
```

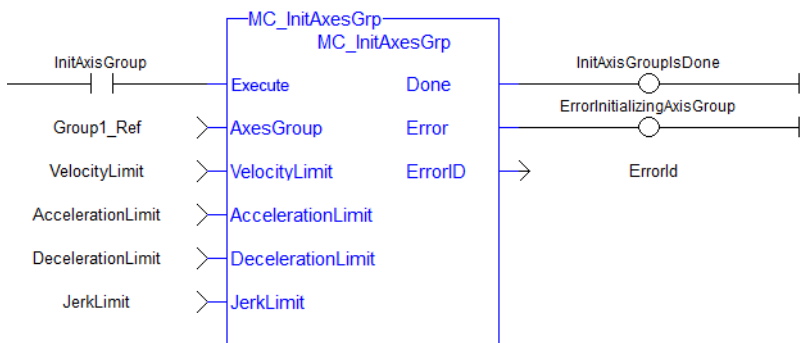
### 2.3.73 IL

```
BEGIN_IL
    CAL Inst_MC_InitAxesGrp( initAxesGrp, grp, velLim,
accelLim, decelLim, jerkLim )
END_IL
```

### 2.3.74 FBD



### 2.3.75 FFLD



### 2.3.75.1.1 MC\_RemAxisFromGrp PLCopen ✓ Pipe Network ✓

#### 2.3.75.2.2.1 Description

MC\_RemAxisFromGrp removes a single axis from a group. This function block can be issued in the group states: (GroupDisabled, GroupStandby, or GroupErrorStop). The group's state will change to GroupDisabled if the axis removed is the last valid axis in the group. This function block does not cause any motion.

#### NOTE

MC\_RemAxisFromGrp will fail if the group is in any state other than GroupStandby or GroupDisabled. Refer to [Group State Diagrams](#) for details.

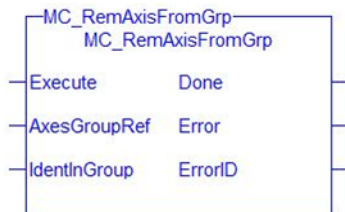


Figure 1-103: MC\_RemAxisFromGrp

#### 2.3.76 Related Functions

"MC\_AddAxisToGrp" (p. 582), "MC\_UngroupAllAxes" (p. 614), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

#### 2.3.76.1.1.1 Arguments

#### 2.3.77 Input

<b>Execute</b>	<b>Description</b>	On the rising edge, request to remove an axis from the group
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroupRef</b>	<b>Description</b>	The axis group from which the axis will be removed
	<b>Data type</b>	AXIS_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>IdentInGroup</b>	<b>Description</b>	The zero-based index of the axis in the group. <ul style="list-style-type: none"> <li>• The axis index in the group must contain a valid axis</li> <li>• The index must be less than the maximum number of axes the group can contain. <code>MaxNumberOfAxes</code> is a property of the axes group and is set when the group is created.</li> </ul>
	<b>Data type</b>	UINT
	<b>Range</b>	[0, MaxNumberOfAxes - 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 2.3.78 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error identifier if Error output is set to True. See the table in <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Data type</b>	INT

#### 2.3.78.1.1.1 Example

### 2.3.79 ST

```
(* Inst_MC_InitAxisGrpST example *)
Inst_MC_RemAxisFromGrp( ExecuteRemAxisFromGrp, Group1_Ref,
AxisId );
```

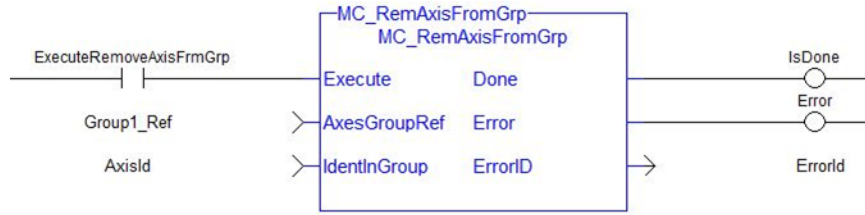
### 2.3.80 IL

```
BEGIN_IL
    CAL Inst_MC_RemAxisFromGrp( ExecuteRemAxisFromGrp, Group1_
Ref, AxisId )
END_IL
```

### 2.3.81 FBD



### 2.3.82 FFLD



**2.3.82.1.1 MC\_UngroupAllAxes** PLCopen ✓ Pipe Network ✓

**2.3.82.2.2.1 Description**

MC\_UngroupAllAxes removes all axes from an axes group. This function block can be issued in the group states: (GroupDisabled, GroupStandby, or GroupErrorStop). The axes group state will be changed to GroupDisabled upon successful completion. This function block does not cause any motion.

**NOTE**

MC\_UngroupAllAxes will fail if the group is in any state other than GroupStandby or GroupDisabled. Refer to [Group State Diagrams](#) for details.

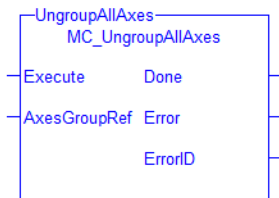


Figure 1-104: MC\_UngroupAllAxes

**2.3.83 Related Functions**

"MC\_AddAxisToGrp" (p. 582), "MC\_RemAxisFromGrp" (p. 611), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

**2.3.83.1.1.1 Arguments**

**2.3.84 Input**

<b>Execute</b>	<b>Description</b>	On the rising edge, request to remove all axes in the axes group
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroupRef</b>	<b>Description</b>	The axis group from which to remove all axes
	<b>Data type</b>	AXIS_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—

**2.3.85 Output**

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data type</b>	BOOL

<b>Error</b>	<b>Description</b>	If True, an error has occurred
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error identifier if 'Error' output is set to TRUE. See the table in <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Data type</b>	INT

### 2.3.85.1.1.1 Examples

### 2.3.86 ST

```
Inst_MC_UngroupAllAxes( ExecuteUngroup, Group1_Ref );
```

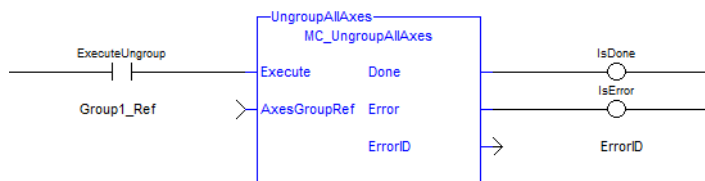
### 2.3.87 IL

```
BEGIN_IL
    CAL Inst_MC_UngroupAllAxes( ExecuteUngroup, Group1_Ref )
END_IL
```

### 2.3.88 FBD



### 2.3.89 FFLD



### 2.3.89.1 Coordinated Motion Info Library

Function	Description
"MC_GrpReadActAcc" (p. 616)	Reads the actual acceleration of the group and the axes in the group.
"MC_GrpReadActPos" (p. 620)	Reads the actual position of the axes in the group.
"MC_GrpReadActVel" (p. 623)	Reads the actual velocity of the group and the axes in the group.
"MC_GrpReadCmdPos" (p. 626)	Reads the command position of the axes in the group.
"MC_GrpReadCmdVel" (p. 629)	Reads the command velocity of the axes in the group and the path velocity.
"MC_GrpReadError" (p. 632)	Reads the Group ErrorID in State ERRORSTOP.
"MC_GrpReadStatus" (p. 634)	Returns the status of an axes group.

#### 2.3.89.2.1 MC\_GrpReadActAcc PLCopen ✓ Pipe Network ✓

##### 2.3.89.3.2.1 Description

The MC\_GrpReadActAcc function block fills the array specified by the 'Acceleration' argument with the actual acceleration of the system in the coordinate system specified by the `CoordSystem` argument. The measured path acceleration is also calculated and reported via the 'PathAcceleration' output. This function block does not cause any motion.

#### NOTE

- The actual acceleration is smoothed over the last 10 samples. This reduces the error in acceleration estimation, but introduces a small amount of phase delay in the reported accelerations.
- Currently, only the ACS coordinate system is supported. See [Coordinate Systems](#) to learn more.

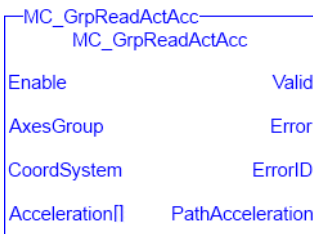


Figure 1-105: MC\_GrpReadActAcc

There is a one-to-one correspondence between the axes in the Axes Group array and the acceleration values in the Acceleration array. Each element in the Acceleration array corresponds to the axis element in the Axes Group array. If an index in the Axes Group is unassigned then the acceleration value for that array element in the Acceleration array will be 0. If the element does contain an axis then the acceleration value will be filled with the current actual acceleration for that axis. Here is an example to illustrate how this works:



### 2.3.90 Related Functions



["MC\\_GrpReadActPos" \(p. 620\)](#), ["MC\\_GrpReadActVel" \(p. 623\)](#), ["MC\\_GrpReadCmdPos" \(p. 626\)](#), ["MC\\_GrpReadCmdVel" \(p. 629\)](#)

See also ["Coordinated Motion"](#), the top-level topic for Coordinated Motion.

### 2.3.90.1.1.1 Arguments

For more detail on how inputs and outputs work, refer to [PLCopen Function Blocks - General Rules](#).

### 2.3.91 Input

<b>Enable</b>	<b>Description</b>	If True, then this function block will read the current actual acceleration of the group and the axes in the group
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	The axes group from which the actual acceleration will be read.
	<b>Data type</b>	AXES_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>CoordSystem</b>	<b>Description</b>	The coordinate system used when reading the actual acceleration
	<b>Data type</b>	SINT
	<b>Range</b>	One of the following enumeration values: <ul style="list-style-type: none"> <li>• MC_COORDINATE_SYSTEM_ACS = 0</li> <li>• MC_COORDINATE_SYSTEM_MCS = 1</li> <li>• MC_COORDINATE_SYSTEM_PCS = 2</li> </ul>
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Acceleration[]</b>	<b>Description</b>	An array where the acceleration data will be written. The length of the array must equal the maximum number of axes allowed in the group. The maximum number of axes is an argument to <a href="#">"MC_CreateAxesGrp" (p. 585)</a> that is used to create axes groups.
	<b>Data type</b>	LREAL
	<b>Range</b>	n/a
	<b>Unit</b>	User units per second <sup>2</sup>

<b>Default</b>	—
----------------	---

### 2.3.92 Output

<b>Valid</b>	<b>Description</b>	If true, the accelerations have been read without error.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If true, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output was set to TRUE. See the table <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data type</b>	INT
<b>PathAcceleration</b>	<b>Description</b>	The current measured path acceleration of the group, measured by taking the square root of the sum of the squared accelerations of each axis.
	<b>Data type</b>	LREAL
	<b>Unit</b>	User units per second <sup>2</sup>

#### 2.3.92.1.1.1 Example

### 2.3.93 Structured Text

```
Inst_MC_GrpReadActAcc( DoRead, Group, CoordSys, AccList );
```

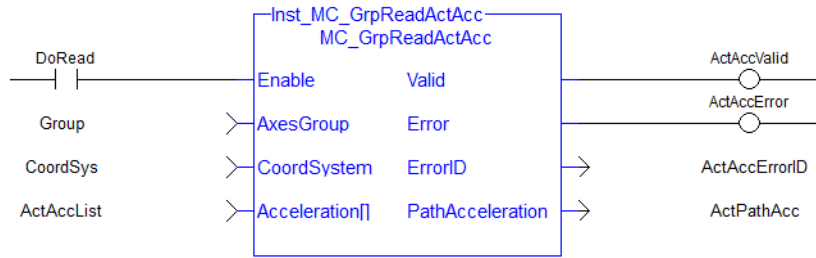
### 2.3.94 IL

```
BEGIN_IL
    CAL Inst_MC_GrpReadActAcc( DoRead, Group, CoordSys, AccList
)
END_IL
```

### 2.3.95 FBD



### 2.3.96 Ladder Diagram



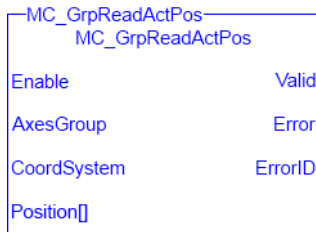
**2.3.96.1.1 MC\_GrpReadActPos** PLCopen ✓ Pipe Network ✓

**2.3.96.2.2.1 Description**

MC\_GrpReadActPos fills the array specified by the 'Position' argument with the actual position of the system in the coordinate system specified by the `CoordSystem` argument. This function block does not cause any motion.

**NOTE**

Currently, only the ACS coordinate system is supported. See [Coordinate Systems](#) to learn more.



**Figure 1-106:** MC\_GrpReadActPos

There is a one to one correspondence between the axes in the Axes Group and the position values in the Position Array. Each element in the Position Array corresponds to the axis element in the Axis Group array. If an index in the Axes Group is unassigned then the position value for that array element in the Position Array will be 0. If the element does contain an axis then the position value will be filled with the current actual position for that axis. Here is an example to illustrate how this works:



**2.3.97 Related Functions**

"MC\_GrpReadActVel" (p. 623), "MC\_GrpReadActAcc" (p. 616), "MC\_GrpReadCmdPos" (p. 626), "MC\_GrpReadCmdVel" (p. 629)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

**2.3.97.1.1.1 Arguments**

For more detail on how inputs and outputs work, refer to [PLCopen Function Blocks - General Rules](#).

**2.3.98 Input**

Enable	Description
	If True, then this function block will read the current actual position of the axes in the group
	<b>Data type</b> BOOL
	<b>Range</b> 0, 1
	<b>Unit</b> n/a
	<b>Default</b> —

<b>AxesGroup</b>	<b>Description</b>	The axes group from which the actual position will be read
	<b>Data type</b>	AXES_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>CoordSystem</b>	<b>Description</b>	The coordinate system used when reading the actual position
	<b>Data type</b>	SINT
	<b>Range</b>	One of the following enumeration values: <ul style="list-style-type: none"> <li>• MC_COORDINATE_SYSTEM_ACS = 0</li> <li>• MC_COORDINATE_SYSTEM_MCS = 1</li> <li>• MC_COORDINATE_SYSTEM_PCS = 2</li> </ul>
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Position[]</b>	<b>Description</b>	An array where the position data will be written. The length of the array must equal the maximum number of axes allowed in the group. The maximum number of axes is an argument to the CreateAxesGrp function block that is used to create axes groups.
	<b>Data type</b>	LREAL
	<b>Range</b>	n/a
	<b>Unit</b>	User units
	<b>Default</b>	—

### 2.3.99 Output

<b>Valid</b>	<b>Description</b>	If true, the positions have been read without error
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If true, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See the table <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data type</b>	INT

#### 2.3.99.1.1.1 Example

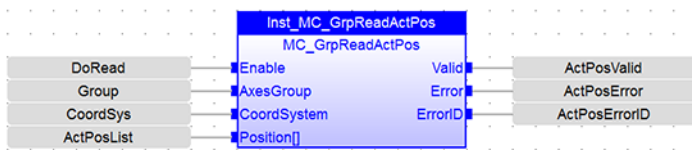
### 2.3.100 Structured Text

```
Inst_MC_GrpReadActPos( DoRead, Group, CoordSys, PosList );
```

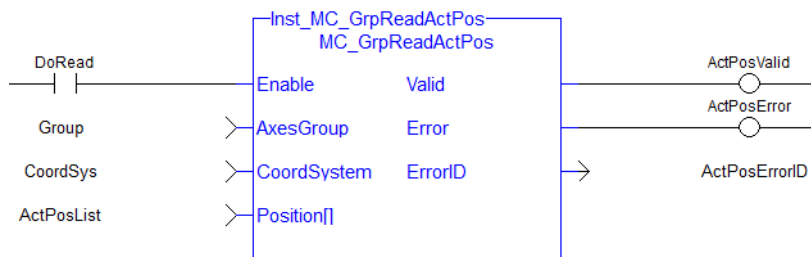
### 2.3.101 IL

```
BEGIN_IL
    CAL Inst_MC_GrpReadActPos( DoRead, Group, CoordSys, PosList
)
END_IL
```

### 2.3.102 FBD



### 2.3.103 Ladder Diagram



2.3.103.1.1 MC\_GrpReadActVel PLCopen ✓ Pipe Network ✓

2.3.103.2.2.1 Description

MC\_GrpReadActVel fills the array specified by the 'Velocity' argument with the actual velocity of the system in the coordinate system specified by the CoordSystem argument. The measured path velocity is also calculated and reported via the 'PathVelocity' output. This function block does not cause any motion.

<b>NOTE</b>	
<ul style="list-style-type: none"> <li>The actual velocity is smoothed over the last 10 samples. This reduces the error in velocity estimation, but introduces a small amount of phase delay in the reported velocities.</li> <li>Currently, only the ACS coordinate system is supported. See <a href="#">Coordinate Systems</a> to learn more.</li> </ul>	

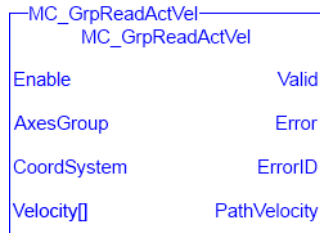
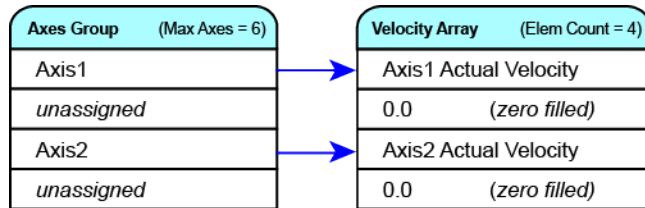


Figure 1-107: MC\_GrpReadActVel

There is a one to one correspondence between the axes in the Axes Group and the velocity values in the Velocity Array. Each element in the Velocity array corresponds to the axis element in the Axes Group array. If a index in the Axes Group is unassigned then the velocity value for that array element in the Velocity array will be 0. If the element does contain an axis then the velocity value will be filled with the current actual velocity for that axis. Here is an example to illustrate how this works:



2.3.104 Related Functions

"MC\_GrpReadActPos" (p. 620), "MC\_GrpReadActAcc" (p. 616), "MC\_GrpReadCmdPos" (p. 626), "MC\_GrpReadCmdVel" (p. 629)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

2.3.104.1.1.1 Arguments

For more detail on how inputs and outputs work, refer to [PLCopen Function Blocks - General Rules](#).

2.3.105 Input

<b>Enable</b>	<b>Description</b>	If True, then this function block will read the current actual velocity of the group and the axes in the group
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a

	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	The axes group from which the actual velocity will be read
	<b>Data type</b>	AXES_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>CoordSystem</b>	<b>Description</b>	The coordinate system used when reading the actual velocity
	<b>Data type</b>	SINT
	<b>Range</b>	One of the following enumeration values: <ul style="list-style-type: none"> <li>• MC_COORDINATE_SYSTEM_ACS = 0</li> <li>• MC_COORDINATE_SYSTEM_MCS = 1</li> <li>• MC_COORDINATE_SYSTEM_PCS = 2</li> </ul>
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Velocity[]</b>	<b>Description</b>	An array where the velocity data will be written. The length of the array must equal the maximum number of axes allowed in the group. The maximum number of axes is an argument to " <a href="#">"MC_CreateAxesGrp"</a> (p. 585) that is used to create axes groups.
	<b>Data type</b>	LREAL
	<b>Range</b>	n/a
	<b>Unit</b>	User units per second
	<b>Default</b>	—

### 2.3.106 Output

<b>Valid</b>	<b>Description</b>	If true, the velocities have been read without error.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If true, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See the table <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data type</b>	INT



<b>PathVelocity</b>	<b>Description</b>	The current measured path velocity of the group, measured by taking the square root of the sum of the squared velocities of each axis.
	<b>Data type</b>	LREAL
	<b>Unit</b>	User units per second

**2.3.106.1.1.1 Example**

**2.3.107 Structured Text**

```
Inst_MC_GrpReadActVel (DoRead, Group, CoordSys, VelList);
```

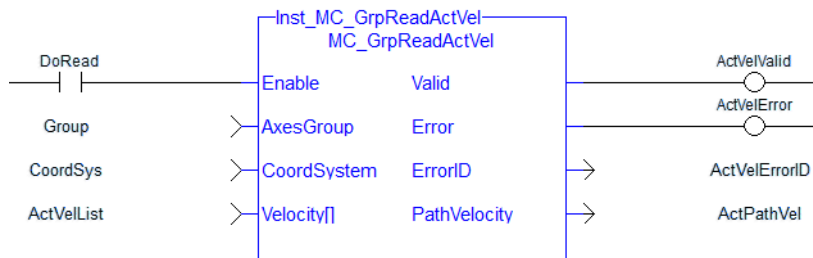
**2.3.108 IL**

```
BEGIN_IL
    CAL Inst_MC_GrpReadActVel (DoRead, Group, CoordSys,
    VelList)
END_IL
```

**2.3.109 FBD**



**2.3.110 FFLD**



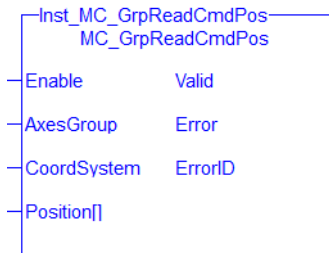
**2.3.110.1.1 MC\_GrpReadCmdPos** PLCopen ✓ Pipe Network ✓

**2.3.110.2.2.1 Description**

MC\_GrpReadCmdPos fills the array (specified by the `Position` argument) with the commanded position of the coordinate system specified by the `CoordSystem` argument. This function block does not cause any motion.

**NOTE**

Currently, only the ACS coordinate system is supported. See [Coordinate Systems](#) to learn more.



**Figure 1-108:** MC\_GrpReadCmdPos

There is a one to one correspondence between the axes in the Axes Group and the position values in the Position Array. Each element in the Position Array corresponds to the axis element in the Axis Group array. If an index in the Axes Group is unassigned then the position value for that array element in the Position Array will be 0. If the element does contain an axis then the position value will be filled with the current actual position for that axis. Here is an example to illustrate how this works:



**2.3.111 Related Function Blocks**

"MC\_GrpReadActPos" (p. 620), "MC\_GrpReadActVel" (p. 623), "MC\_GrpReadActAcc" (p. 616), "MC\_GrpReadCmdVel" (p. 629)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

**2.3.111.1.1.1 Arguments**

For more detail on how inputs and outputs work, refer to [PLCopen Function Blocks - General Rules](#).

**2.3.112 Input**

Enable	Description
	If True, then this function block will read the current commanded position of the axes in the group
	<b>Data type</b> BOOL
	<b>Range</b> 0, 1
	<b>Unit</b> n/a
	<b>Default</b> —

<b>AxesGroup</b>	<b>Description</b>	The axes group from which the commanded position will be read.
	<b>Data type</b>	AXES_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>CoordSystem</b>	<b>Description</b>	The coordinate system used when reading the commanded position.
	<b>Data type</b>	SINT
	<b>Range</b>	One of the following enumeration values: <ul style="list-style-type: none"> <li>• MC_COORDINATE_SYSTEM_ACS = 0</li> <li>• MC_COORDINATE_SYSTEM_MCS = 1</li> <li>• MC_COORDINATE_SYSTEM_PCS = 2</li> </ul>
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Position[]</b>	<b>Description</b>	An array where the position data will be written. The length of the array must equal the maximum number of axes allowed in the group. The maximum number of axes is an argument to <a href="#">"MC_CreateAxesGrp"</a> (p. 585), which is used to create axes groups.
	<b>Data type</b>	LREAL
	<b>Range</b>	n/a
	<b>Unit</b>	User units
	<b>Default</b>	—

### 2.3.113 Output

<b>Valid</b>	<b>Description</b>	If true, that the positions have been read without error.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If true, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See the table in <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data type</b>	INT

#### 2.3.113.1.1.1 Example

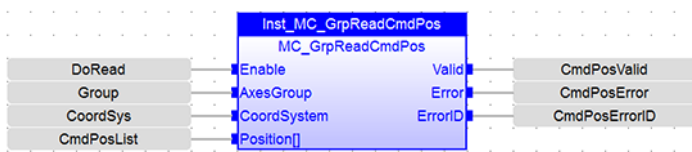
### 2.3.114 Structured Text

```
(*MC_GrpReadCmdPos ST example *)
Inst_MC_GrpReadCmdPos(DoRead, Group, CoordSys, PosList );
```

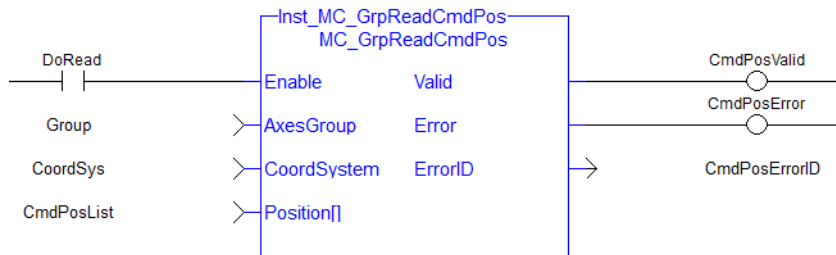
### 2.3.115 IL

```
BEGIN_IL
    CAL Inst_MC_GrpReadCmdPos( DoRead, Group, CoordSys, PosList
)
END_IL
```

### 2.3.116 FBD



### 2.3.117 FFLD



2.3.117.1.1 MC\_GrpReadCmdVel PLCopen ✓ Pipe Network ✓

2.3.117.2.2.1 Description

MC\_GrpReadCmdVel fills the array specified by the `Velocity` argument with the commanded velocity for the coordinate system, which is specified by the `CoordSystem` argument. The path velocity is also reported via the 'PathVelocity' output. This function block does not cause any motion.

**NOTE**  
Currently, only the ACS coordinate system is supported. See [Coordinate Systems](#) to learn more.

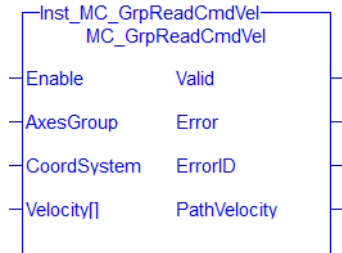
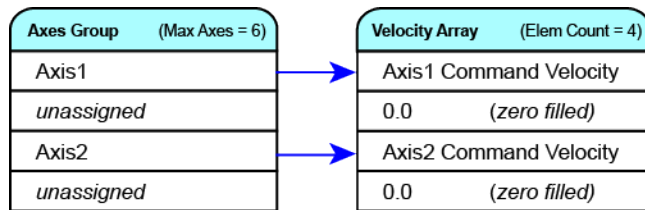


Figure 1-109: MC\_GrpReadCmdVel

There is a one to one correspondence between the axes in the Axes Group and the velocity values in the Velocity Array. Each element in the Velocity Array corresponds to the axis element in the Axis Group array. If an index in the Axes Group is unassigned then the velocity value for that array element in the Velocity Array will be 0. If the element does contain an axis then the velocity value will be filled with the current velocity for that axis. Here is an example to illustrate how this works:



2.3.118 Related Function Blocks

"MC\_GrpReadActPos" (p. 620), "MC\_GrpReadActVel" (p. 623), "MC\_GrpReadActAcc" (p. 616), "MC\_GrpReadCmdPos" (p. 626)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

2.3.118.1.1.1 Arguments

For more detail on how inputs and outputs work, refer to [PLCopen Function Blocks - General Rules](#).

2.3.119 Input

<b>Enable</b>	<b>Description</b>	If True, then this function block will read the current commanded velocity of the group and the axes in the group
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>AxesGroup</b>	<b>Description</b>	The axes group from which the commanded velocity will be read.
	<b>Data type</b>	AXES_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>CoordSystem</b>	<b>Description</b>	The coordinate system used when reading the commanded velocity.
	<b>Data type</b>	SINT
	<b>Range</b>	One of the following enumeration values: <ul style="list-style-type: none"> <li>• MC_COORDINATE_SYSTEM_ACS = 0</li> <li>• MC_COORDINATE_SYSTEM_MCS = 1</li> <li>• MC_COORDINATE_SYSTEM_PCS = 2</li> </ul>
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Velocity[]</b>	<b>Description</b>	An array where the velocity data will be written. The length of the array must equal the maximum number of axes allowed in the group. The maximum number of axes is an argument to " <a href="#">MC_CreateAxesGrp</a> " (p. 585), which is used to create axes groups.
	<b>Data type</b>	LREAL
	<b>Range</b>	n/a
	<b>Unit</b>	User units per second
	<b>Default</b>	—

### 2.3.120 Output

<b>Valid</b>	<b>Description</b>	If true, that the velocities have been read without error.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If true, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See the table in <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data type</b>	INT

<b>PathVelocity</b>	<b>Description</b>	The current commanded path velocity of the group, measured by taking the square root of the sum of the squared velocities of each axis.
	<b>Data type</b>	LREAL
	<b>Unit</b>	User units per second

### 2.3.120.1.1.1 Example

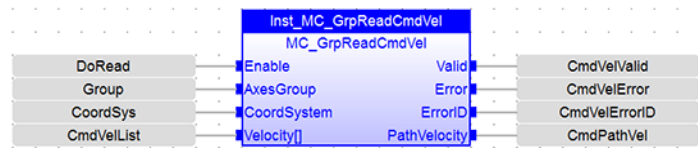
### 2.3.121 Structured Text

```
(*MC_GrpReadCmdVel ST example *)
Inst_MC_GrpReadCmdVel(DoRead, Group, CoordSys, VelList );
```

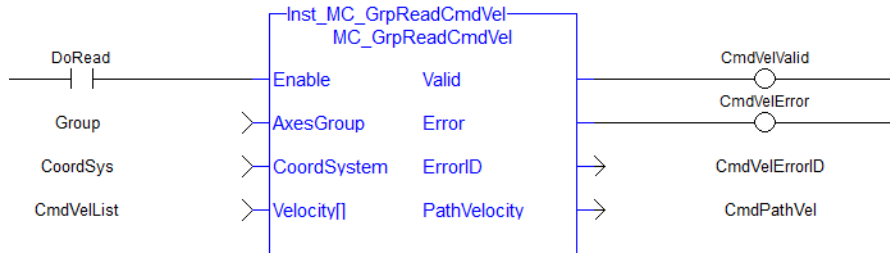
### 2.3.122 IL

```
BEGIN_IL
    CAL Inst_MC_GrpReadCmdVel( DoRead, Group, CoordSys, VelList
)
END_IL
```

### 2.3.123 FBD



### 2.3.124 FFLD



### 2.3.124.1.1 MC\_GrpReadError

#### 2.3.124.2.2.1 Description

This function describes general axes group errors. This function does not cause any motion.

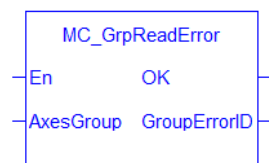


Figure 1-110: MC\_GrpReadError

#### 2.3.125 Related Functions

"MC\_GrpReset" (p. 597), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

##### 2.3.125.1.1.1 Arguments

#### 2.3.126 Input

<b>En</b>	<b>Description</b>	Enables execution
	<b>Data Type</b>	BOOL
	<b>Range</b>	0,1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	The axes group from which the GroupErrorID will be read.
	<b>Data Type</b>	AXES_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 2.3.127 Output

<b>OK</b>	<b>Description</b>	Indicates the function executed successfully
	<b>Data Type</b>	BOOL
<b>GroupErrorID</b>	<b>Description</b>	Displays the Error ID for the given Axis Group. See table in <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data Type</b>	INT

##### 2.3.127.1.1.1 Examples

#### 2.3.128 Structured Text



```
//Read a group error number
GroupErrorID:= MC_GrpReadError( Axis_Group );
```

### 2.3.129 FBD



### 2.3.130 FFLD



**2.3.130.1.1 MC\_GrpReadStatus** PLCopen ✓ Pipe Network ✓

**2.3.130.2.2.1 Description**

MC\_GrpReadStatus returns the status of an axes group. This function block does not cause any motion. Refer to [Group State Diagrams](#) for details.

**NOTE**

The following output is not currently supported. It will be supported in a future release.

- GroupHoming



Figure 1-111: MC\_GrpReadStatus

**2.3.131 Related Functions**

"MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

**2.3.131.1.1.1 Arguments**

**2.3.131.2.2.2 Input**

<b>Enable</b>	<b>Description</b>	If True, then the axes group status will be read.
	<b>Data type</b>	BOOL
	<b>Range</b>	0..1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	The axis group from which the status will be read
	<b>Data type</b>	AXIS_GROUP_REF

<b>Range</b>	n/a
<b>Unit</b>	n/a
<b>Default</b>	—

### 2.3.131.3.3.3 Output

<b>Valid</b>	<b>Description</b>	True if valid outputs are available
	<b>Data type</b>	BOOL
<b>GroupMoving <sup>1</sup></b>	<b>Description</b>	The axes group is in the Moving state, indicating that the group is enabled and currently executing a coordinated motion command.
	<b>Data type</b>	BOOL
<b>GroupHoming <sup>1</sup></b>	<b>Description</b>	Not supported
	<b>Data type</b>	BOOL
<b>GroupErrorStop <sup>1</sup></b>	<b>Description</b>	The axes group is in the ErrorStop state due to an axis error or group error. The group cannot accept coordinated motion commands. The execution of MC_GrpReset is required to change the group's state from ErrorStop to Standby.
	<b>Data type</b>	BOOL
<b>GroupStandby <sup>1</sup></b>	<b>Description</b>	The axes group is in the Standby state, meaning that the group is enabled and all its axes are enabled and the group is not currently executing a coordinated motion command. The axes group is ready to accept coordinated motion commands.
	<b>Data type</b>	BOOL
<b>GroupStopping <sup>1</sup></b>	<b>Description</b>	The axes group is in the Stopping state due the execution of MC_GrpStop. The axes group is enabled but cannot accept coordinated motion commands while in the Stopping state. The axes group remains in the Stopping state while MC_GrpStop is executing and will remain in the Stopping state while MC_GrpStop's Execute input is held high.
	<b>Data type</b>	BOOL
<b>GroupDisabled <sup>1</sup></b>	<b>Description</b>	The axis group is in the Disabled state and cannot accept coordinated motion commands.
	<b>Data type</b>	BOOL
<b>ConstantVelocity</b>	<b>Description</b>	True if the commanded path velocity is the same between the current scan of the application program and the previous scan.  ConstantVelocity is always TRUE for Direct moves. The commanded path velocity of Direct moves is always zero.
	<b>Data type</b>	BOOL

<b>Accelerating</b>	<b>Description</b>	True if the commanded path velocity is accelerating between the current scan of the application program and the previous scan.
	<b>Data type</b>	BOOL
<b>Decelerating</b>	<b>Description</b>	True if the commanded path velocity is decelerating between the current scan of the application program and the previous scan.
	<b>Data type</b>	BOOL
<b>InPosition</b>	<b>Description</b>	True indicates that the axes group is "in position". The following must be true for the axes group to be "in position": <ul style="list-style-type: none"> <li>• The axes group is enabled.</li> <li>• There are no moves in the group's queue.</li> <li>• The servo loop is closed for each axis in the group.</li> <li>• There are no moves in the individual axis queue for each axis in the group.</li> <li>• The command delta is zero for each axis in the group.</li> <li>• The actual position is within the In-Position Bandwidth of the command position for each axis in the group.</li> </ul>
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error identifier if the Error output is set to TRUE. See the table in <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Data type</b>	BOOL

<sup>1</sup> These outputs are mutually exclusive, meaning only one will be true at a time. All others will be false. Please refer to the [Group State Diagrams](#).

#### 2.3.131.4.4.4 Example

#### 2.3.132 Structured Text

```
//Check boolean status bits for an Axis Group
Inst_MC_GrpReadStatus( EnableGrpReadStatus, Group1_Ref );

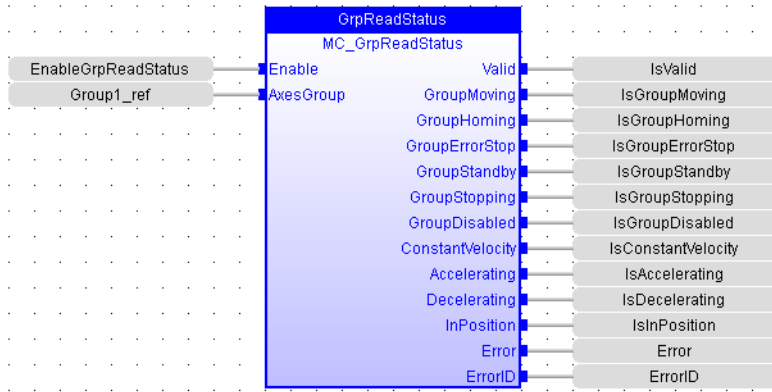
IsGroupMoving:= Inst_MC_GrpReadStatus.GroupMoving;
IsGroupErrorStop:= Inst_MC_GrpReadStatus.GroupErrorStop;
IsGroupStandby:= Inst_MC_GrpReadStatus.GroupStandby;
IsGroupDisabled:= Inst_MC_GrpReadStatus.GroupDisabled;
Accelerating:= Inst_MC_GrpReadStatus.Accelerating;
IsConstantVelocity:= Inst_MC_GrpReadStatus.ConstantVelocity;
IsInPosition:= Inst_MC_GrpReadStatus.InPosition;
```

#### 2.3.133 IL

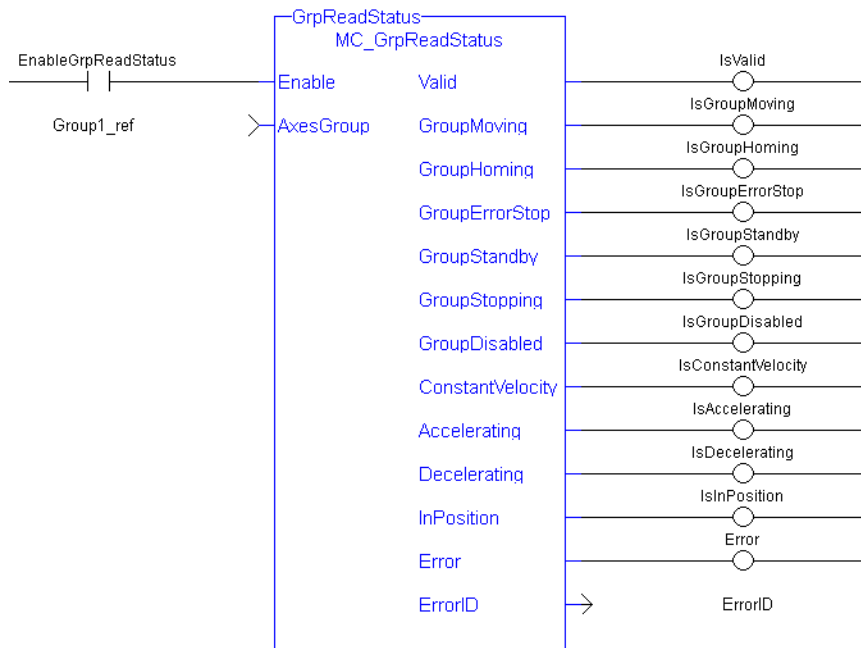
```

BEGIN_IL
  CAL Inst_MC_GrpReadStatus( EnableGrpReadStatus, Group1_Ref
)
END_IL
    
```

### 2.3.134 FBD



### 2.3.135 FFLD



### 2.3.135.1 Coordinated Motion Motion Library

Function	Description
"MC_AxisSetDefaults" (p. 638)	Sets the default kinematic parameters for an axis.
"MC_GrpHalt" (p. 642)	Performs a controlled motion stop of all the axes in the group
"MC_GrpSetOverride" (p. 645)	Sets the velocity factor that is multiplied to the commanded velocity of all axes in the group.
"MC_MoveCircAbs" (p. 647)	Commands interpolated circular movement on an axes group to the specified absolute positions.
"MC_MoveCircRel" (p. 655)	Commands interpolated circular movement on an axes group to the specified relative positions.
"MC_MoveDirAbs" (p. 662)	Commands movement of an axes group to an absolute position regardless of path.
"MC_MoveDirRel" (p. 665)	Commands movement of an axes group to a relative position regardless of path.
"MC_MoveLinAbs" (p. 668)	Commands interpolated linear movement on an axes group to the specified absolute positions.
"MC_MoveLinRel" (p. 674)	Commands interpolated linear movement on an axes group to the specified relative positions.

#### 2.3.135.2.1 MC\_AxisSetDefaults

#### 2.3.135.3.2.1 Description

MC\_AxisSetDefaults sets the default kinematic variables for "MC\_MoveDirAbs" (p. 662) and "MC\_MoveDirRel" (p. 665). These variables are only used with the MC\_MoveDir function blocks.

Each axis within the group must have the default kinematic parameters of Velocity, Acceleration, Deceleration, and Jerk set to values greater than zero. A non-zero Jerk value will perform an S-Curve rather than a trapezoidal move. Each axis within the group must have these values set before a direct move can be started.

#### NOTE

Jerk with a non-zero value is currently not supported for coordinated motion. Jerk parameters are currently ignored.

The function block returns an error if the group state is not GroupStandby or GroupDisabled.

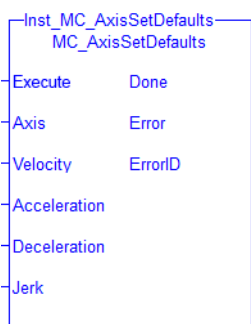


Figure 1-112: MC\_AxisSetDefaults

### 2.3.136 Related Functions

"MC\_MoveDirAbs" (p. 662), "MC\_MoveDirRel" (p. 665), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

See also:

- [Differences Between Functions and Function Blocks](#)
- Calling a function

### 2.3.136.1.1.1 Arguments

### 2.3.137 Input

<b>Execute</b>	<b>Description</b>	On the rising edge, request to set the default kinematic parameters.
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Reference to the axis which will have its default kinematic parameters set.
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Velocity</b>	<b>Description</b>	The default velocity.
	<b>Data type</b>	LREAL
	<b>Range</b>	$0 < \text{Velocity} < ( 20 * \text{Acceleration} )$ and $0 < \text{Velocity} < ( 20 * \text{Deceleration} )$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	User units per second
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Trapezoidal: Acceleration rate <a href="#">S-curve</a> : Maximum acceleration <a href="#">see "Selection of Acceleration and Jerk Parameters for Function Blocks"</a>
	<b>Data type</b>	LREAL
	<b>Range</b>	$( \text{Velocity} / 20 ) < \text{Acceleration} < ( 2 * \text{Jerk} )$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	User units per second <sup>2</sup>
	<b>Default</b>	—

<b>Deceleration</b>	<b>Description</b>	Trapezoidal: Deceleration rate <a href="#">S-curve</a> : Unused
	<b>Data type</b>	LREAL
	<b>Range</b>	$(\text{Velocity} / 20) < \text{Deceleration} < (2 * \text{Jerk})$
	<b>Unit</b>	User unit per second <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Trapezoidal: 0 <a href="#">S-curve</a> : Constant jerk
		<b>NOTE</b> Currently the Jerk value is ignored for motion. Only trapezoidal motion is supported.  <a href="#">see "Selection of Acceleration and Jerk Parameters for Function Blocks"</a>
	<b>Data type</b>	LREAL
	<b>Range</b>	$(\text{Velocity} / 20) < \text{Acceleration} < (2 * \text{Jerk})$ and $(\text{Velocity} / 20) < \text{Deceleration} < (2 * \text{Jerk})$
	<b>Unit</b>	User units per second <sup>3</sup>
	<b>Default</b>	—

### 2.3.138 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, then an error has occurred
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See the table in <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Data type</b>	DINT

#### 2.3.138.1.1.1 Example

### 2.3.139 Structured Text

```
(* ST MC_AxisSetDefaults Example *)
```

```
default_velocity      := 50.0;
default_acceleration  := 250.0;
default_deceleration  := 300.0;
default_jerk          := 1000.0;
```

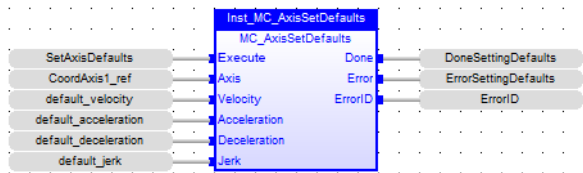


```
Inst_MC_AxisSetDefaults ( TRUE, CoordAxis1_ref, default_
velocity, default_acceleration, default_deceleration, default_
jerk);
```

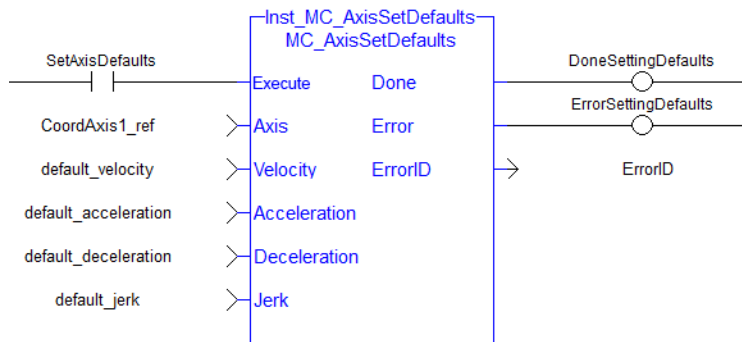
### 2.3.140 Instruction List

```
BEGIN_IL
    CAL Inst_MC_AxisSetDefaults( TRUE, CoordAxis1_Ref, default_
velocity, default_acceleration, default_deceleration, default_
jerk)
END_IL
```

### 2.3.141 Function Block Diagram



### 2.3.142 Ladder Diagram



**2.3.142.1.1 MC\_GrpHalt** PLCopen ✓ Pipe Network ✓

**2.3.142.2.2.1 Description**

MC\_GrpHalt performs a controlled motion stop of all the axes in the group. When the path velocity reaches zero any queued moves are flushed from the buffer, the Done output is set, and the state transitions to GroupStandby. Unlike MC\_GrpStop, MC\_GrpHalt can be aborted.

**NOTE**

MC\_GrpHalt does NOT prevent a single axis from executing nor does it prevent other Coordinated Motion moves from executing once MC\_GrpHalt has completed.



**2.3.143 Related Functions**

"MC\_GrpStop" (p. 599), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

**2.3.143.1.1.1 Arguments**

For more details on how inputs and outputs work, refer to [PLCopen Function Blocks - General Rules](#).

**2.3.144 Input**

<b>Execute</b>	<b>Description</b>	On the rising edge the command to halt all of the axes in the group is initiated.
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	The axes group in which the axes will be stopped.
	<b>Data type</b>	AXES_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	The path deceleration rate for all of the axes in the group

	<b>Data type</b>	LREAL
	<b>Range</b>	0 < Deceleration See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>JerK</b>	<b>Description</b>	Not supported
	<b>Data type</b>	LREAL
	<b>Range</b>	0 ≤ Jerk See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 2.3.145 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	TRUE from the moment the EXECUTE input is TRUE until the time the halt is completed.
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	Indicates that the halt is still executing.
	<b>Data type</b>	BOOL
<b>CommandAborted</b>	<b>Description</b>	If True, command was aborted by another FB.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error identifier if Error output is set to TRUE.. See table in <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Data type</b>	INT

#### 2.3.145.1.1.1 Example

### 2.3.146 Structured Text

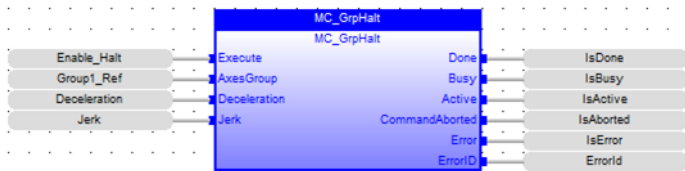
```
Inst_MC_GrpHalt ( EnableHalt, Group1_Ref, Deceleration, Jerk );
```

### 2.3.147 IL

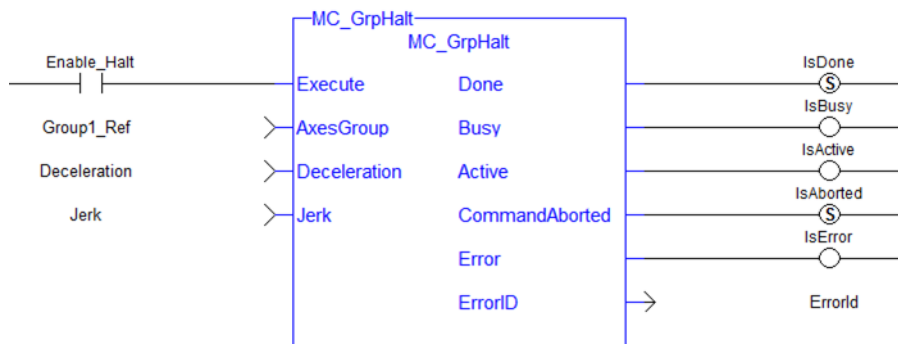
```

BEGIN_IL
    CAL Inst_MC_GrpHalt ( EnableHalt, Group1_Ref, Deceleration,
    Jerk )
END_IL
    
```

### 2.3.148 FBD



### 2.3.149 FFLD



### 2.3.149.1.1 MC\_GrpSetOverride

#### 2.3.149.2.2.1 Description

MC\_GrpSetOverride sets the velocity factor that is multiplied to the commanded velocity of all axes in the group. This function block in itself does not cause any motion.

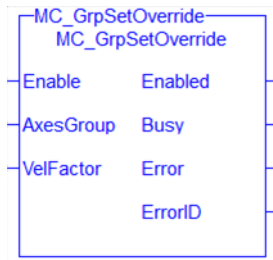


Figure 1-113: MC\_GrpSetOverride

### 2.3.150 Related Functions

"MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

#### 2.3.150.1.1.1 Arguments

For more detail on how inputs and outputs work, refer to [PLCopen Function Blocks - General Rules](#).

### 2.3.151 Input

<b>Enable</b>	<b>Description</b>	On the rising edge, changes the velocity multiplier for the axes group.
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	The axes group in which the velocity multiplier will be applied.
	<b>Data type</b>	AXES_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>VelFactor</b>	<b>Description</b>	The new multiplier factor for the commanded velocity of the axes group.
	<b>Data type</b>	REAL
	<b>Range</b>	[0.0 .. 2.0]
	<b>Unit</b>	n/a

<b>Default</b>	—
----------------	---

### 2.3.152 Output

<b>Enabled</b>	<b>Description</b>	Indicates that the override was successful.
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	If True, then the FB is executing.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error identifier if Error output is set to TRUE. See the table <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Data type</b>	INT

#### 2.3.152.1.1.1 Example

### 2.3.153 ST

```
Inst_MC_GrpSetOverride( EnableOverride, Group1_Ref,
VelocityFactor );
```

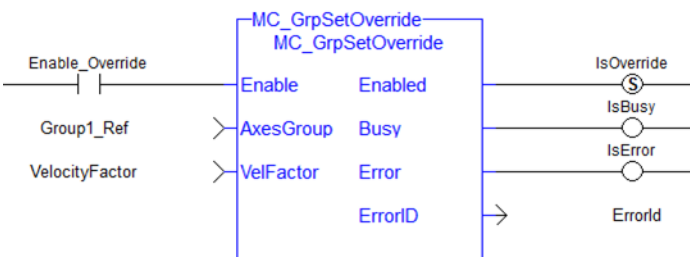
### 2.3.154 IL

```
BEGIN_IL
    CAL Inst_MC_GrpSetOverride( EnableOverride, Group1_Ref,
VelocityFactor )
END_IL
```

### 2.3.155 FBD



### 2.3.156 FFLD



### 2.3.156.1.1 MC\_MoveCircAbs

#### 2.3.156.2.2.1 Description

MC\_MoveCircAbs commands interpolated circular movement on an axes group to the specified absolute positions in the coordinate system as specified by the 'CoordSystem' argument. See [Circular Moves Diagrams](#) for detailed information on the movement options.

#### NOTE

- An error is returned if the group is in the GroupDisabled state.
- An error is returned if the input parameters do not meet the required precision. See [Precision Requirements for Circular Move Input Parameters](#) for more information.

#### NOTE

- Circular motion is only supported for axes groups with only two attached axes
- S-Curve motion is not currently supported. The *Jerk* input is currently ignored. S-Curve motion and the *Jerk* argument will be supported in a future release. .

When all motion has completed successfully, the state of the axes group goes to GroupStandby.

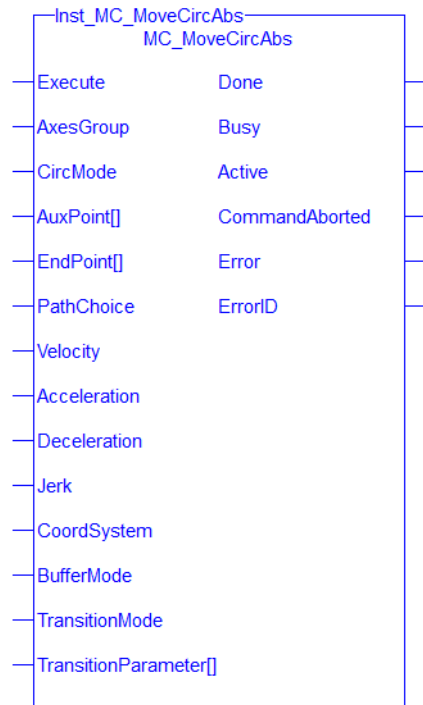


Figure 1-114: MC\_MoveCircAbs

### 2.3.157 Related Functions

"MC\_MoveCircRel" (p. 655), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

#### 2.3.157.1.1.1 Arguments

### 2.3.158 Input

Execute	Description
	On the rising edge request to perform a circular absolute move

	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	The axis group that will perform the circular absolute move
	<b>Data type</b>	AXIS_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>CircMode</b>	<b>Description</b>	Specifies the meaning of the AuxPoint[ ] input.
	<b>Data type</b>	SINT One of the following enumeration values: <ul style="list-style-type: none"> <li>• MC_CIRC_MODE_BORDER = 0</li> <li>• MC_CIRC_MODE_CENTER = 1</li> </ul>
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AuxPoint[]</b>	<b>Description</b>	Array of absolute positions for each axis in the group. The meaning depends on the value of the CircMode input: <ul style="list-style-type: none"> <li>• MC_CIRC_MODE_BORDER: AuxPoint defines a point on the circle which is crossed on the path from the starting to the end point.</li> <li>• MC_CIRC_MODE_CENTER: AuxPoint defines the center point of the circle.</li> </ul>
	<b>Data type</b>	LREAL
	<b>Range</b>	[0, Number of axes in group - 1]
	<b>Unit</b>	n/a



	<b>Required Precision</b>	1 part in 100,000. See <a href="#">Precision Requirements for Circular Move Input Parameters</a> .
	<b>Default</b>	—
<b>EndPoint[]</b>	<b>Description</b>	Array of absolute end positions for each axis in the group
	<b>Data type</b>	LREAL
	<b>Range</b>	[0, Number of axes in group - 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
	<b>Required Precision</b>	1 part in 100,000. See <a href="#">Precision Requirements for Circular Move Input Parameters</a> .
<b>PathChoice</b>	<b>Description</b>	Specifies the direction of the path. This argument is only relevant when CircMode is MC_CIRC_MODE_CENTER.
	<b>Data type</b>	SINT One of the following enumeration values: <ul style="list-style-type: none"> <li>• MC_CIRC_PATHCHOICE_CLOCKWISE = 0 = Clockwise</li> <li>• MC_CIRC_PATHCHOICE_COUNTERCLOCKWISE = 1 = Counterclockwise</li> </ul>
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Velocity</b>	<b>Description</b>	Maximum velocity of the defined path
	<b>Data type</b>	LREAL
	<b>Range</b>	$0 < \text{Velocity} < (20 * \text{Acceleration})$ and $0 < \text{Velocity} < (20 * \text{Deceleration})$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	user units per second
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Maximum acceleration
	<b>Data type</b>	LREAL

	<b>Range</b>	$0 < \text{Velocity} < ( 20 * \text{Acceleration} )$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	user units per second <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Maximum Deceleration
	<b>Data type</b>	LREAL
	<b>Range</b>	$0 < \text{Velocity} < ( 20 * \text{Deceleration} )$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	user units per second <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Maximum jerk
	<b>Data type</b>	LREAL
	<b>Range</b>	<p><b>For trapezoidal velocity profiles: 0</b></p> <p><b>For S-Curve velocity profiles: (</b>  <math>\text{Velocity} / 20 ) &lt; \text{Acceleration} &lt; ( 2 * \text{Jerk} )</math>  <b>and (</b> <math>\text{Velocity} / 20 ) &lt; \text{Deceleration} &lt; ( 2 * \text{Jerk} )</math>  <b>)</b></p> See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
		<div style="background-color: #f0f0f0; padding: 5px;"> <p><b>NOTE</b></p> <p>S-Curve motion is currently not supported and the <i>Jerk</i> input is currently ignored. S-Curve motion and the <i>Jerk</i> argument will be supported in a future release.</p> </div>
	<b>Unit</b>	user units per second <sup>3</sup>
	<b>Default</b>	—
<b>CoordSystem</b>	<b>Description</b>	The coordinate system used when commanding the circular absolute move.  Currently, only the ACS coordinate system is supported. See Coordinate Systems to learn more.
	<b>Data type</b>	SINT

	<b>Range</b>	One of the following enumeration values: <ul style="list-style-type: none"> <li>• MC_COORDINATE_SYSTEM_ACS = 0</li> <li>• MC_COORDINATE_SYSTEM_MCS = 1</li> <li>• MC_COORDINATE_SYSTEM_PCS = 2</li> </ul>
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>BufferMode</b>	<b>Description</b>	<p>Defines the chronological sequence of the function block relative to the previous block.</p> <p>The blending modes (2, 3, 4, &amp; 5) match the <b>path velocity</b> at the active move's endpoint. Some individual <b>axis velocities</b> may make an abrupt change if the path of the next move travels in a different direction. A transition move may be programmed at the <code>TransitionMode</code> input to avoid this. See the table in <a href="#">Buffer Modes</a>.</p>
	<b>Data type</b>	<p>SINT</p> <p>One of the following enumeration values:</p> <ul style="list-style-type: none"> <li>• MC_BUFFER_MODE_BUFFERED = 1 = Buffered</li> <li>• MC_BUFFER_MODE_BLENDED_PREVIOUS = 2 = Blending Previous</li> <li>• MC_BUFFER_MODE_BLENDED_NEXT = 3 = Blending Next</li> <li>• MC_BUFFER_MODE_BLENDED_LOW = 4 = Blending Low</li> <li>• MC_BUFFER_MODE_BLENDED_HIGH = 5 = Blending High</li> </ul> <p><i>BufferMode = Abort = 0 is not allowed with this function block.</i></p>
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>TransitionMode</b>	<b>Description</b>	Coupled with the TransitionParameter[ ], this input defines the shape and dynamics of the inserted contour to connect the current motion with the next motion in the queue.  See <a href="#">Transition Between Moves</a> for additional information.
	<b>Data type</b>	SINT
	<b>Range</b>	The value is limited to the following: <ul style="list-style-type: none"> <li>MC_TRANSITION_MODE_NONE = 0</li> <li>MC_TRANSITION_MODE_CORNER_DISTANCE = 3</li> </ul>
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>TransitionParameter[ ]</b>	<b>Description</b>	This array is dependent on the TransitionMode specified. The transition parameter values are applied to the axis group. See <a href="#">table: "Transition Mode Parameters"</a> for details.
	<b>Data type</b>	LREAL
	<b>Range</b>	[1, N]  N values are supplier specified dependent on the TransitionMode selected.
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 2.3.159 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	If True, then the function block is executing.
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	If True, then the function block is controlling motion.
	<b>Data type</b>	BOOL
<b>CommandAborted</b>	<b>Description</b>	If True, command was aborted by another function block.
	<b>Data type</b>	BOOL

<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See table in <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data type</b>	INT

**2.3.159.1.1.1 Example**

**2.3.160 ST**

```

Inst_MC_MoveCircAbs( ExecuteMove, Group1_Ref, MC_CIRC_MODE_
BORDER, AuxPoints, EndPoints, MC_CIRC_PATHCHOICE_CLOCKWISE,
Velocity, Acceleration, Deceleration, Jerk, MC_COORDINATE_
SYSTEM_ACS, MC_BUFFER_MODE_ABORTING, MC_TRANSITION_MODE_NONE,
TransitionParams );
    
```

**2.3.161 IL**

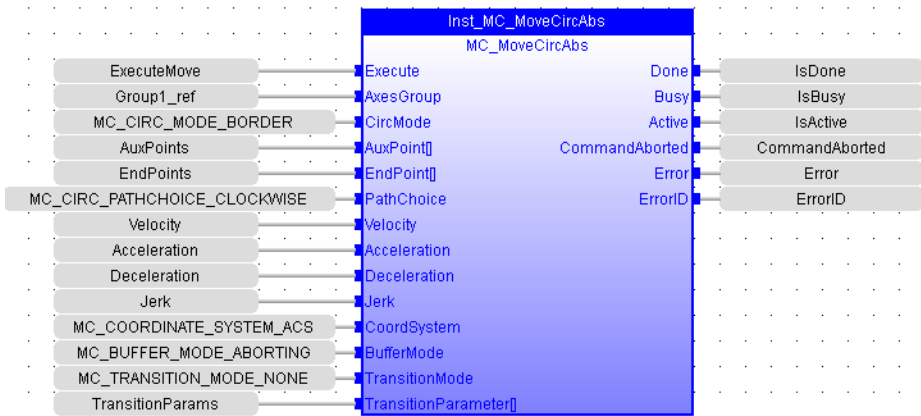
```

BEGIN_IL

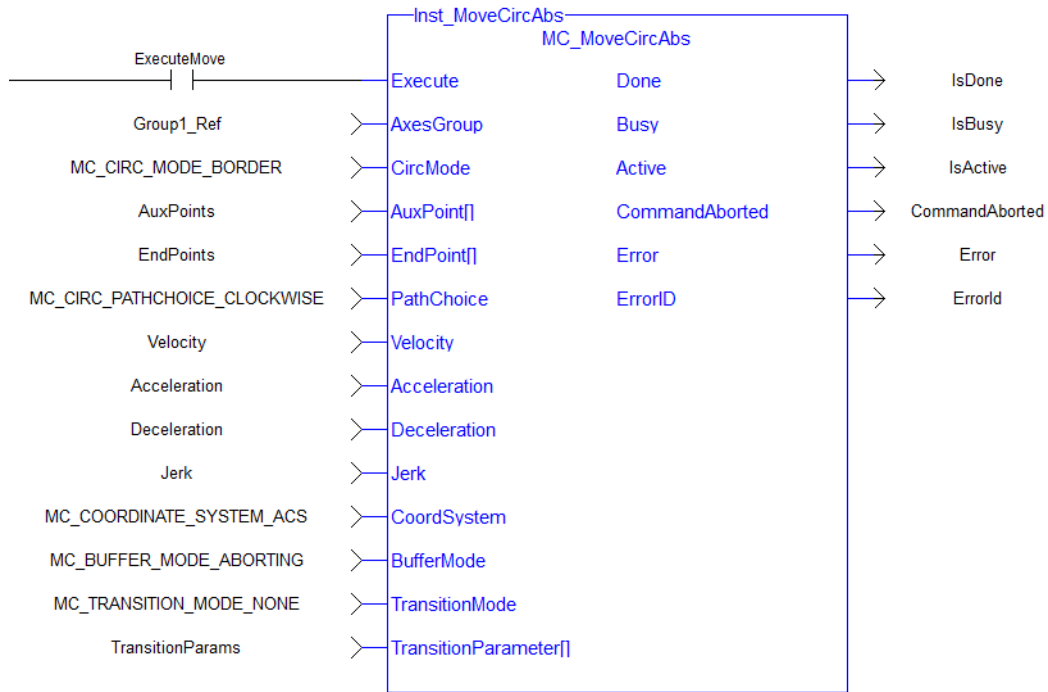
    CAL Inst_MC_MoveCircAbs( ExecuteMove, Group1_Ref, MC_CIRC_
MODE_BORDER, AuxPoints, EndPoints, MC_CIRC_PATHCHOICE_CLOCKWISE,
Velocity, Acceleration, Deceleration, Jerk, MC_COORDINATE_
SYSTEM_ACS, MC_BUFFER_MODE_ABORTING, MC_TRANSITION_MODE_NONE,
TransitionParams )

END_IL
    
```

**2.3.162 FBD**



**2.3.163 FFLD**



### 2.3.163.1.1 MC\_MoveCircRel PLCopen ✓ Pipe Network ✓

#### 2.3.163.2.2.1 Description

MC\_MoveCircRel commands interpolated circular movement on an axes group to the specified relative positions in the coordinate system as specified by the 'CoordSystem' argument. See [Circular Moves Diagrams](#) for detailed information on the movement options.

#### NOTE

An error is returned if the group is in the GroupDisabled state.

#### NOTE

- Circular motion is only supported for axes groups with only two attached axes
- S-Curve motion is not currently supported. The *Jerk* input is currently ignored. S-Curve motion and the *Jerk* argument will be supported in a future release. .

When all motion has completed successfully, the state of the axes group goes to GroupStandby.

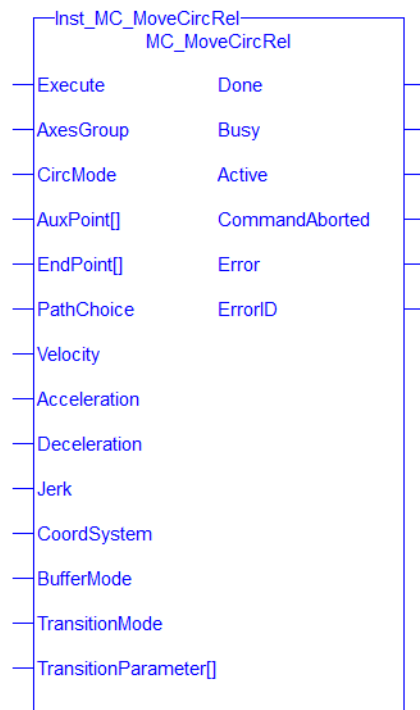


Figure 1-115: MC\_MoveCircRel

### 2.3.164 Related Functions

"MC\_MoveCircAbs" (p. 647), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

#### 2.3.164.1.1.1 Arguments

#### 2.3.165 Input

Execute	Description
	On the rising edge request to perform a circular relative move.
	Data type
	BOOL

	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	The axis group that will perform the circular relative move
	<b>Data type</b>	AXIS_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>CircMode</b>	<b>Description</b>	Specifies the meaning of the AuxPoint[ ] input (see AuxPoint[ ] below).
	<b>Data type</b>	SINT One of the following enumeration values: <ul style="list-style-type: none"> <li>• MC_CIRC_MODE_BORDER = 0</li> <li>• MC_CIRC_MODE_CENTER = 1</li> </ul>
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AuxPoint[]</b>	<b>Description</b>	Array of relative positions for each axis in the group. The meaning depends on the value of the CircMode input: <ul style="list-style-type: none"> <li>• MC_CIRC_MODE_BORDER: AuxPoint defines a point on the circle which is crossed on the path from the starting to the end point.</li> <li>• MC_CIRC_MODE_CENTER: AuxPoint defines the center point of the circle.</li> </ul> <p>In all cases the points are relative to the starting point.</p>
	<b>Data type</b>	LREAL
	<b>Range</b>	[0, Number of axes in group - 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
	<b>Required Precision</b>	1 part in 100,000. See <a href="#">Precision Requirements for Circular Move Input Parameters</a> .



<b>EndPoint[]</b>	<b>Description</b>	Array of relative end positions for each axis in the group.
	<b>Data type</b>	LREAL
	<b>Range</b>	[0, Number of axes in group - 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
	<b>Required Precision</b>	1 part in 100,000. See <a href="#">Precision Requirements for Circular Move Input Parameters</a> .
<b>PathChoice</b>	<b>Description</b>	Specifies the direction of the path. This argument is only relevant when CircMode is MC_CIRC_MODE_CENTER.
	<b>Data type</b>	SINT One of the following enumeration values: <ul style="list-style-type: none"> <li>MC_CIRC_PATHCHOICE_CLOCKWISE = 0 = Clockwise</li> <li>MC_CIRC_PATHCHOICE_COUNTERCLOCKWISE = 1 = Counterclockwise</li> </ul>
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Velocity</b>	<b>Description</b>	Maximum velocity of the defined path
	<b>Data type</b>	LREAL
	<b>Range</b>	$0 < \text{Velocity} < (20 * \text{Acceleration})$ and $0 < \text{Velocity} < (20 * \text{Deceleration})$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	user units per second
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Maximum acceleration
	<b>Data type</b>	LREAL
	<b>Range</b>	$0 < \text{Velocity} < (20 * \text{Acceleration})$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	user units per second <sup>2</sup>

	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Maximum Deceleration
	<b>Data type</b>	LREAL
	<b>Range</b>	$0 < \text{Velocity} < (20 * \text{Deceleration})$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	user units per second <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Maximum jerk
	<b>Data type</b>	LREAL
	<b>Range</b>	<p><b>For trapezoidal velocity profiles:</b> 0</p> <p><b>For S-Curve velocity profiles:</b> <math>(\text{Velocity} / 20) &lt; \text{Acceleration} &lt; (2 * \text{Jerk})</math> and <math>(\text{Velocity} / 20) &lt; \text{Deceleration} &lt; (2 * \text{Jerk})</math></p> See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
		<div style="background-color: #f0f0f0; padding: 5px;"> <p><b>NOTE</b></p> <p>S-Curve motion is currently not supported and the <i>Jerk</i> input is currently ignored. S-Curve motion and the <i>Jerk</i> argument will be supported in a future release.</p> </div>
	<b>Unit</b>	user units per second <sup>3</sup>
	<b>Default</b>	—
<b>CoordSystem</b>	<b>Description</b>	<p>The coordinate system used when commanding the circular relative move</p> <p>Currently, only the ACS coordinate system is supported. See Coordinate Systems to learn more.</p>
	<b>Data type</b>	SINT
	<b>Range</b>	<p>One of the following enumeration values:</p> <ul style="list-style-type: none"> <li>• MC_COORDINATE_SYSTEM_ACS = 0</li> <li>• MC_COORDINATE_SYSTEM_MCS = 1</li> <li>• MC_COORDINATE_SYSTEM_PCS = 2</li> </ul>
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>BufferMode</b>	<b>Description</b>	<p>Defines the chronological sequence of the function block relative to the previous block.</p> <p>The blending modes (2, 3, 4, &amp; 5) match the <b>path velocity</b> at the active move's endpoint. Some individual <b>axis velocities</b> may make an abrupt change if the path of the next move travels in a different direction. A transition move may be programmed at the <code>TransitionMode</code> input to avoid this.</p> <p>See the table in <a href="#">Buffer Modes</a>.</p>
	<b>Data type</b>	<p>SINT</p> <p>One of the following enumeration values:</p> <ul style="list-style-type: none"> <li>• <code>MC_BUFFER_MODE_ABORTING = 0 = Abort</code></li> <li>• <code>MC_BUFFER_MODE_BUFFERED = 1 = Buffered</code></li> <li>• <code>MC_BUFFER_MODE_BLENDED_PREVIOUS = 2 = Blending Previous</code></li> <li>• <code>MC_BUFFER_MODE_BLENDED_NEXT = 3 = Blending Next</code></li> <li>• <code>MC_BUFFER_MODE_BLENDED_LOW = 4 = Blending Low</code></li> <li>• <code>MC_BUFFER_MODE_BLENDED_HIGH = 5 = Blending High</code></li> </ul>
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>TransitionMode</b>	<b>Description</b>	<p>Coupled with the <code>TransitionParameter[ ]</code>, this input defines the shape and dynamics of the inserted contour to connect the current motion with the next motion in the queue.</p> <p>See <a href="#">Transition Between Moves</a> for additional information.</p>
	<b>Data type</b>	SINT
	<b>Range</b>	<p>The value is limited to the following:</p> <ul style="list-style-type: none"> <li>• <code>MC_TRANSITION_MODE_NONE = 0</code></li> <li>• <code>MC_TRANSITION_MODE_CORNER_DISTANCE = 3</code></li> </ul>
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>TransitionParameter[ ]</b>	<b>Description</b>	This array is dependent on the TransitionMode specified. The transition parameter values are applied to the axis group. See <a href="#">table: "Transition Mode Parameters"</a> for details.
	<b>Data type</b>	LREAL
	<b>Range</b>	[1, N ] N values are supplier specified dependent on the TransitionMode selected.
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 2.3.166 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	If True, then the function block is executing.
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	If True, then the function block is controlling motion.
	<b>Data type</b>	BOOL
<b>CommandAborted</b>	<b>Description</b>	If True, command was aborted by another function block.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See table in <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data type</b>	INT

#### 2.3.166.1.1.1 Example

### 2.3.167 ST

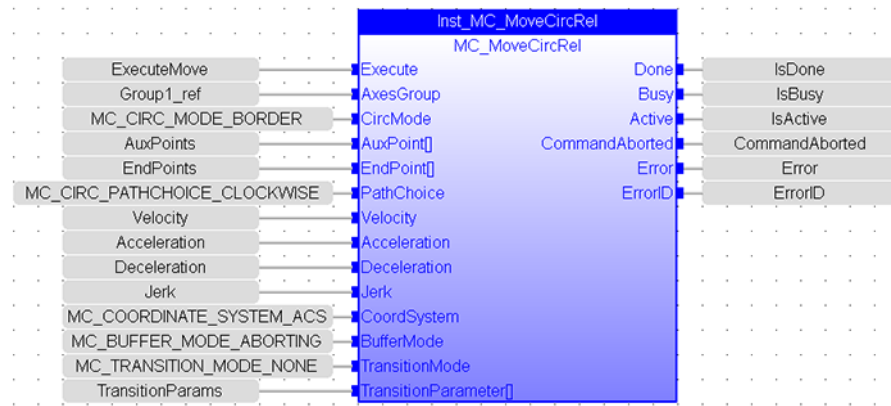
```
Inst_MC_MoveCircRel( ExecuteMove, Group1_Ref, MC_CIRC_MODE_
BORDER, AuxPoints, EndPoints, MC_CIRC_PATHCHOICE_CLOCKWISE,
Velocity, Acceleration, Deceleration, Jerk, MC_COORDINATE_
SYSTEM_ACS, MC_BUFFER_MODE_ABORTING, MC_TRANSITION_MODE_NONE,
TransitionParams );
```

### 2.3.168 IL

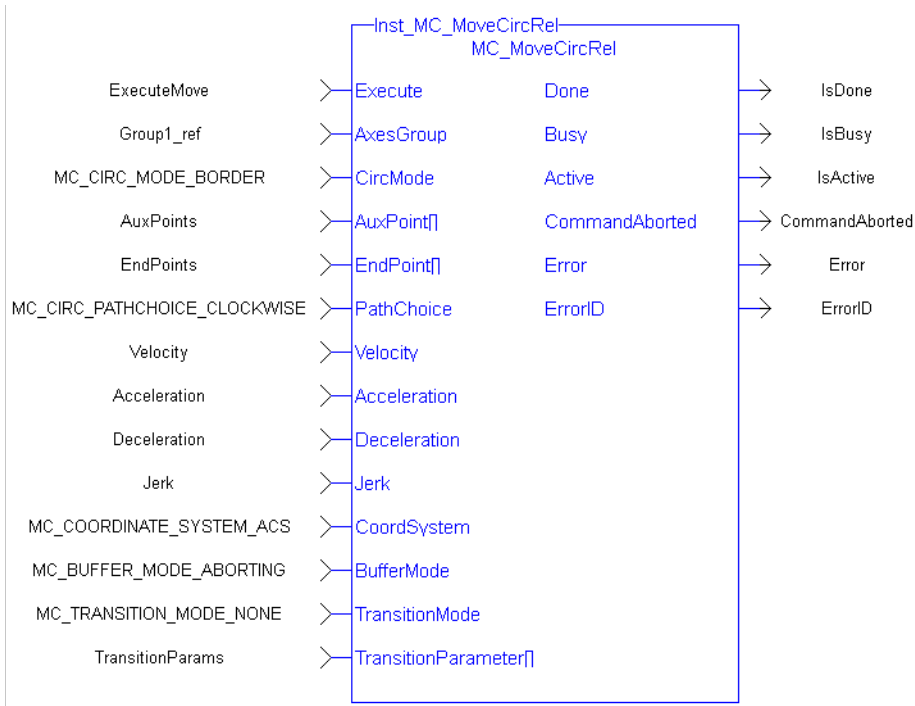
```

BEGIN_IL
CAL Inst_MC_MoveCircRel( ExecuteMove, Group1_Ref, MC_CIRC_
MODE_BORDER, AuxPoints, EndPoints, MC_CIRC_PATHCHOICE_CLOCKWISE,
Velocity, Acceleration, Deceleration, Jerk, MC_COORDINATE_
SYSTEM_ACS, MC_BUFFER_MODE_ABORTING, MC_TRANSITION_MODE_NONE,
TransitionParams )
END_IL
    
```

**2.3.169 FBD**



**2.3.170 FFLD**



**2.3.170.1.1 MC\_MoveDirAbs** PLCopen ✓ Pipe Network ✓

**2.3.170.2.2.1 Description**

MC\_MoveDirAbs commands the movement of an axes group to a specified absolute position in the specified coordinate system without taking care of how (on which path) the target position is reached.

**NOTE**

- An error is returned if the group is in the GroupDisabled state.
- This function block does not have its own Acceleration, Deceleration, Velocity, and Jerk arguments. These are set using "MC\_AxisSetDefaults" (p. 638).
- The maximum number of axes is set by the **MaxNumberOfAxes** input set in the "MC\_CreateAxesGrp" (p. 585) function block.
- S-Curve motion is not currently supported. The *Jerk* input is currently ignored. S-Curve motion and the *Jerk* argument will be supported in a future release. .

When all motion is completed successfully, the state becomes GroupStandby.

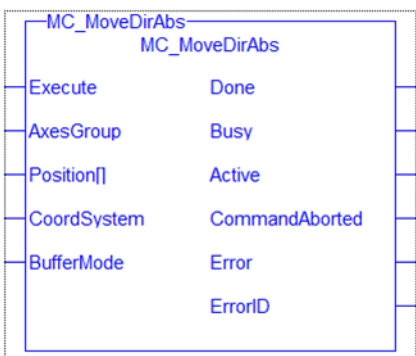


Figure 1-116: MC\_MoveDirAbs

**2.3.171 Related Functions**

"MC\_MoveDirRel" (p. 665), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

**2.3.171.1.1.1 Arguments**

**2.3.172 Input**

<b>Execute</b>	<b>Description</b>	On the rising edge, request to perform a direct absolute move.
	<b>Data Type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	Reference to an axes group
	<b>Data Type</b>	AXES_GROUP_REF
	<b>Range</b>	—
	<b>Unit</b>	n/a

	<b>Default</b>	—
<b>Position[ ]</b>	<b>Description</b>	Array of absolute end positions for each axis in the group.
	<b>Data Type</b>	LREAL
	<b>Range</b>	[0, Number of axes in group - 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>CoordSystem</b>	<b>Description</b>	The coordinate system used when commanding the direct absolute move  Currently, only the ACS coordinate system is supported. See Coordinate Systems to learn more.
	<b>Data Type</b>	SINT
	<b>Range</b>	<ul style="list-style-type: none"> <li>• MC_COORDINATE_SYSTEM_ACS = 0</li> <li>• MC_COORDINATE_SYSTEM_MCS = 1</li> <li>• MC_COORDINATE_SYSTEM_PCS = 2</li> </ul>
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>BufferMode</b>	<b>Description</b>	Defines the chronological sequence of the function block relative to the previous block.  See the table in <a href="#">Buffer Modes</a> .
	<b>Data Type</b>	SINT  MC_BUFFER_MODE_ABORTING = 0 = Abort MC_BUFFER_MODE_BUFFERED = 1 = Buffered
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 2.3.173 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	If True, then the function block is executing.
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	If True, then the function block is controlling motion.

	<b>Data type</b>	BOOL
<b>CommandAborted</b>	<b>Description</b>	If True, command was aborted by another function block.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See table in <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Data type</b>	INT

### 2.3.173.1.1.1 Example

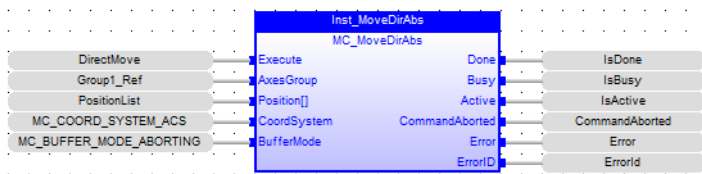
### 2.3.174 Structure Text

```
Inst_MC MoveDirAbs( DirectMove, Group1_Ref, PositionList, MC_
COORDSYSTEM_ACS, MC_BUFFER_MODE_ABORTING);
```

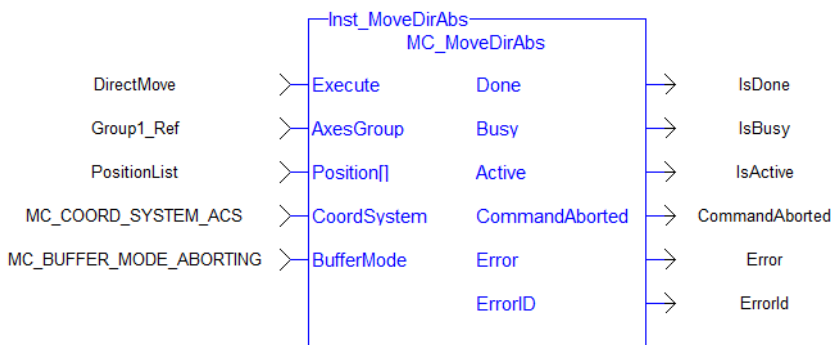
### 2.3.175 IL

```
BEGIN_IL
    CAL Inst_MC_MoveDirAbs( DirectMove, Group1_Ref,
PositionList, MC_COORD_SYSTEM_ACS, MC_BUFFER_MODE_ABORTING)
END_IL
```

### 2.3.176 Function Block Diagram



### 2.3.177 Ladder Diagram





### 2.3.177.1.1 MC\_MoveDirRel

#### 2.3.177.2.2.1 Description

MC\_MoveDirRel commands a movement of an axes group to a relative position in the specified coordinate system without taking care of how (on which path) the target position is reached.

#### NOTE

- An error is returned if the group is in the GroupDisabled state.
- This function block does not have its own Acceleration, Deceleration, Velocity, and Jerk arguments. These are set using "MC\_AxisSetDefaults" (p. 638).
- The maximum number of axes is set by the **MaxNumberOfAxes** input set in the "MC\_CreateAxesGrp" (p. 585) function block.
- S-Curve motion is not currently supported. The *Jerk* input is currently ignored. S-Curve motion and the *Jerk* argument will be supported in a future release. .

When all motion has completed successfully, the state of the axes group goes to GroupStandby.

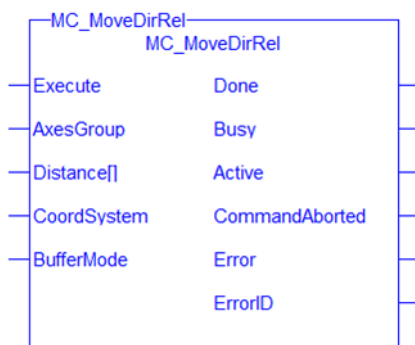


Figure 1-117: MC\_MoveDirRel

### 2.3.178 Related Functions

"MC\_MoveDirAbs" (p. 662), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

#### 2.3.178.1.1.1 Arguments

#### 2.3.179 Input

<b>Execute</b>	<b>Description</b>	On the rising edge request to perform a direct relative move.
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	Reference to an axes group
	<b>Data type</b>	AXES_GROUP_REF
	<b>Range</b>	—
	<b>Unit</b>	n/a

	<b>Default</b>	—
<b>Distance[ ]</b>	<b>Description</b>	An array containing the distance for each axis in the group.
	<b>Data type</b>	LREAL
	<b>Range</b>	[0, Number of axes in group - 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>CoordSystem</b>	<b>Description</b>	The coordinate system used when commanding the direct relative move.  Currently, only the ACS coordinate system is supported. See Coordinate Systems to learn more.
	<b>Data type</b>	SINT
	<b>Range</b>	<ul style="list-style-type: none"> <li>• MC_COORDINATE_SYSTEM_ACS = 0</li> <li>• MC_COORDINATE_SYSTEM_MCS = 1</li> <li>• MC_COORDINATE_SYSTEM_PCS = 2</li> </ul>
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>BufferMode</b>	<b>Description</b>	Defines the chronological sequence of the function block relative to the previous block.  See the table in <a href="#">Buffer Modes</a>
	<b>Data type</b>	SINT  MC_BUFFER_MODE_ABORTING = 0 = Abort MC_BUFFER_MODE_BUFFERED = 1 = Buffered
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 2.3.180 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	If True, then the function block is executing. .
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	If True, then the function block is controlling motion.
	<b>Data type</b>	BOOL

<b>CommandAborted</b>	<b>Description</b>	If True, command was aborted by another function block.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See table in <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Data type</b>	INT

### 2.3.180.1.1 Example

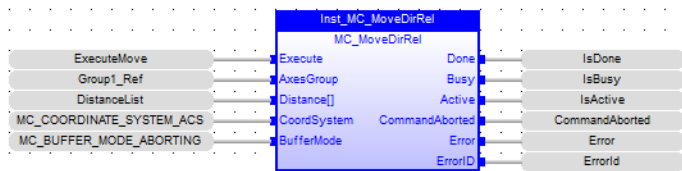
### 2.3.181 Structure Text

```
Inst_MC_MoveDirRel( ExecuteMove, Group1_Ref, DistanceList, MC_
COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_ABORTING );
```

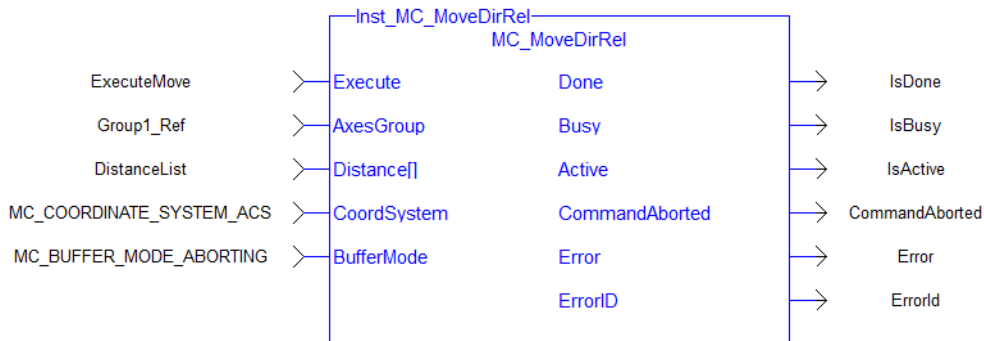
### 2.3.182 IL

```
BEGIN_IL
    CAL Inst_MC_MoveDirRel( ExecuteMove, Group1_Ref,
DistanceList, MC_COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_ABORTING
)
END_IL
```

### 2.3.183 Function Block Diagram



### 2.3.184 Ladder Diagram



**2.3.184.1.1 MC\_MoveLinAbs** PLCopen ✓ Pipe Network ✓

**2.3.184.2.2.1 Description**

MC\_MoveLinAbs commands interpolated linear movement on an axes group to the specified absolute positions in the coordinate system as specified by the 'CoordSystem' argument. The dimensionality of the move is determined by the number of axes mapped to the group.

**NOTE**

- An error is returned if the group is in the GroupDisabled state.
- The maximum number of axes is set by the **MaxNumberOfAxes** input set in the "MC\_CreateAxesGrp" (p. 585) function block.
- S-Curve motion is not currently supported. The *Jerk* input is currently ignored. S-Curve motion and the *Jerk* argument will be supported in a future release. .

When all motion has completed successfully, the state of the axes group goes to GroupStandby.

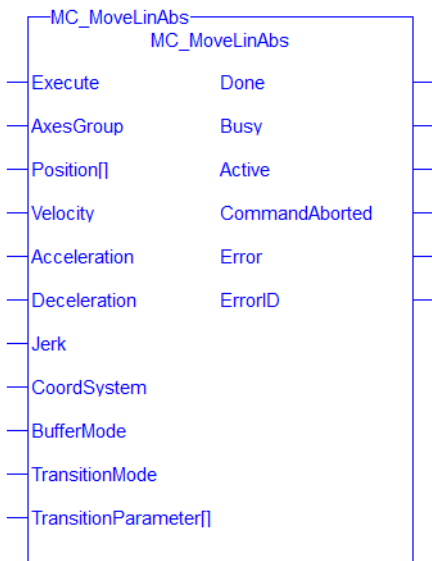


Figure 1-118: MC\_MoveLinAbs

**2.3.185 Related Functions**

"MC\_MoveLinRel" (p. 674), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

**2.3.185.1.1.1 Arguments**

**2.3.186 Input**

<b>Execute</b>	<b>Description</b>	On the rising edge request to perform a linear absolute move.
	<b>Data Type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>AxesGroup</b>	<b>Description</b>	The axis group that will perform the linear absolute move
	<b>Data Type</b>	AXIS_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Position[]</b>	<b>Description</b>	Array of absolute end positions for each axis in the group.
	<b>Data Type</b>	LREAL
	<b>Range</b>	n/a
	<b>Unit</b>	user units
	<b>Default</b>	—
<b>Velocity</b>	<b>Description</b>	Maximum velocity of the defined path
	<b>Data Type</b>	LREAL
	<b>Range</b>	$0 < \text{Velocity} < (20 * \text{Acceleration})$ and $0 < \text{Velocity} < (20 * \text{Deceleration})$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	user units per second
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Maximum acceleration
	<b>Data Type</b>	LREAL
	<b>Range</b>	$0 < \text{Velocity} < (20 * \text{Acceleration})$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	user units per second <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Maximum deceleration
	<b>Data Type</b>	LREAL
	<b>Range</b>	$0 < \text{Velocity} < (20 * \text{Deceleration})$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	user units per second <sup>2</sup>

	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Maximum jerk
	<b>Data Type</b>	LREAL
	<b>Range</b>	<p><b>For trapezoidal velocity profiles:</b> 0</p> <p><b>For S-Curve velocity profiles:</b> <math>(Velocity / 20) &lt; Acceleration &lt; (2 * Jerk)</math> and <math>(Velocity / 20) &lt; Deceleration &lt; (2 * Jerk)</math></p> <p>See <a href="#">Limitations on Acceleration and Jerk</a> for more information.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p><b>NOTE</b></p> <p>S-Curve motion is currently not supported and the <i>Jerk</i> input is currently ignored. S-Curve motion and the <i>Jerk</i> argument will be supported in a future release.</p> </div>
	<b>Unit</b>	user units per second <sup>3</sup>
	<b>Default</b>	—
<b>CoordSystem</b>	<b>Description</b>	<p>The coordinate system used when commanding the linear absolute move.</p> <p>Currently, only the ACS coordinate system is supported. See Coordinate Systems to learn more.</p>
	<b>Data Type</b>	SINT
	<b>Range</b>	<p>One of the following enumeration values:</p> <ul style="list-style-type: none"> <li>• MC_COORDINATE_SYSTEM_ACS = 0</li> <li>• MC_COORDINATE_SYSTEM_MCS = 1</li> <li>• MC_COORDINATE_SYSTEM_PCS = 2</li> </ul>
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>BufferMode</b>	<p><b>Description</b></p> <p>Defines the chronological sequence of the function block relative to the previous block.</p> <p>MC_BUFFER_MODE_ABORTING = 0 = Abort  MC_BUFFER_MODE_BUFFERED = 1 = Buffered  MC_BUFFER_MODE_BLENDED_PREVIOUS = 2 = Blending Previous  MC_BUFFER_MODE_BLENDED_NEXT = 3 = Blending Next  MC_BUFFER_MODE_BLENDED_LOW = 4 = Blending Low  MC_BUFFER_MODE_BLENDED_HIGH = 5 = Blending High</p> <p>The buffer mode is limited to MC_BUFFER_MODE_BUFFERED for groups with more than two axes.</p> <p>The blending modes (2, 3, 4, &amp; 5) match the <b>path velocity</b> at the active move's endpoint. Some individual <b>axis velocities</b> may make an abrupt change if the path of the next move travels in a different direction. A transition move may be programmed at the <code>TransitionMode</code> input to avoid this.</p> <p>See the table in <a href="#">Buffer Modes</a></p>
	<p><b>Data Type</b></p> <p>SINT</p>
	<p><b>Range</b></p> <p>—</p>
	<p><b>Unit</b></p> <p>n/a</p>
	<p><b>Default</b></p> <p>—</p>
<b>TransitionMode</b>	<p><b>Description</b></p> <p>Coupled with the <code>TransitionParameter[ ]</code>, this input defines the shape and dynamics of the inserted contour to connect the current motion with the next motion in the queue.</p> <p>See <a href="#">Transition Between Moves</a> for additional information.</p>
	<p><b>Data type</b></p> <p>SINT</p>
	<p><b>Range</b></p> <p>The value is limited to the following:</p> <ul style="list-style-type: none"> <li>• MC_TRANSITION_MODE_NONE = 0</li> <li>• MC_TRANSITION_MODE_CORNER_DISTANCE = 3</li> </ul> <p>The transition mode is limited to MC_TRANSITION_MODE_NONE for groups with more than two axes.</p>
	<p><b>Unit</b></p> <p>n/a</p>
	<p><b>Default</b></p> <p>—</p>

<b>TransitionParameter[ ]</b>	<b>Description</b>	This array is dependent on the TransitionMode specified. The transition parameter values are applied to the axis group. See <a href="#">table: "Transition Mode Parameters"</a> for details.
	<b>Data Type</b>	LREAL
	<b>Range</b>	[1, N] N values are supplier specified dependent on the TransitionMode selected.
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 2.3.187 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	If True, then the function block is executing.
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	If True, then the function block is controlling motion.
	<b>Data type</b>	BOOL
<b>CommandAborted</b>	<b>Description</b>	If True, then the command was aborted by another function block.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See table in <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data type</b>	INT

#### 2.3.187.1.1.1 Example

### 2.3.188 Structured Text

```
(* Inst_MC_MoveLinAbsST example *)
Inst_MC_MoveLinAbs( ExecuteLinearMove, Group1_Ref, PositionList,
Velocity, Acceleration, Deceleration, Jerk, CoordSystem,
BufferMode, TransitionMode, TransitionParams );
```

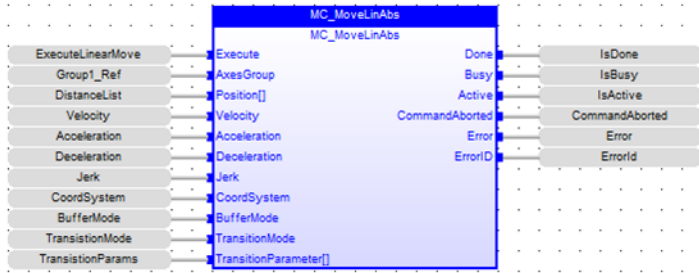
### 2.3.189 IL



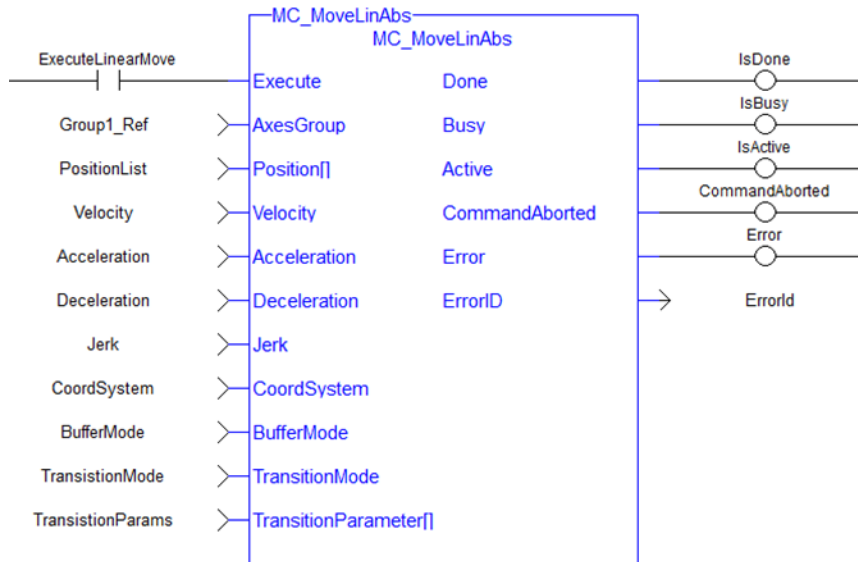
```

BEGIN_IL
CAL Inst_MC_MoveLinAbs( ExecuteLinearMove, Group1_Ref,
PositionList, Velocity, Acceleration, Deceleration, Jerk,
CoordSystem, BufferMode, TransitionMode, TransitionParams )
END_IL
    
```

### 2.3.190 FBD



### 2.3.191 FFLD



**2.3.191.1.1 MC\_MoveLinRel** PLCopen ✓ Pipe Network ✓

**2.3.191.2.2.1 Description**

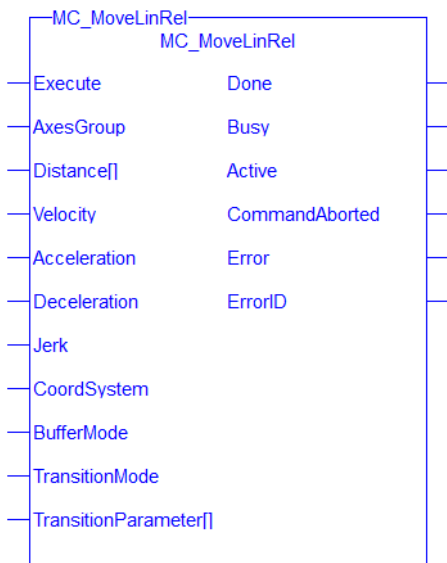
MC\_MoveLinRel commands interpolated linear movement of an axes group to the specified relative positions. The dimensionality of the move is determined by the number of axes mapped to the group.

**NOTE**

- An error is returned if the group is in the GroupDisabled state.
- The maximum number of axes is set by the **MaxNumberOfAxes** input set in the "MC\_CreateAxesGrp" (p. 585) function block.
- S-Curve motion is not currently supported. The *Jerk* input is currently ignored. S-Curve motion and the *Jerk* argument will be supported in a future release. .

When all motion has completed successfully, the state of the axes group goes to GroupStandby.

See [Transition Between Moves](#) for additional information.



**Figure 1-119:** MC\_MoveLinRel

**2.3.192 Related Functions**

"MC\_MoveLinAbs" (p. 668), "MC\_ErrorDescription" (p. 549)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

**2.3.192.1.1.1 Arguments**

**2.3.193 Input**

<b>Execute</b>	<b>Description</b>	On the rising edge request to perform a linear relative move
	<b>Data Type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>AxesGroup</b>	<b>Description</b>	The axis group that will perform the linear relative move
	<b>Data Type</b>	AXIS_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Distance[]</b>	<b>Description</b>	Array of distances for each axis in the group.
	<b>Data Type</b>	LREAL
	<b>Range</b>	n/a
	<b>Unit</b>	user units
	<b>Default</b>	—
<b>Velocity</b>	<b>Description</b>	Maximum velocity of the defined path
	<b>Data Type</b>	LREAL
	<b>Range</b>	$0 < \text{Velocity} < (20 * \text{Acceleration})$ and $0 < \text{Velocity} < (20 * \text{Deceleration})$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	user units per second
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Maximum acceleration
	<b>Data Type</b>	LREAL
	<b>Range</b>	$0 < \text{Velocity} < (20 * \text{Acceleration})$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	user units per second <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Maximum deceleration
	<b>Data Type</b>	LREAL
	<b>Range</b>	$0 < \text{Velocity} < (20 * \text{Deceleration})$ See <a href="#">Limitations on Acceleration and Jerk</a> for more information.
	<b>Unit</b>	user units per second <sup>2</sup>

	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Maximum jerk
	<b>Data Type</b>	LREAL
	<b>Range</b>	<p><b>For trapezoidal velocity profiles:</b> 0</p> <p><b>For S-Curve velocity profiles:</b> <math>(\text{Velocity} / 20) &lt; \text{Acceleration} &lt; (2 * \text{Jerk})</math> and <math>(\text{Velocity} / 20) &lt; \text{Deceleration} &lt; (2 * \text{Jerk})</math></p> <p>See <a href="#">Limitations on Acceleration and Jerk</a> for more information.</p> <p><b>NOTE</b></p> <p>S-Curve motion is currently not supported and the <i>Jerk</i> input is currently ignored. S-Curve motion and the <i>Jerk</i> argument will be supported in a future release.</p>
	<b>Unit</b>	user units per second <sup>3</sup>
	<b>Default</b>	—
<b>CoordSystem</b>	<b>Description</b>	<p>The coordinate system used when commanding the linear relative move</p> <p>Currently, only the ACS coordinate system is supported. See Coordinate Systems to learn more.</p>
	<b>Data Type</b>	SINT
	<b>Range</b>	<p>One of the following enumeration values:</p> <ul style="list-style-type: none"> <li>• MC_COORDINATE_SYSTEM_ACS = 0</li> <li>• MC_COORDINATE_SYSTEM_MCS = 1</li> <li>• MC_COORDINATE_SYSTEM_PCS = 2</li> </ul>
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>BufferMode</b>	<p><b>Description</b></p> <p>Defines the chronological sequence of the function block relative to the previous block.</p> <p>MC_BUFFER_MODE_ABORTING = 0 = Abort  MC_BUFFER_MODE_BUFFERED = 1 = Buffered  MC_BUFFER_MODE_BLENDED_PREVIOUS = 2 = Blending Previous  MC_BUFFER_MODE_BLENDED_NEXT = 3 = Blending Next  MC_BUFFER_MODE_BLENDED_LOW = 4 = Blending Low  MC_BUFFER_MODE_BLENDED_HIGH = 5 = Blending High</p> <p>The buffer mode is limited to MC_BUFFER_MODE_BUFFERED for groups with more than two axes.</p> <p>The blending modes (2, 3, 4, &amp; 5) match the <b>path velocity</b> at the active move's endpoint. Some individual <b>axis velocities</b> may make an abrupt change if the path of the next move travels in a different direction. A transition move may be programmed at the <code>TransitionMode</code> input to avoid this.</p> <p>See the table in <a href="#">Buffer Modes</a></p>
	<p><b>Data Type</b></p> <p>SINT</p>
	<p><b>Range</b></p> <p>—</p>
	<p><b>Unit</b></p> <p>n/a</p>
	<p><b>Default</b></p> <p>—</p>
<b>TransitionMode</b>	<p><b>Description</b></p> <p>Coupled with the <code>TransitionParameter[ ]</code>, this input defines the shape and dynamics of the inserted contour to connect the current motion with the next motion in the queue.</p> <p>See <a href="#">Transition Between Moves</a> for additional information.</p>
	<p><b>Data Type</b></p> <p>SINT</p>
	<p><b>Range</b></p> <p>The value is limited to the following:</p> <ul style="list-style-type: none"> <li>• MC_TRANSITION_MODE_NONE = 0</li> <li>• MC_TRANSITION_MODE_CORNER_DISTANCE = 3</li> </ul> <p>The transition mode is limited to MC_TRANSITION_MODE_NONE for groups with more than two axes.</p>
	<p><b>Unit</b></p> <p>n/a</p>
	<p><b>Default</b></p> <p>—</p>

<b>TransitionParameter[ ]</b>	<b>Description</b>	This array is dependent on the TransitionMode specified. The transition parameter values are applied to the axis group. See <a href="#">table: "Transition Mode Parameters"</a> for details.
	<b>Data Type</b>	LREAL
	<b>Range</b>	[0, N] The value of N is dependent on the TransitionMode specified.
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 2.3.194 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data type</b>	BOOL
<b>Busy</b>	<b>Description</b>	If True, then the function block is executing.
	<b>Data type</b>	BOOL
<b>Active</b>	<b>Description</b>	If True, then the function block is still controlling motion.
	<b>Data type</b>	BOOL
<b>CommandAborted</b>	<b>Description</b>	If True, then the command was aborted by another function block.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, then an error has occurred
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. See table in <a href="#">PLCopen Function Block ErrorID Output</a>
	<b>Data type</b>	INT

#### 2.3.194.1.1.1 Example

### 2.3.195 Structured Text

```
(* Inst_MC_MoveLinRelST example *)
```

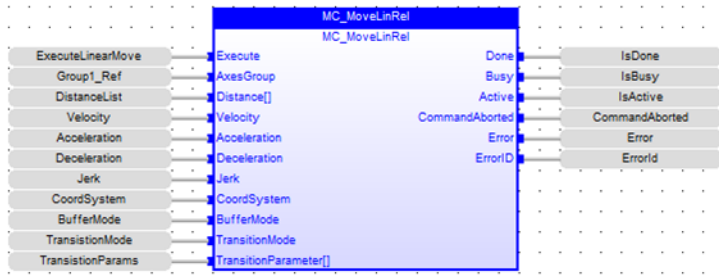
```
Inst_MC_MoveLinRel( ExecuteLinearMove, Group1_Ref, DistanceList,
Velocity, Acceleration, Deceleration, Jerk, CoordSystem,
BufferMode, TranstionMode, TransitionParams );
```

### 2.3.196 IL

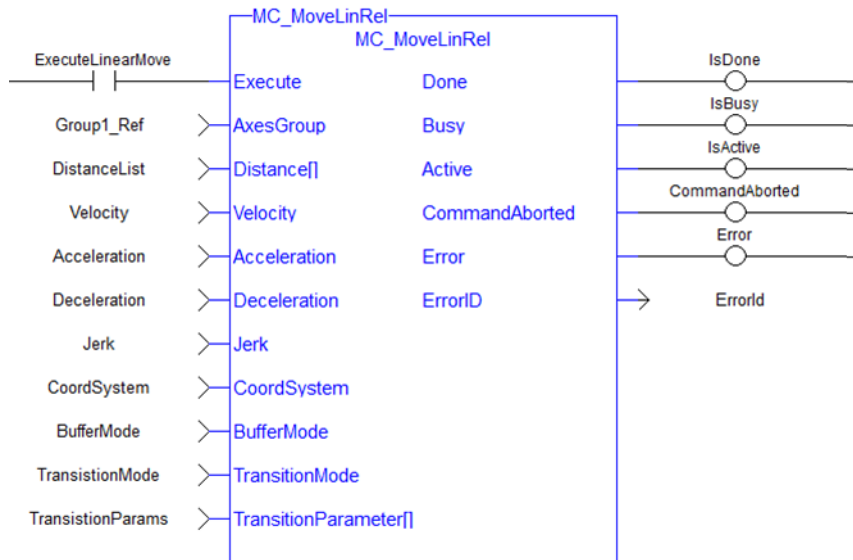
```

BEGIN_IL
    CAL Inst_MC_MoveLinRel( ExecuteLinearMove, Group1_Ref,
        DistanceList, Velocity, Acceleration, Deceleration, Jerk,
        CoordSystem, BufferMode, TransitionMode, TransitionParams )
END_IL
    
```

### 2.3.197 FBD



### 2.3.198 FFLD



### 2.3.198.1 Coordinated Motion Reference Library

Function	Description
"MC_GrpSetPos" (p. 680)	Sets the position of the group.

#### 2.3.198.2.1 MC\_GrpSetPos PLCopen ✓ Pipe Network ✓

##### 2.3.198.3.2.1 Description

MC\_GrpSetPos sets the axis command position for all of the axes in an axes group to the positions specified in the `Position` input. This function block does not cause any motion. The axes group must be enabled and in Standby mode for MC\_GrpSetPos to execute. If it is not, this FB will return an error and the axis positions will remain unchanged. The command position is that returned by the Function Block MC\_GrpReadCmdPos.

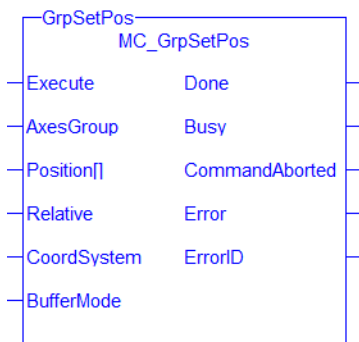


Figure 1-120: MC\_GrpSetPos

### 2.3.199 Related Functions

"MC\_ErrorDescription" (p. 549), "MC\_GrpReadCmdPos" (p. 626)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

#### 2.3.199.1.1.1 Arguments

#### 2.3.200 Input

<b>Execute</b>	<b>Description</b>	On the rising edge, request to set the position of the group
	<b>Data Type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxesGroup</b>	<b>Description</b>	The axis group for which to set the positions
	<b>Data Type</b>	AXIS_GROUP_REF
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—



<b>Position[]</b>	<b>Description</b>	An array containing the position for each axis in the group. If "Relative" is set, position represents a distance rather than an absolute position. The length of the array must equal the maximum number of axes allowed in the group. The maximum number of axes is an argument to MC_CreateAxesGrp, which is used to create axes groups.
	<b>Data Type</b>	LREAL
	<b>Range</b>	[0, Number of axes in group-1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Relative</b>	<b>Description</b>	Request to set relative (1) or absolute (0) position
	<b>Data Type</b>	BOOL
	<b>Range</b>	1, 0
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>CoordSystem</b>	<b>Description</b>	The coordinate system used when setting the positions.
	<b>Data Type</b>	SINT
	<b>Range</b>	One of the following enumeration values: <ul style="list-style-type: none"> <li>• MC_COORDINATE_SYSTEM_ACS = 0</li> <li>• MC_COORDINATE_SYSTEM_MCS = 1</li> <li>• MC_COORDINATE_SYSTEM_PCS = 2</li> </ul> <p>Currently, only the ACS coordinate system is supported. See <a href="#">Coordinate Systems</a> for more information.</p>
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>BufferMode</b>	<b>Description</b>	Currently unused
	<b>Data Type</b>	SINT
	<b>Range</b>	[0, 0]
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 2.3.201 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data Type</b>	BOOL

<b>Busy</b>	<b>Description</b>	Currently unused, returns FALSE
	<b>Data Type</b>	BOOL
<b>CommandAborted</b>	<b>Description</b>	Currently unused, returns FALSE
	<b>Data Type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data Type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error identifier if Error output is set to TRUE. See the table in <a href="#">PLCopen Function Block ErrorID Output</a> .
	<b>Data Type</b>	INT

### 2.3.201.1.1.1 Example

### 2.3.202 ST

```
Inst_MC_GrpSetPos( DoSetPos, Group1, PositionArray, Relative,
MC_COORDINATE_SYSTEM_ACS, 0 );
```

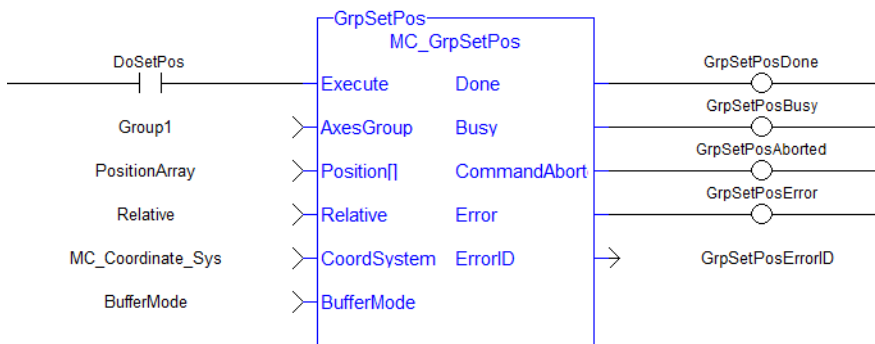
### 2.3.203 FBD



### 2.3.204 IL

```
BEGIN_IL
    CAL Inst_MC_GrpSetPos( DoSetPos, Group1, PositionArray,
Relative, MC_COORDINATE_SYSTEM_ACS, BufferMode);
END_IL
```

### 2.3.205 FFLD



### 3 Fieldbus Library

---

3.1 EtherCAT Library .....	684
3.2 Profibus Library .....	723

### 3.1 EtherCAT Library

Name	Object Type	Description
<a href="#">DriveParamRead</a>	SDO	Reads a drive parameter (ASCII format)
<a href="#">DriveParamWrite</a>	SDO	Writes a drive parameter (ASCII format)
" <a href="#">ECATCommErrors</a> " (p. 708)		Returns a list of bad EtherCAT connections
" <a href="#">ECATDeviceStatus</a> " (p. 711)		Provides EtherCAT state and port link status information for an EtherCAT device.
<a href="#">ECATGetObjVal</a>	PDO	Reads cyclic drive parameter (String format) by returning the value of an EtherCAT PDO element
<a href="#">ECATGetStatus</a>	PDO	Reads cyclic status word (Index 6041)
" <a href="#">ECATMasterStatus</a> " (p. 717)		Reads the EtherCAT master state and the lost frame counter
<a href="#">ECATReadData</a>	PDO	Reads cyclic parameter (byte offset format)
<a href="#">ECATReadSdo</a>	SDO	Reads parameter (32 bit format) using SDO command
<a href="#">ECATSetControl</a>	PDO	Manipulates the state of a drive by setting its control word (Index 6040)
" <a href="#">ECATWCStatus</a> " (p. 721)		Returns the current number of working counter errors for the Sync unit
<a href="#">ECATWriteData</a>	PDO	Writes cyclic parameter (byte offset format)
<a href="#">ECATWriteSdo</a>	SDO	Writes parameter (32 bit format) using SDO command

Table 1-4: 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 AKC (Celeron 1.2GHz)	High-range AKC (Core 2 Duo 1.86GHz)
Mean	60 µs	30 µs
Min	48 µs	24 µs

	Mid-range AKC (Celeron 1.2GHz)	High-range AKC (Core 2 Duo 1.86GHz)
Max	64 $\mu$ s	38 $\mu$ s

(these values have been computed with the TraceTimes command)

### 3.1.1 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.

#### 3.1.1.1.1 Execution Time

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

#### NOTE

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

#### 3.1.1.2.2.1 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.

#### 3.1.1.3.3.2 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"*

#### 3.1.1.4.4.3 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

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

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

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

	4 kHz	1 kHz
Mean	<b>20 ms</b>	<b>11 ms</b>
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 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.

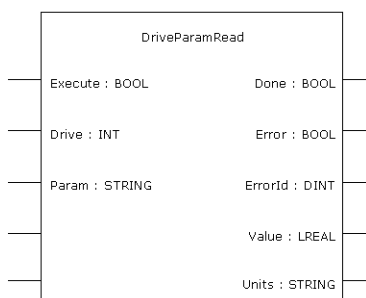
**3.1.1.5 DriveParamRead** PLCopen ✓ Pipe Network ✓

**3.1.1.6.1 Description**

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

**It takes multiple cycles to complete this function block.** Typically only *one* DriveParamRead or "DriveParamWrite" (p. 691) function should be active for *each axis* at one time. If executing this function block continuously or multiple times is required, add code that waits for this function block to complete (Done bit = 1) before executing it again, as shown in the example below.

See also some **stats** about the execution time [here](#).



**Figure 1-121:** DriveParamRead

**NOTE**

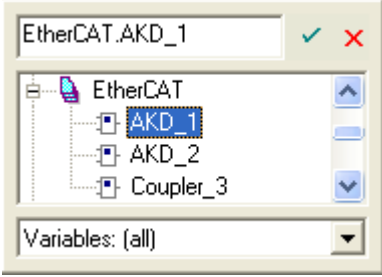
This function block uses *and reserves* the EtherCAT SDO Channel. The SDO Channel will remain reserved until the done output is "true". Therefore, this FB should be called at each cycle until the done output is true. If it is not called at each cycle the rest of SDO communication (the AKD GUI Views, 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.

**3.1.1.7.2 Arguments**

**3.1.1.8.3.1 Input**

<b>Execute</b>	<p><b>Description</b></p> <p>On the rising edge of Execute, a drive parameter is read. 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.</p>
	<p><b>Data type</b></p> <p>BOOL</p>
	<p><b>Range</b></p> <p>0, 1</p>
	<p><b>Unit</b></p> <p>n/a</p>
	<p><b>Default</b></p> <p>—</p>

<b>Drive</b>	<b>Description</b>	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 <a href="#">create the variable</a> .
		
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
<b>Param</b>	<b>Description</b>	The parameter to read.
	<b>Data type</b>	STRING
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 3.1.1.9.4.2 Output

<b>Done</b>	<b>Description</b>	Indicates whether the DriveParamRead function block has completed without error.
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Error</b>	<b>Description</b>	Indicates whether the DriveParamRead function block call has completed with error:
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>ErrorID</b>	<b>Description</b>	The DriveParamRead error result if Error is TRUE (see list of Error Codes in table below). Upon success, Error is set to zero.
	<b>Data type</b>	DINT
	<b>Unit</b>	n/a

<b>Value</b>	<b>Description</b>	The value of the drive parameter. Value is only set when the function block has successfully completed.
	<b>Data type</b>	LREAL
	<b>Unit</b>	n/a
<b>Units</b>	<b>Description</b>	The units of the drive parameter. Value is only set when the function block has successfully completed.
	<b>Data type</b>	STRING
	<b>Unit</b>	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
ECERR_DEVICE_INVALIDSIZE	1797 (0x705)	An invalid size for the parameter was specified
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_INVALIDOFFSET	1827 (0x723)	An invalid value for the SubIndex input was specified
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_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



Error Code	Value, dec (hex)	Description
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 1-5: List of EtherCAT Error Codes

### 3.1.1.10.5 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.

### 3.1.1.11.6 Related Functions

[DriveParamWrite](#)

### 3.1.1.12.7 Example

#### 3.1.1.13.8.1 Structured Text

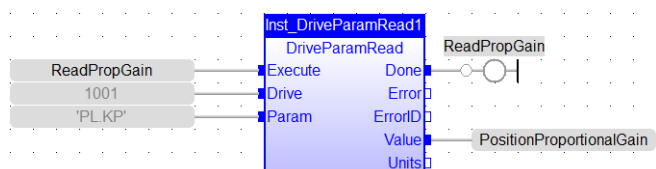
```
(* Read PL.KP on first AKD Drive on EtherCAT network *)

(* 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 *)

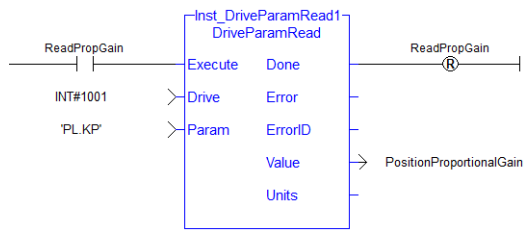
IF ReadPropGain then
  Inst_DriveParamRead1( 1, 1001, 'PL.KP' );
End_If;

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;
```

### 3.1.1.14.9.2 FBD



### 3.1.1.15.10.3 FFLD



### 3.1.1.16 DriveParamWrite PLCopen ✓ Pipe Network ✓

#### 3.1.1.17.1 Description

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

**It takes multiple cycles to complete this function block.** Typically only *one* DriveParamRead or DriveParamWrite function should be active for *each axis* at one time. If executing this function block continuously or multiple times is required, add code that waits for this function block to complete (Done bit = 1) before executing it again, as shown in DriveParamRead.

See also some **stats** about the execution time [here](#).

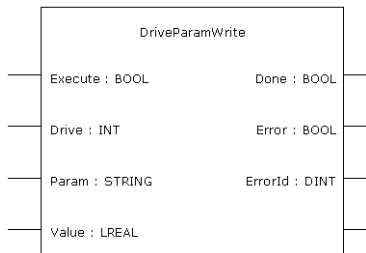


Figure 1-122: DriveParamWrite

#### NOTE

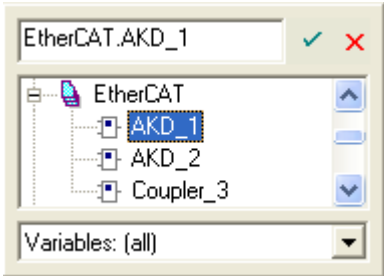
This function block uses *and reserves* the EtherCAT SDO Channel. The SDO Channel will remain reserved until the done output is "true". Therefore, this FB should be called at each cycle until the done output is true. If it is not called at each cycle the rest of SDO communication (the AKD GUI Views, 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.

#### 3.1.1.18.2 Arguments

##### 3.1.1.19.3.1 Input

Execute	Description	
		On the rising edge of Execute, a drive parameter is set. 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.
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>Drive</b>	<b>Description</b>	<p>The address of the drive to which data is written. 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 <a href="#">create the variable</a>.</p>
		
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
<b>Default</b>	—	
<b>Param</b>	<b>Description</b>	The parameter to write.
	<b>Data type</b>	STRING
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Value</b>	<b>Description</b>	The value to set the drive parameter to.
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 3.1.1.20.4.2 Output

<b>Done</b>	<b>Description</b>	Indicates whether the DriveParamWrite function block has completed without error.
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Error</b>	<b>Description</b>	Indicates whether the DriveParamWrite function block call has completed with error.

	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>ErrorID</b>	<b>Description</b>	The DriveParamWrite error result if Error is TRUE (see <a href="#">List of EtherCAT Error Codes (→ p. 689)</a> ) Upon success, Error is set to zero.
	<b>Data type</b>	DINT
	<b>Unit</b>	n/a

### 3.1.1.21.5 Usage

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

### 3.1.1.22.6 Related Functions

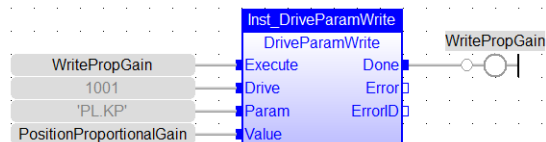
[DriveParamRead](#)

### 3.1.1.23.7 Example

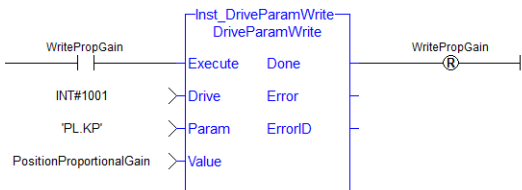
#### 3.1.1.24.8.1 Structured Text

```
(* Write 58.000 to PL.KP of first AKD Drive on EtherCAT network *)
Inst_DriveParamWrite( TRUE, 1001, 'PL.KP', 58 );
```

### 3.1.1.25.9.2 FBD



### 3.1.1.26.10.3 FFLD



### 3.1.2 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

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

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

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

	4 kHz	1 kHz
Mean	<b>20 ms</b>	<b>11 ms</b>
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 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.

#### 3.1.2.1 ECATReadSDO PLCopen ✓ Pipe Network ✓

##### 3.1.2.2.1 Description

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

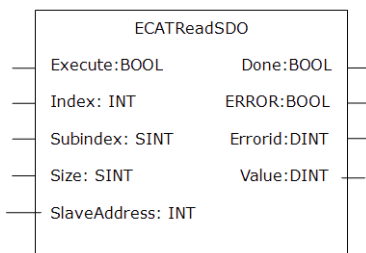


Figure 1-123: ECATReadSdo

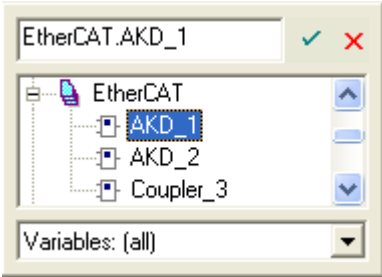
[Manufacturer specific SDOs](#), [Object Dictionary](#)

##### 3.1.2.3.2.1 State Diagram



	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Index</b>	<b>Description</b>	<p>The object directory index of the data to be read.</p> <p>For more details, refer to:</p> <ul style="list-style-type: none"> <li>• Communication SDOs</li> <li>• Manufacturer specific SDOs</li> <li>• Profile specific SDOs</li> </ul> <p>To read/write an SDO object with an index greater than 16#7FFF (32767), the value must be entered in the form <code>any_to_int(index # in hex format)</code>. For example <code>any_to_int(16#8321)</code>.</p>
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Subindex</b>	<b>Description</b>	<p>The sub-index of the object directory variable to be read.</p> <p>For more details, refer to:</p> <ul style="list-style-type: none"> <li>• Communication SDOs</li> <li>• Manufacturer specific SDOs</li> <li>• Profile specific SDOs</li> </ul> <p>To read/write an SDO object with an index greater than 16#7FFF (32767), the value must be entered in the form <code>any_to_int(index # in hex format)</code>. For example <code>any_to_int(16#8321)</code>.</p>
	<b>Data type</b>	SINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Size</b>	<b>Description</b>	The size (number of bytes) to write.
	<b>Data type</b>	SINT
	<b>Range</b>	1 - 4
	<b>Unit</b>	n/a
	<b>Default</b>	—



<b>SlaveAddress</b>	<b>Description</b>	<p>The EtherCAT address of the slave from which data is written to. 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 <a href="#">create the variable</a>.</p>
		
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 3.1.2.6.5.2 Output

<b>Done</b>	<b>Description</b>	Indicates whether the SDO call has completed without error.
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Error</b>	<b>Description</b>	Indicates whether the SDO call has completed with error:
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>ErrorID</b>	<b>Description</b>	The SDO call error result, if Error is TRUE (see list of Error Codes in table below). Upon success, Error is set to zero.
	<b>Data type</b>	DINT
	<b>Unit</b>	n/a
<b>Value</b>	<b>Description</b>	The value of the object directory variable being read. Value is only set when an SDO read command has successfully completed.
	<b>Data type</b>	DINT
	<b>Unit</b>	n/a

Table 1-6: List of EtherCAT Error Codes

### 3.1.2.7.6 Related Functions

ECATWriteSDO

**3.1.2.8.7 Example**

**3.1.2.9.8.1 Structured Text**

```

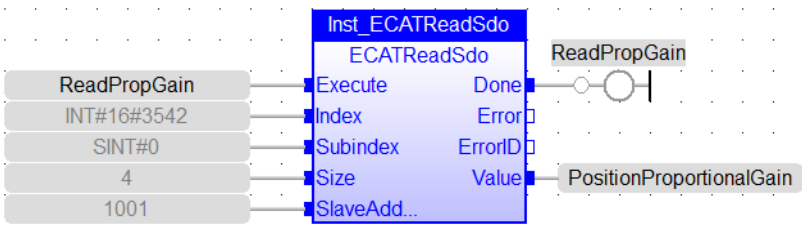
(* Read PL.KP on first AKD Dr
ive on EtherC

AT network *)
Inst_ECATReadSdo( TRUE, 16#3542, 0, 4, 1001 );
PositionProportionalGain := Inst_ECATReadSdo.Value;
    
```

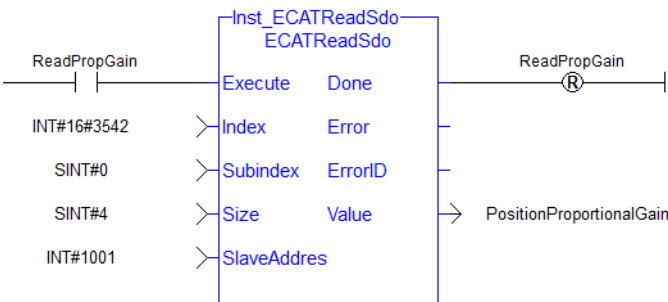
```

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

**3.1.2.10.9.2 FBD**



**3.1.2.11.10.3 FFLD**





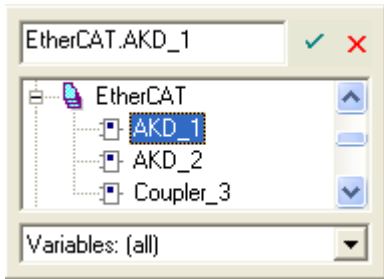
**NOTE**

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.

**3.1.2.15.3 Arguments****3.1.2.16.4.1 Input**

<b>Execute</b>	<b>Description</b>	On the rising edge of Execute, an SDO write command will be issued. 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.
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Index</b>	<b>Description</b>	The object directory index of the data to be written to. For more details, refer to: <ul style="list-style-type: none"> <li>• Communication SDOs</li> <li>• Manufacturer specific SDOs</li> <li>• Profile specific SDOs</li> </ul> To read/write an SDO object with an index greater than 16#7FFF (32767), the value must be entered in the form <code>any_to_int(index # in hex format)</code> . For example <code>any_to_int(16#8321)</code> .
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Subindex</b>	<b>Description</b>	The sub-index of the object directory variable to be written to. For more details, refer to: <ul style="list-style-type: none"> <li>• Communication SDOs</li> <li>• Manufacturer specific SDOs</li> <li>• Profile specific SDOs</li> </ul> To read/write an SDO object with an index greater than 16#7FFF (32767), the value must be entered in the form <code>any_to_int(index # in hex format)</code> . For example <code>any_to_int(16#8321)</code> .

	<b>Data type</b>	SINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Size</b>	<b>Description</b>	The size (number of bytes) to write.
	<b>Data type</b>	SINT
	<b>Range</b>	1 - 4
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>SlaveAddress</b>	<b>Description</b>	<p>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'.</p> <p>Alternately, you can use the members of the EtherCAT structure to specify a drive's address when you <a href="#">create the variable</a>.</p>
		 <p>The screenshot shows a software interface for editing a variable named 'EtherCAT.AKD_1'. It features a tree view under the 'EtherCAT' folder, with 'AKD_1' selected. Other items in the tree are 'AKD_2' and 'Coupler_3'. A 'Variables: (all)' dropdown is visible at the bottom of the window.</p>
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Value</b>	<b>Description</b>	The value to write to the object directory variable.
	<b>Data type</b>	DINT
	<b>Range</b>	[-2147483648, 2147483648]
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 3.1.2.17.5.2 Output

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

Table 1-7: List of EtherCAT Error Codes

### 3.1.2.18.6 Related Functions

#### [ECATReadSDO](#)

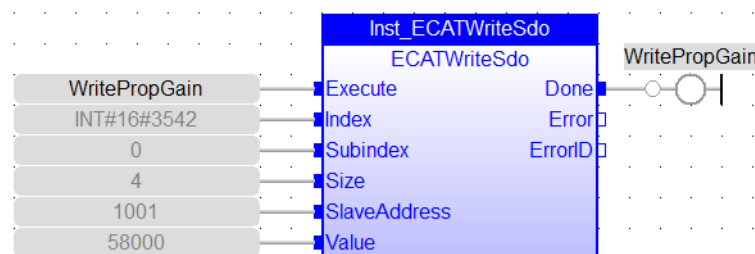
### 3.1.2.19.7 Example

#### 3.1.2.20.8.1 Structured Text

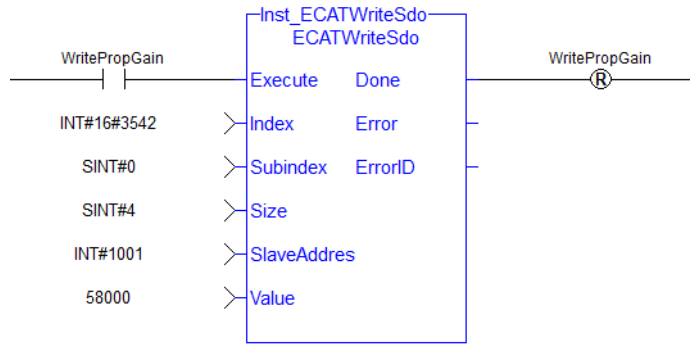
```
(* Write 58.000 to PL.KP of first AKD Drive on EtherCAT network *)
Inst_ECATWriteSdo( TRUE, 16#3542, 0, 4, 1001, 58000 );
```

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

### 3.1.2.21.9.2 FBD



### 3.1.2.22.10.3 FFLD



### 3.1.3 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.

#### 3.1.3.1 ECATGetObjVal

##### NOTE

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

#### 3.1.3.2 ECATReadData

PLCopen ✓

Pipe Network ✓

##### ⓘ IMPORTANT

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

##### 3.1.3.3.1 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).

##### 3.1.3.4.2 Arguments

##### 3.1.3.5.3.1 Input

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



<b>Unit</b>	n/a
<b>Default</b>	—

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

**3.1.3.6.4.2 Output**

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

**3.1.3.7.5 Related Functions**

[ECATGetObjVal](#)

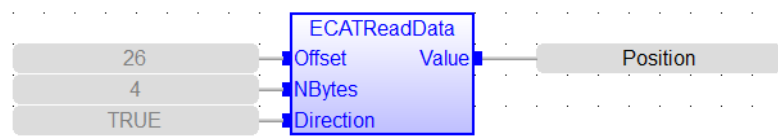
**3.1.3.8.6 Example**

**3.1.3.9.7.1 Structured Text**

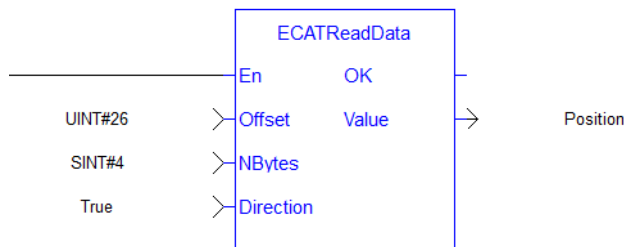
```
// Read 4 bytes starting at offset 26 of the output image
```

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

**3.1.3.10.8.2 FBD**



**3.1.3.11.9.3 FFLD**



### 3.1.3.12 ECATWriteData

#### **IMPORTANT**

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

#### 3.1.3.13.1 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.

#### 3.1.3.14.2 Arguments

##### 3.1.3.15.3.1 Input

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

#### **NOTE**

The valid ranges for the **Value** parameter are:

For 1 byte: 0 to 255

For 2 bytes: 0 to 65535

For 4 bytes: - 2147483648 to 2147483648 (The sign bit represents the most significant bit in the data word)

#### 3.1.3.16.4.2 Output

<b>Default (.Q)</b>	<b>Description</b>	True if data was written
---------------------	--------------------	--------------------------

<b>Data type</b>	BOOL
<b>Unit</b>	n/a

**3.1.3.17.5 Related Functions**

[ECATReadData](#)

**3.1.3.18.6 Example**

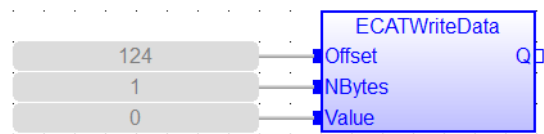
**3.1.3.19.7.1 Structured Text**

```

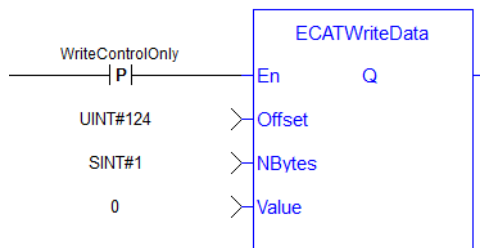
//For use with Kollmorgen Thermocouple slice I/O to read in deg
C
//Lookup offset by exporting ENI file after EtherCAT network is
scanned
//Use offst 124 (byte) to write 0 in control word to allow
temperature to be shown on status byte

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

**3.1.3.20.8.2 FBD**



**3.1.3.21.9.3 FFLD**



### 3.1.4 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.

#### 3.1.4.1 ECATCommErrors PLCopen

##### 3.1.4.2.1 Description

This function block returns a list of bad EtherCAT connections. If EtherCAT network communication is shutdown, the failed connections are based on information that was taken at the time the network was shutdown.

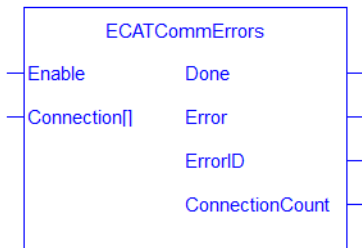


Figure 1-127: ECATCommErrors function block

##### ECATCommErr\_ref Structure

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

##### EtherCAT Port Number Defines

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

#### 3.1.4.3.2 Arguments

##### 3.1.4.4.3.1 Input

<b>Execute</b>	<b>Description</b>	Read the communication errors on the rising edge
	<b>Data type</b>	BOOL
	<b>Range</b>	0,1
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>Connection</b>	<b>Description</b>	Array of bad connections. The safe size for the list is $[2 * (\text{number of devices}) - 1]$ .
	<b>Data type</b>	ECATCommErr_ref (see "ECATCommErr_ref Structure" (p. 708) table above)
	<b>Range</b>	N= 0 to 2 times the number of EtherCAT devices.
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 3.1.4.5.4.2 Output

<b>Done</b>	<b>Description</b>	Indicates when the function is complete
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Error</b>	<b>Description</b>	Indicates the function failed due to an error.
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>ErrorID</b>	<b>Description</b>	The function call error result, if Error is TRUE (see "Possible Error Codes and Descriptions" (p. 709) table below). Upon success, Error is set to zero..
	<b>Data type</b>	DINT
	<b>Unit</b>	n/a
<b>ConnectionCount</b>	<b>Description</b>	The number of bad connections. Valid indices in the Connection array range from zero to ConnectionCount -1 (assuming that ConnectionCount != 0).
	<b>Data type</b>	UINT
	<b>Unit</b>	n/a

#### Possible Error Codes and Descriptions

Error Code	Description
<b>ECERR_DEVICE_ERROR</b>	The EtherCAT driver is in a bad state
<b>ECERR_FEATURE_NOT_AVAILABLE</b>	The feature is not available on the PAC
<b>ECERR_INVALID_ARRAY_SIZE</b>	The size of the Connections is too small

#### NOTE

When the array size is smaller than the number of bad connections, the array is filled with the data to the extent of the size of the array and the error 'ECERR\_INVALID\_ARRAY\_SIZE' is also set. In this scenario, the output **ConnectionCount** is set to the size of the array and it is smaller than the number of actual bad connections.

#### 3.1.4.6.5 Related Functions

- "ECATDeviceStatus" (p. 711)
- "ECATMasterStatus" (p. 717)

### 3.1.4.7.6 Example



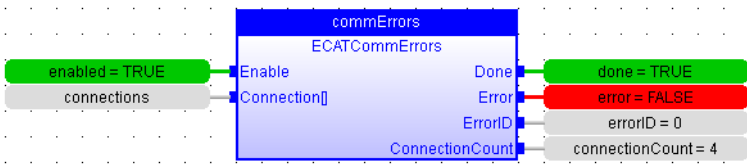
See [Checking the Connections for Errors](#): in the [EtherCAT Communication Diagnosis Steps](#) section of the Troubleshooting chapter for an example of implementing this function.

### 3.1.4.8.7.1 Structured Text

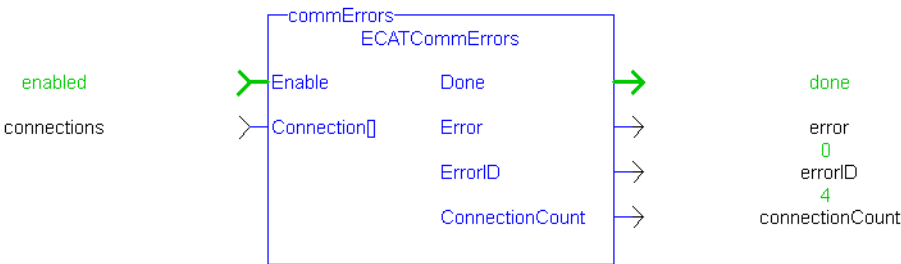
```

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

### 3.1.4.9.8.2 FBD



### 3.1.4.10.9.3 FFLD



3.1.4.11 ECATDeviceStatus PLCopen ✓ Pipe Network ✓

3.1.4.12.1 Description

This function block provides the EtherCAT state and the port link status information for the EtherCAT device. If the EtherCAT network communication is not running due to a shutdown, the device status contains information that was taken at the time the network was shutdown. This function block is useful in locating the device(s) with communication errors when the "ECATWCStatus" (p. 721) function indicates there are EtherCAT communication errors.

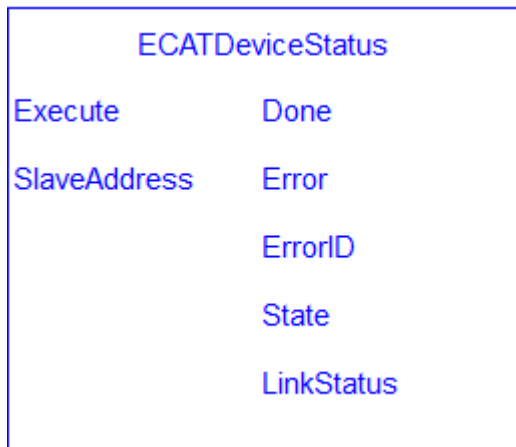


Figure 1-128: ECATDeviceStatus function block

3.1.4.13.2 Arguments

3.1.4.14.3.1 Input

<b>Execute</b>	<b>Description</b>	Read the device status on the rising edge
	<b>Data type</b>	BOOL
	<b>Range</b>	0,1
	<b>Unit</b>	n/a
	<b>Default</b>	
<b>SlaveAddress</b>	<b>Description</b>	The address of the device 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 device's address when you create the variable.
	<b>Data type</b>	INT
	<b>Range</b>	-
	<b>Unit</b>	n/a
	<b>Default</b>	-

3.1.4.15.4.2 Output

<b>Done</b>	<b>Description</b>	Indicates when the function is complete
-------------	--------------------	---

	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>State</b>	<b>Description</b>	<p>Indicates the EtherCAT state of the device. See <a href="#">"State Defines"</a> (p. 712) below for details.</p> <ul style="list-style-type: none"> <li>• A value of zero indicates that there is no communication with the device and the state is unknown.</li> <li>• Bits 3:0 indicate the actual state of the Device.</li> <li>• An EC_STATE_ERROR (bit 4 set to 1) indicates the device is not in the EtherCAT Master requested State due to error conditions such as loss of communication..</li> </ul>
	<b>Data type</b>	UINT
	<b>Unit</b>	n/a
<b>LinkStatus</b>	<b>Description</b>	<p>Provides the physical link status of the device's ports. See <a href="#">"LinkStatus Defines"</a> (p. 713) below.</p> <ul style="list-style-type: none"> <li>• If no communication is possible with the device, then bit 0 is set to '1'.</li> <li>• If a link is detected on a port (A-D), then the corresponding bit (4-7) will be set to '1'. If no link is detected then the corresponding bit will be set to '0'.</li> </ul>
	<b>Data type</b>	UINT
	<b>Unit</b>	n/a
<b>Error</b>	<b>Description</b>	Indicates the function failed due to an error.
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>ErrorID</b>	<b>Description</b>	The function call error result, if Error is TRUE (see list of Error Codes in table below). Upon success, Error is set to zero..
	<b>Data type</b>	DINT
	<b>Unit</b>	

### State Defines

```
#define EC_STATE_NO_COMMUNICATION 0 (* 0x00 = No
Communication to device *)
#define EC_STATE_INIT 1 (* 0x01 = Device in
Init state *)
#define EC_STATE_PREOP 2 (* 0x02 = Device in
Pre-operational state *)
#define EC_STATE_BOOTSTRAP 3 (* 0x03 = Device in
Bootstrap state *)
#define EC_STATE_SAFEOP 4 (* 0x04 = Device in
Safe-Operational state *)
#define EC_STATE_OP 8 (* 0x08 = Device in
```



```
Operational state *)
#define EC_STATE_ERROR 16 (* 0x10 bit 4 set to
1; Device not in requested state error *)
```

**LinkStatus Defines**

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

**3.1.4.16.5 Related Functions**

"ECATWCStatus" (p. 721), "ECATMasterStatus" (p. 717)

**3.1.4.17.6 Example**

**TIP**

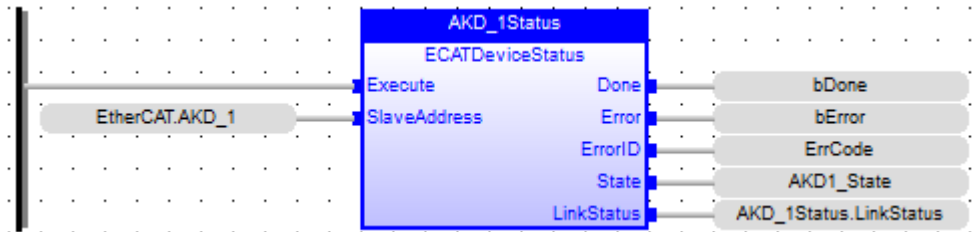
See [Checking the Device \(slave\) States](#): in the [EtherCAT Communication Diagnosis Steps](#) section of the [Troubleshooting](#) chapter for an example of implementing this function.

**3.1.4.18.7.1 Structured Text**

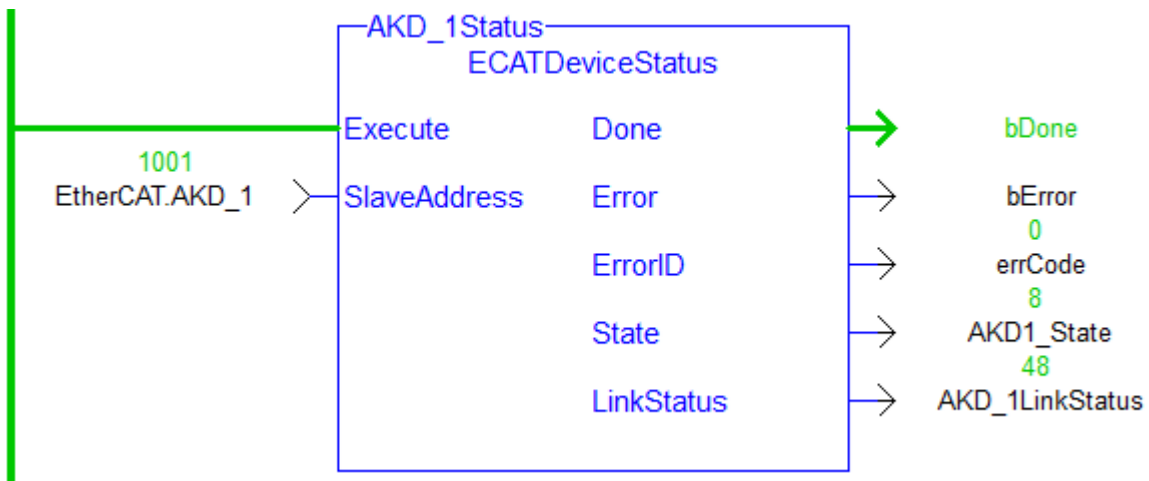
```
(*****
(* Read AKD_1 device state and link status*)
*****)

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

**3.1.4.19.8.2 FBD**



**3.1.4.20.9.3 FFLD**



**3.1.4.21 ECATGetStatus** PLCopen ✓ Pipe Network ✓

**3.1.4.22.1 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.

**3.1.4.23.2 Arguments**

**3.1.4.24.3.1 Input**

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

**3.1.4.25.4.2 Output**

<b>Status</b>	<b>Description</b>	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).
	<b>Data type</b>	UINT
	<b>Unit</b>	n/a

**3.1.4.26.5 Related Functions**

[ECATSetControl](#)

**3.1.4.27.6 Example**

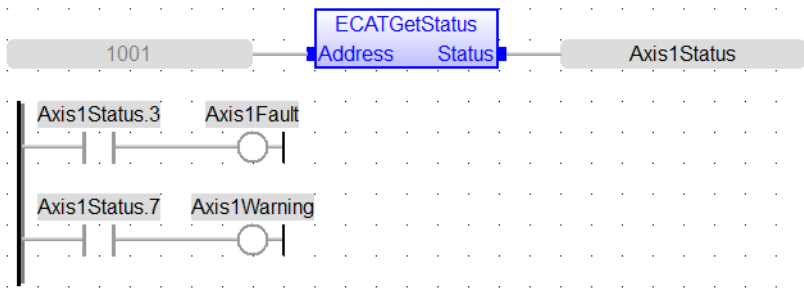
**3.1.4.28.7.1 Structured Text**

```

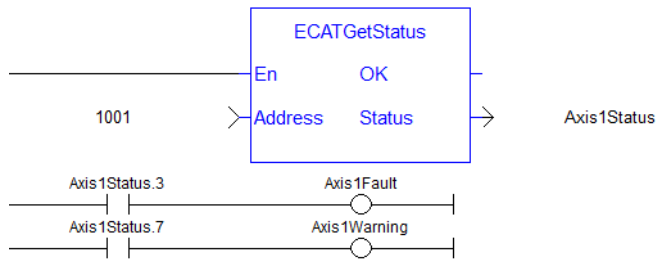
(*****
(* read EtherCAT axis status (Bit3: Fault, Bit7: Warning) *)
(*****

Axis1Status := ECATGetStatus(1001); //Read the ECAT Status Word
(SDO 6041) of the Axis
Axis1Fault := Axis1Status.3;
Axis1Warning := Axis1Status.7;
    
```

**3.1.4.29.8.2 FBD**



### 3.1.4.30.9.3 FFLD



3.1.4.31 ECATMasterStatus PLCopen ✓ Pipe Network ✓

3.1.4.32.1 Description

This function block reads the EtherCAT master state and the lost frame counter, to determine if EtherCAT is running normally.

**TIP**

See the [EtherCAT Communication Diagnosis Steps](#) section of the Troubleshooting chapter for more information.

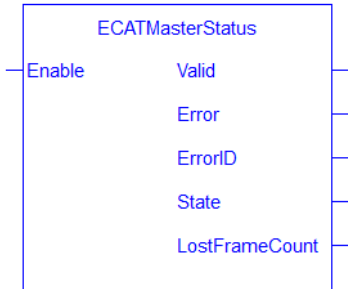


Figure 1-129: ECATMasterStatus function block

3.1.4.33.2 Arguments

3.1.4.34.3.1 Input

<b>Enable</b>	<b>Description</b>	Request to read the ECAT master state and the lost frame count. Keeps continuously reads the master state and the lost frame count as long as the Enable remains high.
	<b>Data type</b>	BOOL
	<b>Range</b>	0,1
	<b>Unit</b>	n/a
	<b>Default</b>	—

3.1.4.35.4.2 Output

<b>Valid</b>	<b>Description</b>	Indicates the values at the 'State' and LostFrameCount outputs are valid.
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Error</b>	<b>Description</b>	Indicates error
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>ErrorID</b>	<b>Description</b>	Error code when when the function block failed due to error.
	<b>Data type</b>	DINT

	<b>Unit</b>	n/a
<b>State</b>	<b>Description</b>	Indicates the ECAT state of the Master. See State Defines for details
	<b>Data type</b>	UINT
	<b>Unit</b>	n/a
<b>LostFrameCount</b>	<b>Description</b>	Total cumulative number of cyclic frames sent with no-response since the ECAT started by calling the MLMotionStart. Missing return frames will generate an A38 alarm. The Counter is reset to 0 when the MLMotionStart is called.
	<b>Data type</b>	UDINT
	<b>Unit</b>	n/a

**State Defines**

```
#define EC_STATE_NO_COMMUNICATION 0 (* 0x00 = No
Communication to device *)
#define EC_STATE_INIT 1 (* 0x01 = Device in
Init state *)
#define EC_STATE_PREOP 2 (* 0x02 = Device in
Pre-operational state *)
#define EC_STATE_BOOTSTRAP 3 (* 0x03 = Device in
Bootstrap state *)
#define EC_STATE_SAFEOP 4 (* 0x04 = Device in
Safe-Operational state *)
#define EC_STATE_OP 8 (* 0x08 = Device in
Operational state *)
```

**3.1.4.36.5 Related Functions**

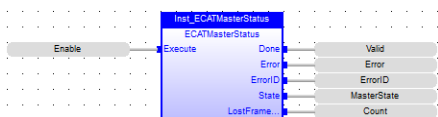
"ECATWCStatus" (p. 721), "ECATDeviceStatus" (p. 711).

**3.1.4.37.6 Examples**

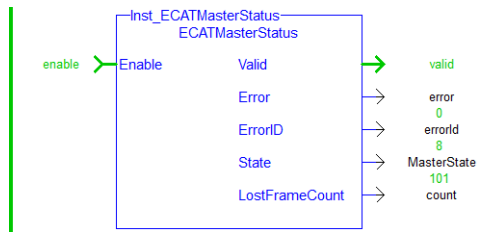
**3.1.4.38.7.1 Structured Text**

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

**3.1.4.39.8.2 FBD**



**3.1.4.40.9.3 FFLD**



### 3.1.4.41 ECATSetControl

#### 3.1.4.42.1 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.

#### 3.1.4.43.2 Arguments

##### 3.1.4.44.3.1 Input

<b>Address</b>	<b>Description</b>	EtherCAT address of the drive
	<b>Data type</b>	DINT
	<b>Range</b>	[0, 65535]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Control</b>	<b>Description</b>	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).
	<b>Data type</b>	UINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

##### 3.1.4.45.4.2 Output

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

#### 3.1.4.46.5 Related Functions

[ECATGetStatus](#)



### 3.1.4.47 ECATWCStatus

#### 3.1.4.48.1 Description

This function returns the current number of working counter errors for the Sync unit. The working counter errors are cleared to zero when the EtherCAT network is taken from **Init** to **OP** state.

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

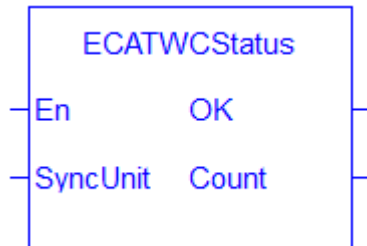


Figure 1-130: ECATWCStatus function

#### 3.1.4.49.2 Arguments

##### 3.1.4.50.3.1 Input

SyncUnit	Description	Sync Unit Index (for future compatibility with multiple frames)
	Data type	INT
	Range	0
	Unit	n/a
	Default	-

##### 3.1.4.51.4.2 Output

Count	Description	Working Counter error
	Data type	UDINT
	Unit	n/a

#### 3.1.4.52.5 Related Functions

["ECATDeviceStatus"](#) (p. 711), ["ECATMasterStatus"](#) (p. 717)

#### 3.1.4.53.6 Example

##### TIP

See [Checking for Working Counter Errors](#): in the [EtherCAT Communication Diagnosis Steps](#) section of the Troubleshooting chapter for an example of implementing this function.

#### 3.1.4.54.7.1 Structured Text

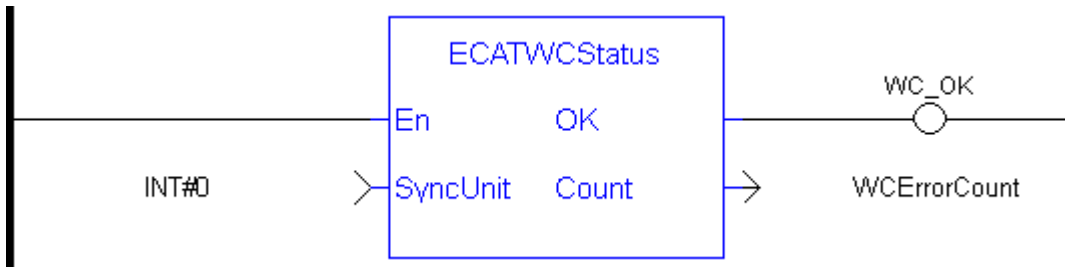
```

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

**3.1.4.55.8.2 FBD**



**3.1.4.56.9.3 FFLD**



### 3.2 Profibus Library PLCopen ✓ Pipe Network ✓

The Profibus driver has a thread that performs the exchange with the HW from time to time. The minimal exchange period (as well as the thread system priority) can be configured.

Name	Description
<b>PBGetExchPrio</b>	Return or change the priority of the I/O exchange thread. Which can be:
<b>PBSetExchPrio</b>	<ul style="list-style-type: none"> <li>• <b>#define CIF_EXCHANGE_PRIORITY_NORMAL 0</b> /* Profibus exchange thread priority lower than VM thread priority */</li> <li>• <b>#define CIF_EXCHANGE_PRIORITY_HIGHER 1</b> /* Profibus exchange thread priority equal to VM thread priority */</li> </ul>
<b>PBGetExchPeriod</b>	Return or change the minimum period of the I/O exchange thread with the HW (in PLC cycles). 2 is the minimum value. Taking the system load into account, it can be set to more than 2.
<b>PBSetExchPeriod</b>	<p><i>Example:</i> set the period to 2 if you want the exchange to be performed at the most once every 2 cycles (compared with the VM cycle)</p>
<b>PBGetExchNb</b>	Return the amount of exchange performed with the HW
<b>PBGetLastPeriod</b>	Return the last period between two exchanges
<b>PBGetMaxPeriod</b>	idem but max period
<b>PBResetStats</b>	Reset the previous statistics of the I/O exchange thread. Which are: <ul style="list-style-type: none"> <li>• the number of exchanges done</li> <li>• the last exchange period</li> <li>• the maximum exchange period</li> </ul> <p>Statistics are reset each time the driver is Open (i.e. when the program start)</p>

Table 1-8: List of Profibus FB

## 4 System Library

---

<b>4.1 PrintMessage</b> .....	<b>725</b>
<b>4.2 GetCtrlErrors</b> .....	<b>728</b>
<b>4.3 ClearCtrlErrors</b> .....	<b>729</b>
<b>4.4 GetCtrlInfo</b> .....	<b>730</b>
<b>4.5 GetCtrlPerf</b> .....	<b>733</b>

Name	Description
"PrintMessage" (p. 725)	Generate an output message in the log windows.
"GetCtrlErrors" (p. 728)	Get a list of the active errors and alarms on the controller.
"ClearCtrlErrors" (p. 729)	Clears the list of active errors and alarms on the controller.
"GetCtrlPerf" (p. 733)	Generate a text file with performance statistics of the controller.
"GetCtrlInfo" (p. 730)	Get the serial, model, and/or part number of the controller.

Table 1-9: List of System Functions

## 4.1 PrintMessage

### 4.1.1 Description

The PrintMessage block is used to generate a log message with any wanted strings in the [Log Messages](#) window.

#### 4.1.1.1 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.

#### 4.1.1.2 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.

### 4.1.2 Arguments

#### 4.1.2.1 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.
	<b>Data type</b> DINT
	<b>Range</b> [0 , 4] Defines are: LEVEL_DEBUG, LEVEL_INFO, LEVEL_WARNING, LEVEL_ERROR, LEVEL_CRITICAL
	<b>Unit</b> n/a

	<b>Default</b>	—
<b>Message</b>	<b>Description</b>	Content of the message. A string of 255 characters maximum.
	<b>Data type</b>	String
	<b>Range</b>	1 to 255 characters
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 4.1.2.2 Output

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

#### 4.1.3 Usage

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

#### 4.1.4 Example

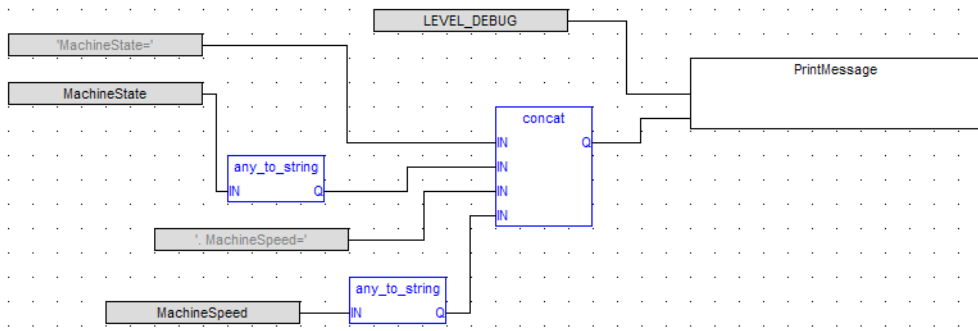
##### 4.1.4.1 Structured Text

```
// It's possible to create a temporary variable with the
// message.
MESSAGE := CONCAT( 'MachineState=', ANY_TO_STRING(MachineState),
'. MachineSpeed=', ANY_TO_STRING(MachineSpeed) );

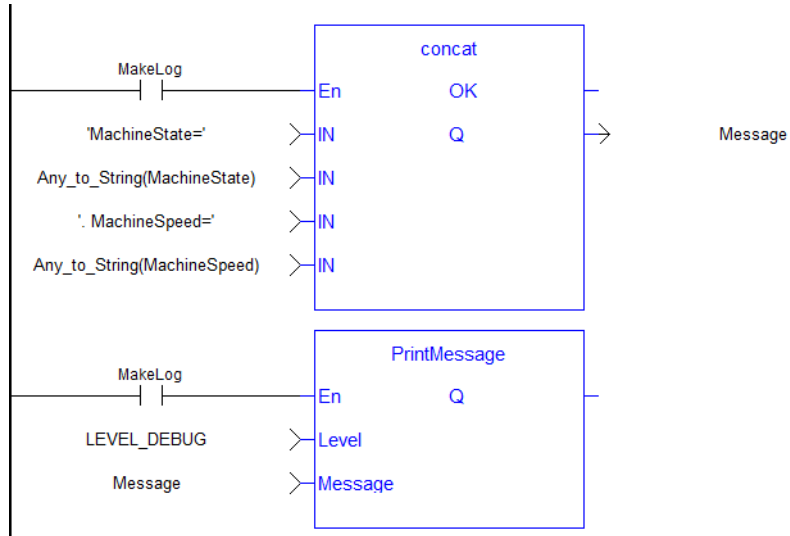
// Then print the message to the log window
PrintMessage( LEVEL_INFO, MESSAGE );
PrintMessage( LEVEL_WARNING, MESSAGE );
PrintMessage( LEVEL_ERROR, MESSAGE );

// Or to create the string directly in the function call:
PrintMessage( LEVEL_CRITICAL, CONCAT( 'MachineState=', ANY_TO_
STRING(MachineState), '. MachineSpeed=', ANY_TO_STRING
(MachineSpeed) ) );
```

##### 4.1.4.2 Function Block Diagram



#### 4.1.4.3 FFLD



## 4.2 GetCtrlErrors PLCopen ✓ Pipe Network ✓

Returns active errors and alarms on the controller in two arrays of hundred Booleans. Every index in the array corresponds to the error and alarm numbers in the tables. See [Errors](#) for a list of errors and alarms that may be generated.

### 4.2.1 Arguments

#### 4.2.1.1 Input

<b>EN</b>	<b>BOOL</b>	Enable
<b>ActiveError</b>	<b>BOOL[100]</b>	Array of bool with the size equal to 100
<b>ActiveAlarm</b>	<b>BOOL[100]</b>	Array of bool with size equal to 100

#### 4.2.1.2 Output

<b>OK</b>	<b>BOOL</b>	
<b>Q</b>	<b>DINT</b>	Status of the execution

Status meaning:

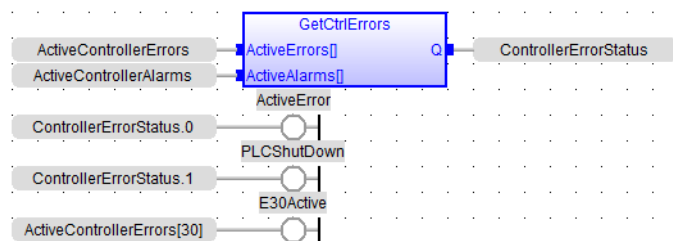
Bit 0	Value 0	No error, no alarm, no shut down
Bit 0	Value 1	There is an active error or an active alarm (i.e. there is something in the array ActiveError/ActiveAlarm)
Bit 1	Value 0	No shut down
Bit 1	Value 1	The PLC processes will be shut down. This will start 10 seconds after the error is triggered
Bit 2-15	Value 2   4   9 to 2147483648	reserved

### 4.2.2 Examples

#### **TIP**

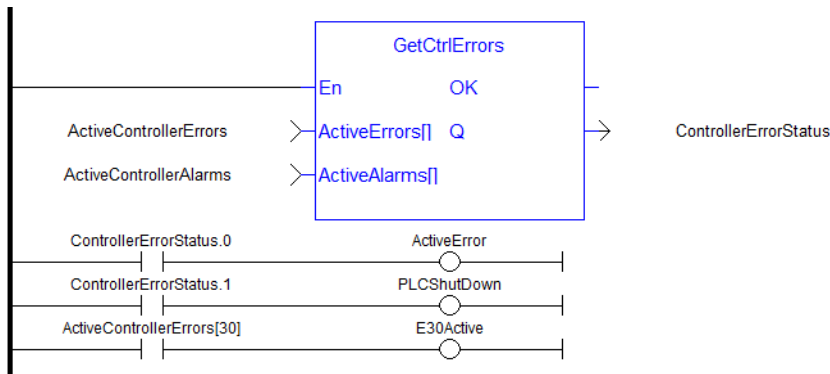
See [Checking for existing EtherCAT Alarms and Errors](#): in the [EtherCAT Communication Diagnosis Steps](#) section of the Troubleshooting chapter for an example of implementing this function.

#### 4.2.2.1 FBD



#### 4.2.2.2 FFLD





### 4.2.2.3 ST

```
//Retrieve active controller level alarm and errors.
//Check status output to see if any error or alarm is active and
//if PLC is shutting down
ControllerErrorStatus:= GetCtrlErrors( ActiveControllerErrors,
ActiveControllerAlarms);
ActiveError:= ControllerErrorStatus.0;
PLCShutDown:= ControllerErrorStatus.1;
E30Active:= ActiveControllerErrors[30];
```

## 4.3 ClearCtrlErrors PLCopen ✓ Pipe Network ✓

Clears the active errors and alarms on the controller. Only clearable errors will be cleared. See [Errors](#) for a list of errors and alarms that may be generated.

### 4.3.1 Arguments

#### 4.3.2 Input

<b>EN</b>	<b>Data Type</b>	Enable the function
-----------	------------------	---------------------

#### 4.3.3 Output

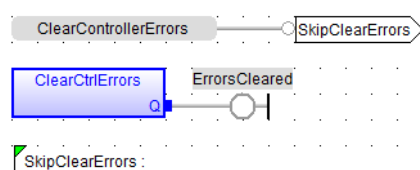
<b>Q</b>	<b>Data Type</b>	BOOL
----------	------------------	------

#### NOTE

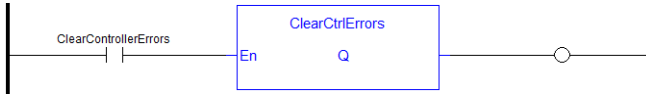
If clearable and non-clearable errors are present and this function is called, the Output Q will be turned to true but the non-clearable errors will remain.

### 4.3.4 Examples

#### 4.3.4.1 FBD



### 4.3.4.2 FFLD



### 4.3.4.3 ST

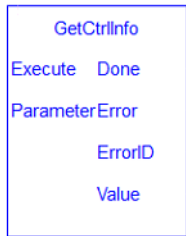
```
//Attempt to clear active controller level errors and alarms
(such as E33)
IF ClearControllerErrors THEN
ClearCtrlErrors();
END_IF;
```

## 4.4 GetCtrlInfo

PLCopen ✓

Pipe Network ✓

This function block returns a String containing the value of the control parameter requested.



### 4.4.1 Arguments

#### 4.4.1.1 Input

<b>Execute</b>	<b>Description</b>	Rising edge of enable initiates read of parameter
	<b>Data Type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Parameter Number</b>	<b>Description</b>	Parameter number to read
	<b>Data Type</b>	INT
	<b>Range</b>	[1,3]
	<b>Unit</b>	n/a

<b>Default</b>	<p>1 = PDMM/PCMM serial number 2 = PDMM/PCMM model number 3 = PDMM/PCMM part number0</p> <p>These parameters are also <a href="#">Internal Defines</a>, as shown in the table below.</p> <pre>#define CTRLINFO_SERIAL_NUMBER 1 #define CTRLINFO_MODEL_NUMBER 2 #define CTRLINFO_PART_NUMBER 3</pre>
----------------	---

#### 4.4.1.2 Output

<b>Done</b>	<p><b>Description</b> Indication that read completed without error</p> <p><b>Data Type</b> BOOL</p>
<b>Error</b>	<p><b>Description</b> Indication that read completed with error</p> <p><b>Data Type</b> BOOL</p>
<b>ErrorID</b>	<p><b>Description</b> Error value to indicate error condition</p> <p><b>Data Type</b> INT NO Error : 0 Invalid parameter : 1 Error reading data : 2 Not valid on non-PDMM : 3</p> <p>These parameters are also <a href="#">Internal Defines</a>, as shown in the table below.</p> <pre>#define CTRLINFO_ERROR_NO_ERROR 0 #define CTRLINFO_ERROR_INV_PARAMETER 1 #define CTRLINFO_ERROR_CANT_READ_DATA 2 #define CTRLINFO_ERROR_NOT_PDMM 3</pre>
<b>Value</b>	<p><b>Description</b> String containing data that was read.</p> <p><b>Data Type</b> STRING</p>

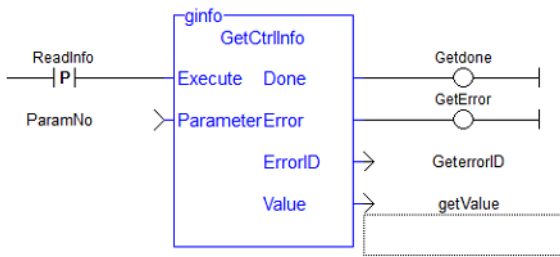
#### 4.4.2 Examples

##### 4.4.2.1 Structured Text

```
Inst_GetCtrlInfo( ExecuteRead, 1);

if Inst_GetCtrlInfo.Done then
    serialNumber := Inst_GetCtrlInfo.Value;
end_if;
```

##### 4.4.2.2 Ladder Diagram



## 4.5 GetCtrlPerf

### 4.5.1 Description

This function block returns controller CPU performance statistics.

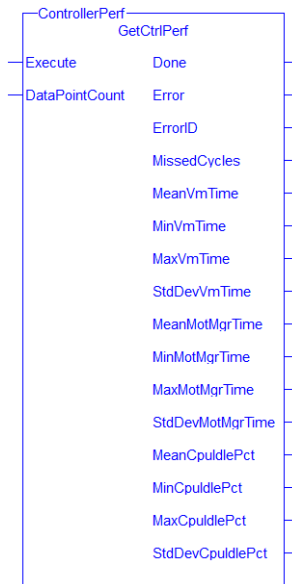


Figure 1-131: GetCtrlPerf

See also:

- [Differences Between Functions and Function Blocks](#)
- [Calling a function block](#)

### 4.5.2 Arguments

#### 4.5.2.1 Input

<b>Execute</b>	<b>Description</b>	On the rising edge, request to collect the controller's performance data.
	<b>Data type</b>	BOOL
	<b>Range</b>	0,1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>DataPointCount</b>	<b>Description</b>	The number of motion manager cycles over which performance statistics will be gathered.
	<b>Data type</b>	UDINT
	<b>Range</b>	2 - 240,000
	<b>Unit</b>	n/a
	<b>Default</b>	—

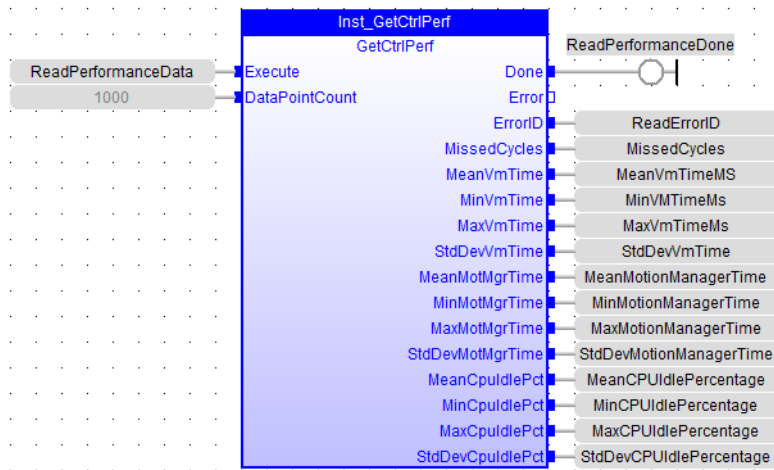
## 4.5.2.2 Output

<b>Done</b>	<b>Description</b>	If True, then the command completed successfully.
	<b>Data type</b>	BOOL
<b>Error</b>	<b>Description</b>	If True, an error has occurred.
	<b>Data type</b>	BOOL
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE. ErrorID = 0 indicates no error, ErrorID = 1 indicates an error
	<b>Data type</b>	INT
<b>MissedCycles</b>	<b>Description</b>	Indicates the number of missing VM cycles.
	<b>Data type</b>	UDINT
<b>MeanVmTime</b>	<b>Description</b>	The mean VM execution time measured in microseconds.
	<b>Data type</b>	LREAL
<b>StdDevVmTime</b>	<b>Description</b>	The standard deviation of the VM execution time measured in microseconds.
	<b>Data type</b>	LREAL
<b>MinVmTime</b>	<b>Description</b>	The minimum VM execution time measured in microseconds.
	<b>Data type</b>	LREAL
<b>MaxVmTime</b>	<b>Description</b>	The maximum VM execution time measured in microseconds.
	<b>Data type</b>	LREAL
<b>MeanMotMgrTime</b>	<b>Description</b>	The mean motion manager execution time measured in microseconds.
	<b>Data type</b>	LREAL
<b>StdDevMotMgrTime</b>	<b>Description</b>	The standard deviation of the motion manager execution time measured in microseconds.
	<b>Data type</b>	LREAL
<b>MinMotMgrTime</b>	<b>Description</b>	The minimum motion manager execution time measured in microseconds.
	<b>Data type</b>	LREAL
<b>MaxMotMgrTime</b>	<b>Description</b>	The maximum motion manager execution time measured in microseconds.
	<b>Data type</b>	LREAL
<b>MeanCpuldlePct</b>	<b>Description</b>	The mean percentage of time the controller CPU is idle.

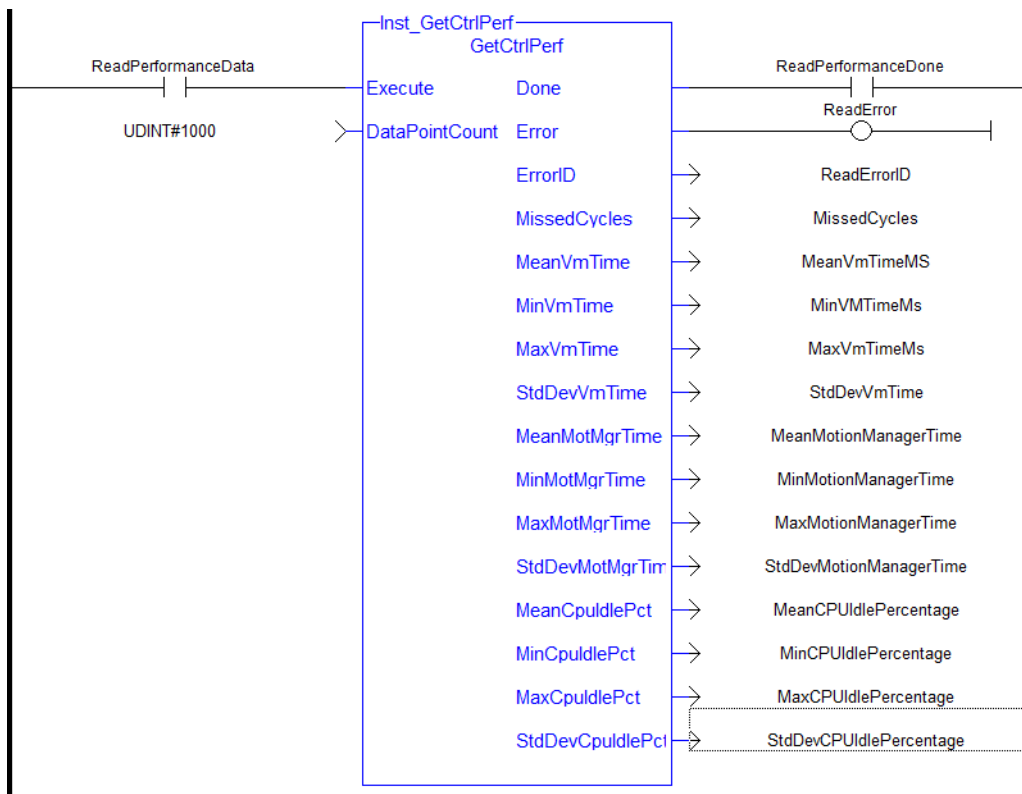
	<b>Data type</b>	LREAL
<b>StdDevCpudlePct</b>	<b>Description</b>	The standard deviation of the measurements of the time that the controller CPU is idle.
	<b>Data type</b>	LREAL
<b>MinCpudlePct</b>	<b>Description</b>	The minimum measurement of the time that the controller CPU is idle.
	<b>Data type</b>	LREAL
<b>MaxCpudlePct</b>	<b>Description</b>	The maximum measurement of the time that the controller CPU is idle.
	<b>Data type</b>	LREAL

### 4.5.3 Example

#### 4.5.3.1 FBD



#### 4.5.3.2 FFLD



#### 4.5.3.3 Structured Text

```
//Read controller performance data from last 1000 cycles (1
second at T#1ms update rate)
Inst_GetCtrlPerf( ReadPerformanceData, 1000 );

IF Inst_GetCtrlInfo.Done THEN
MissedCycles:= Inst_GetCtrlPerf.MissedCycles;
MaxVmTimeMs:= Inst_GetCtrlPerf.MaxVmTime;
MaxMotionManagerTime:= Inst_GetCtrlPerf.MaxMotMgrTime;
MeanCPUIdlePercentage:= Inst_GetCtrlPerf.MeanCpuIdlePct;
MaxCPUIdlePercentage:= Inst_GetCtrlPerf.MaxCpuIdlePct;
END_IF;
```



## 5 Kollmorgen UDFBs

---

5.1 How to create an instance .....	740
5.2 Working with Kollmorgen UDFBs .....	741

A KollmorgenUDFB<sup>1</sup> is a pre-defined function block created by Kollmorgen to simplify certain tasks or demonstrate a particular function. A Kollmorgen UDFB must be instantiated and unlocked before it may be used.

Name	Description
<a href="#">FB_AKDFItRpt</a>	Outputs AKD fault information
<a href="#">FB_AxisPlsPosModulo</a>	
<a href="#">FB_AxisPlsPosNoModulo</a>	
<a href="#">FB_Cylinder</a>	Control a cylinder and the Limit Switches.
<a href="#">FB_ElapseTime</a>	Keeps track of the time that a Boolean input variable is on.
<a href="#">FB_FirstOrderDigitalFilter</a>	Filter an Analog signal.
<a href="#">FB_PWDutyOutput</a>	Converts an input range to a duty cycle percentage
<a href="#">FB_S700FitRpt</a>	Outputs S700 drive fault Information
<a href="#">FB_ScaleInput</a>	Scale DINT to LREAL
<a href="#">FB_ScaleOutput</a>	Scale DINT to LREAL
<a href="#">FB_TemperaturePID (→ p. 765)</a>	{rovides PID temperature control with auto tuning
<a href="#">MCFB_AKDFault (→ p. 886)</a>	Outputs AKD drive fault Information.
<a href="#">MCFB_AKDFaultLookup (→ p. 888)</a>	String message of the corresponding AKD drive fault number
<a href="#">"MCFB_GearedWebTension" (p. 872)</a>	Facilitates dancer and tension control in an electronic geared master/slave machine design
<a href="#">MCFB_Jog</a>	Jog an axis in the selected direction at a defined speed
<a href="#">MCFB_StepAbsolute</a>	Performs a static homing function by setting Actual Position to the position of an absolute encoder
<a href="#">MCFB_StepAbsSwitch</a>	Performs a homing function by searching for an absolute positioned external physical switch
<a href="#">MCFB_StepAbsSwitchFastInput</a>	Performs a homing function by searching for an absolute positioned external physical switch
<a href="#">MCFB_StepBlock</a>	Performs homing against a physical object, mechanically blocking the movement
<a href="#">MCFB_StepLimitSwitch</a>	Performs a single-axis home to a limit switch
<a href="#">MCFB_StepLimitSwitchFastInput</a>	Performs a homing function by searching for an external physical switch
<a href="#">MCFB_StepRefPulse</a>	Performs homing by searching for Zero pulse, Marker, or reference pulse in encoder

<sup>1</sup>"User Defined Function Block" UDFB can be used as a sub-function block in another program of the application. It is described using FBD, LD, ST or IL language. Input / output parameters of a UDFB (as well as private variables) are declared in the variable editor as local variables of the UDFB

Name	Description
<a href="#">MLFB_HomeFindHomeFastInput</a>	Performs a single-axis home to a limit switch connected to a High Speed Input
<a href="#">MLFB_HomeFindHomeFastInputModulo</a>	Performs a single-axis home to a limit switch connected to a High Speed Input
<a href="#">MLFB_HomeFindHomeInput</a>	Fast Homing to a home switch
<a href="#">MLFB_HomeFindHomeInputThenZeroAngle</a>	Fast Homing to a home switch + Zero angle
<a href="#">MLFB_HomeFindLimitFastInput</a>	Homing to a limit switch
<a href="#">MLFB_HomeFindLimitFastInputModulo</a>	Homing to a limit switch: Modulo mode
<a href="#">MLFB_HomeFindLimitInput</a>	Homing to a limit switch
<a href="#">MLFB_HomeFindLimitInputThenZeroAngle</a>	Homing to a limit switch + Zero angle
<a href="#">MLFB_HomeFindZeroAngle</a>	Homing to a zero-angle reference
<a href="#">MLFB_HomeMoveUntilPosErrExceeded</a>	Homing until the position error is exceeded
<a href="#">MLFB_HomeMoveUntilPosErrExceededThenZeroAngle</a>	Homing until the position error is exceeded + Zero angle
<a href="#">MLFB_HomeUsingCurrentPosition</a>	Homing using the current position
<a href="#">MLFB_Jog</a>	Jog in a selected direction at a defined speed
<a href="#">MLFB_PlsPosFw</a>	Forward position range indicator
<a href="#">MLFB_PlsPosFwBw</a>	Forward/Backward position range indicator
<a href="#">MLFB_PlsTimeFw</a>	Forward/Backward position/time range indicator
"PipeNetwork_FFLD" (p. 761)	Used to call the PNCode function block in FFLD POU's
"ProfilesCode_FFLD" (p. 763)	Used to call the ProfilesCode function block in FFLD POU's

Table 1-10: List of System Kollmorgen UDFBs

## 5.1 How to create an instance

- Open your PLC code
- Select the UDFB in the [Library](#) tree
- Drag-and-drop the UDFB in the PLC editor to create the instance of the UDFB

An instance of the UDFB has now been created in **Subprograms**.

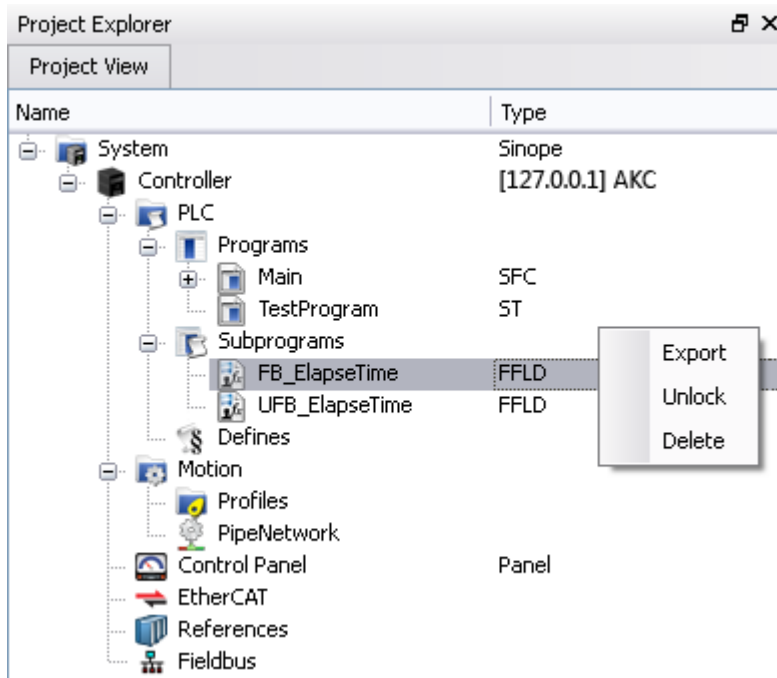
### NOTE

You cannot create the instance of the UDFB directly from the dictionary or from the PLC Editor.

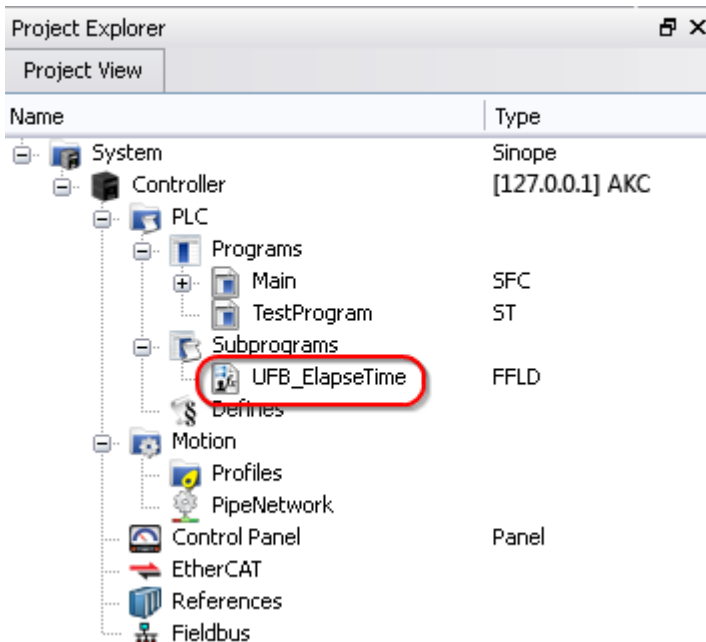
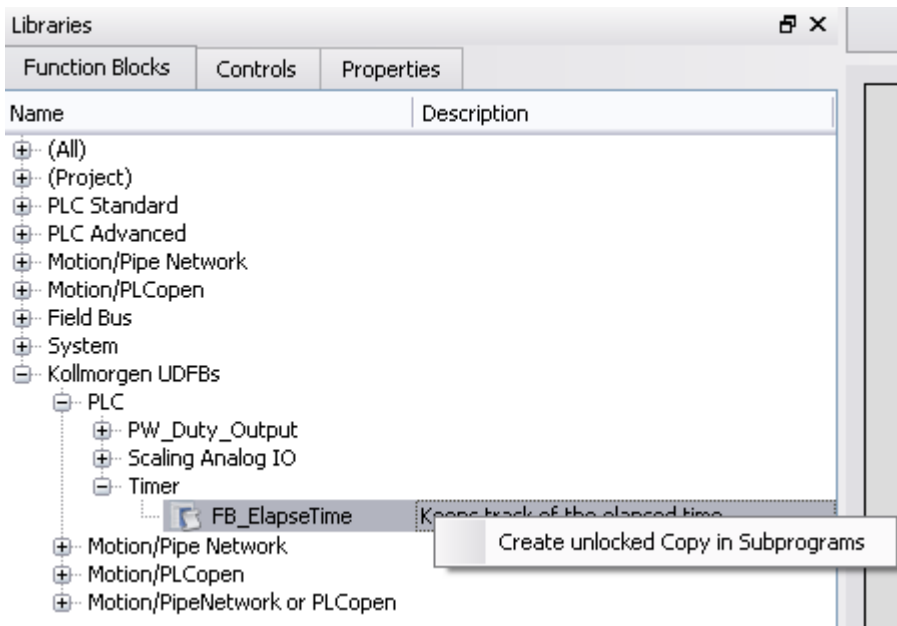
## 5.2 Working with Kollmorgen UDFBs

By default all Kollmorgen UDFBs are protected, meaning they may not be modified or renamed. When a Kollmorgen UDFB is dropped into an instance it is not editable. There are two solutions to make it editable:

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



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



Once a Kollmorgen UDFB has been unlocked it may be renamed and exported by right-clicking on the UDFB. Renamed UDFBs must have unique names. Importing a saved UDFB will increment the UDFB's name.

**TIP**

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

**5.2.1 FB\_FirstOrderDigitalFilter**

**5.2.1.1 Description**

This FB is defined to filter an Analog signal.

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

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

The following figure shows the function block I/O

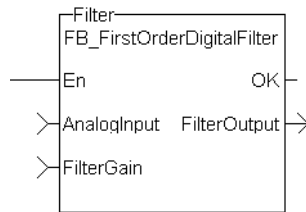


Figure 1-132: CBS First Order Digital Filter

## 5.2.1.2 Arguments

### 5.2.1.3.1 Inputs

<b>EN</b>	<b>Description</b>	Enables execution (FFLD only )
	<b>Data type</b>	BOOL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	
<b>AnalogInput</b>	<b>Description</b>	Analog Input from transducer
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	
<b>FilterGain</b>	<b>Description</b>	Filter Gain
	<b>Data type</b>	REAL
	<b>Range</b>	[1 - 0.05]
	<b>Unit</b>	n/a
	<b>Default</b>	—

### 5.2.1.4.2 Outputs

<b>OK</b>	<b>Description</b>	Execution Complete
	<b>Data type</b>	BOOL
	<b>Range</b>	[0,1]

Unit	
<b>FilterOutput</b>	<b>Description</b> Filtered analog input value
	<b>Data type</b> REAL
	<b>Range</b> [0,1]
Unit	

### 5.2.1.5 Usage

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

- The AnalogInput input is the unfiltered “raw” analog feedback signal from an LVDT, tension transducer, potentiometer, or some other like device.
- The FilterGain defines the amount of filtering to be used. The range of the gain is from 1.0 or no filtering to 0.05 or the maximum filtering.
- The FilterOutput is the filtered analog input and is typically used as an input to some other function block or UDFB that has an analog input, for example the MCFB\_GearedWebTension UDFB.
- The implementation of the digital first order filter is for PLCopen.
- The equation is defined as:  $Input * Gain + Output * (1 - Gain) = Output$
- The steady state filter delay with a gain of 0.8 can be seen in the following table.

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

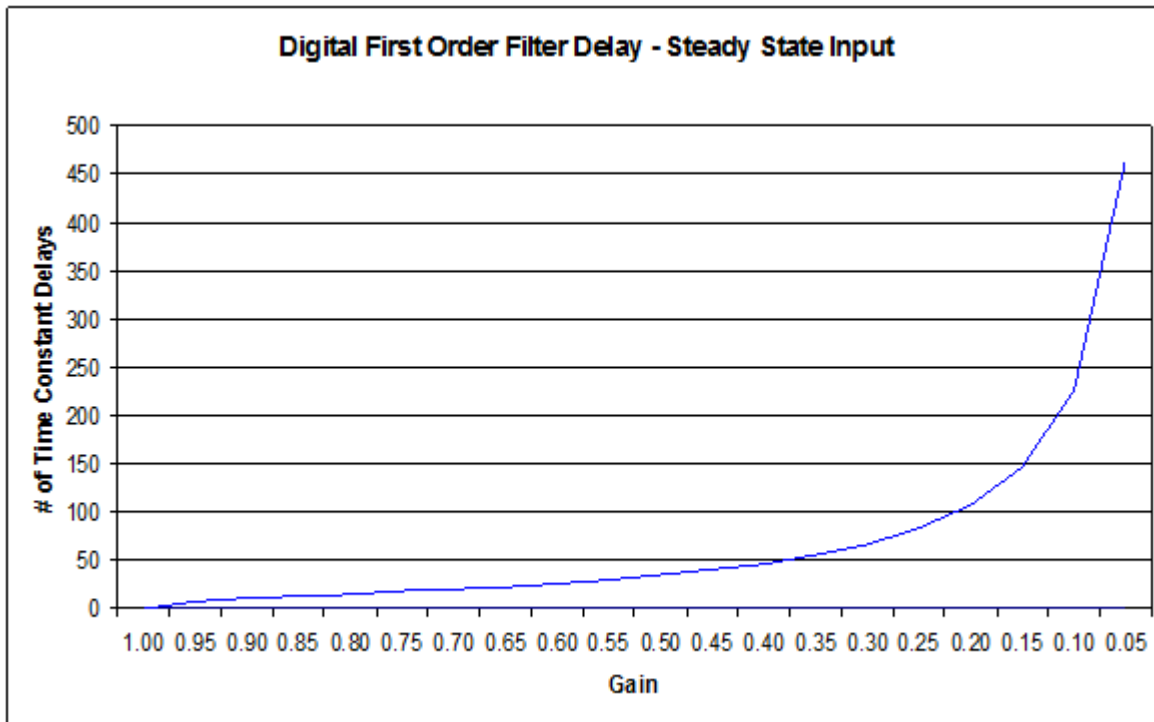


Table 1-11: Filter Input Delay Example

The range of the filter gain is between 1.00 and 0.05. From the table, for a filter gain of 0.8 there is a delay of 15 time constants with a time constant defined as the rate the UDFB is scanned or executed in the application. For example if the UDFB was executed every millisecond a gain of 0.8 would provide a filter delay of 15ms. Conversely a gain of 1.00 provides zero filtering and the output signal follows the input signal, and a gain of 0.05 provides the most filtering for 463 ms.

The numbers of filter delays for a steady state analog input at a given gain are shown in the table and graph below.

Gain	Filter Delay Tn
1.00	0
0.95	8
.90	11
.85	13
.80	15
.75	18
.70	20
.65	23
.60	26
.55	30
.50	35
.45	40
.40	47
.35	56
.30	66
.25	83
.20	107
.15	146
.10	226
.05	463



Of course a real world analog input is most always a varying feedback signal. In Table 2.3 this is shown with an initial input of 100, a gain of 0.8, and a random variability of 10%. Filter Input

Filter Input	Filter Current Output	Amount of Input Filtering	Random Filter % Variation
0	0	0	10%
100	80	-20	
97.38903813	93.9112305	-3.477807626	
92.67638093	92.92335084	0.246969915	
94.12988912	93.88858146	-0.241307655	
103.0835564	101.2445614	-1.838994993	
91.16845433	93.18367575	2.015221422	
93.23936976	93.22823096	-0.011138803	
94.90272089	94.56782291	-0.334897986	
103.3070737	101.5592235	-1.747850153	
96.83149418	97.77704005	0.945545867	
96.35024002	96.63560002	0.285360007	
99.82417525	99.1864602	-0.637715045	
105.0792636	103.9007029	-1.178560685	
97.36988208	98.67604626	1.306164172	
107.82502	105.9952253	-1.829794752	
97.7886524	99.42996698	1.641314572	
108.2038024	106.4490353	-1.754767081	
91.58527607	94.55802792	2.972751845	
93.6783421	93.85427926	0.175937164	
102.8695349	101.0664838	-1.803051129	
93.95916817	95.3806313	1.421463121	

Filter Input	Filter Current Output	Amount of Input Filtering	Random Filter % Variation
108.6579707	106.0025028	-2.655467871	
109.3425748	108.6745604	-0.668014397	
103.9066	104.8601921	0.953592077	
92.30112142	94.81293555	2.511814127	
109.4460726	106.5194452	-2.926627416	
94.88799896	97.21428821	2.326289251	
105.4738635	103.8219484	-1.651915057	
102.988167	103.1549233	0.166756284	
92.92925408	94.97438792	2.045133846	
95.58185568	95.46036213	-0.121493552	
109.414248	106.6234708	-2.790777178	
106.5661311	106.577599	0.011467953	
99.85857253	101.2023778	1.343805301	
107.865421	106.5328124	-1.332608643	
92.19683177	95.0640279	2.867196126	
104.8558146	102.8974573	-1.958357346	
104.5140236	104.1907104	-0.323313268	
104.3675014	104.3321432	-0.035358206	
109.2704266	108.2827699	-0.987656683	
101.4962729	102.8535723	1.35729941	
92.19199163	94.32430776	2.132316128	
99.13065312	98.16938405	-0.961269073	
103.5068114	102.4393259	-1.067485466	
109.502983	108.0902516	-1.412731426	
99.05504822	100.8620889	1.80704068	
94.97711299	96.15410817	1.176995182	
107.1063597	104.9159094	-2.190450308	
91.12245188	93.88114339	2.758691504	
108.130314	105.2804799	-2.849834129	
104.2923832	104.4900025	0.197619344	
101.3775072	102.0000062	0.62249907	
<b>100.5303014</b>	<b>100.0399168</b>	<b>-0.490384645</b>	<b>Averages</b>

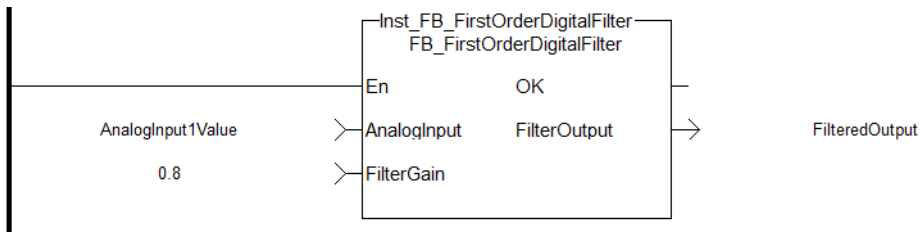
Table 1-12: Filter Input Lag Example - Random Input

### 5.2.1.6 Example

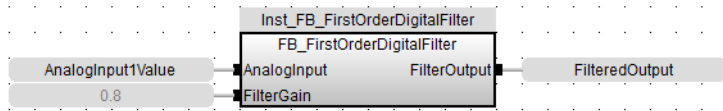
#### 5.2.1.7.1 Structured Text

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

#### 5.2.1.8.2 Ladder Diagram



### 5.2.1.9.3 Function Block Diagram



## 5.2.2 FB\_PWDutyOutput

### 5.2.2.1 Description

The Pulse Width Duty Cycle function block accepts an input value between the minimum and maximum input range and converts this to a duty cycle percentage. The output is then cycled on and off over the period of the duty cycle at the duty cycle percentage. If it is desired to have the output ON time range from 0 to the duty cycle period, the minimum should be set to zero and the maximum to the duty cycle period. If the calculated duty cycle based on the input and range values is less than the minimum ON time (MinTime), the output will not come on. If the calculated duty cycle is between or equal to the range values the output is cycled by the duty cycle. If the calculated duty cycle is greater than the maximum ON time (MaxTime) the output will remain on.

The following figure shows the function block I/O

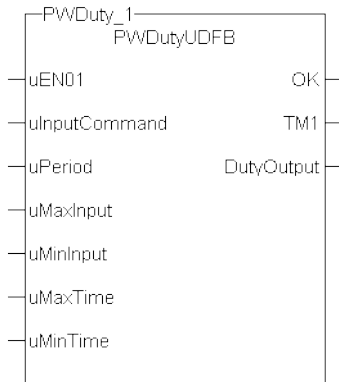


Figure 1-133: Pulse Width Duty Cycle

### 5.2.2.2 Arguments

#### 5.2.2.3.1 Input

uEN01	Description	Enable for the block
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
uInputCommand	Description	Duty Cycle Input (sometimes the output of a PID block).
	Data type	REAL
	Range	[0 , 1]
	Unit	n/a
	Default	—
uPeriod	Description	Period of the duty cycle
	Data type	<a href="#">TIME</a>
	Range	[0 , 1]
	Unit	n/a
	Default	—
uMaxInput	Description	Maximum value for the Input
	Data type	REAL
	Range	[0 , 1]
	Unit	n/a
	Default	—
uMinInput	Description	Minimum value for the Input
	Data type	REAL

	Range	[0 , 1]
	Unit	n/a
	Default	—
uMaxTime	Description	Maximum on time for the Output
	Data type	<a href="#">TIME</a>
	Range	[0 , 1]
	Unit	n/a
	Default	—
uMinTime	Description	Minimum on time for the Output
	Data type	<a href="#">TIME</a>
	Range	[0 , 1]
	Unit	n/a
	Default	—

#### 5.2.2.4.2 Output

OK	Description	Function block is OK.
	Data type	BOOL
	Unit	n/a
TM1	Description	Duty cycle on time
	Data type	<a href="#">TIME</a>
	Unit	n/a
DutyOutput	Description	Indicates if output is on or off
	Data type	BOOL
	Unit	n/a

#### 5.2.2.5 Usage

Flash a warning light for operators.

#### 5.2.2.6 Related Functions

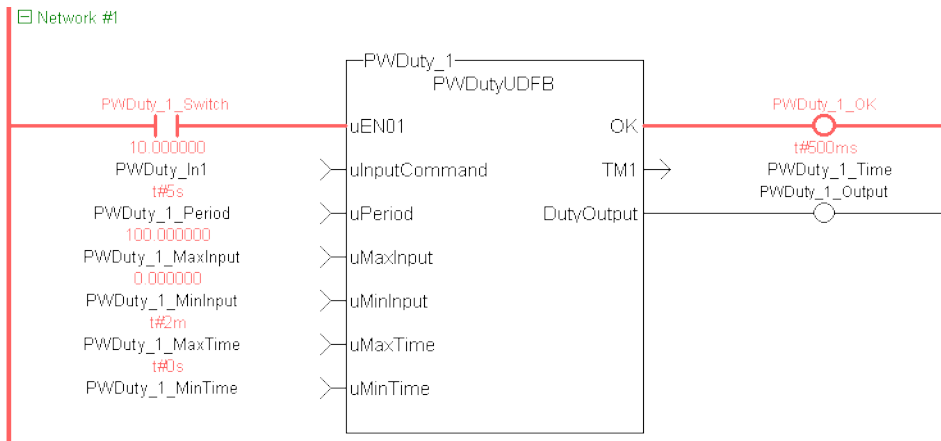
[Timers](#)

#### 5.2.2.7 Example

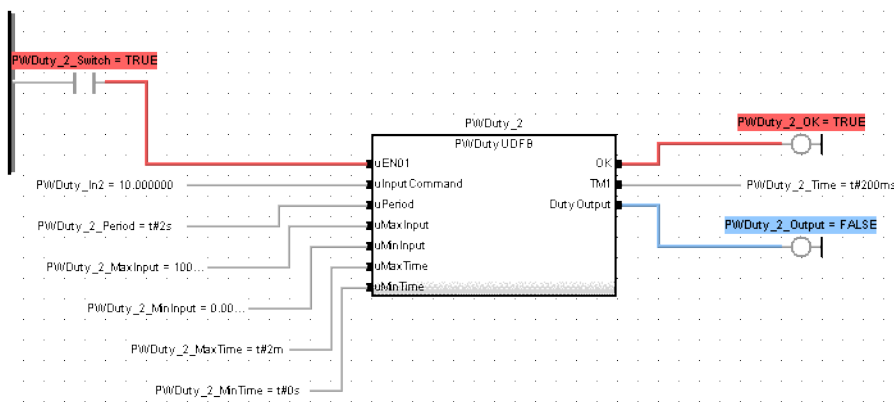
##### 5.2.2.8.1 Structured Text

```
Inst_FB_PWDutyOutput( PWDuty_3_Switch, PWDuty_In3, PWDuty_3_Period, PWDuty_3_MaxInput,
PWDuty_3_MinInput, PWDuty_3_MaxTime, PWDuty_3_MinTime);
PWDuty_3_OK:=Inst_FB_PWDutyOutput.OK;
PWDuty_3_Time:=Inst_FB_PWDutyOutput.TM1;
PWDuty_3_Output:=Inst_FB_PWDutyOutput.DutyOutput;
```

##### 5.2.2.9.2 Ladder Diagram



### 5.2.2.10.3 Function Block Diagram



### 5.2.2.11 FB\_ScaleInput

#### 5.2.2.12.1 Description

Scale DINT to LREAL.

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

The following figure shows the function block I/O:

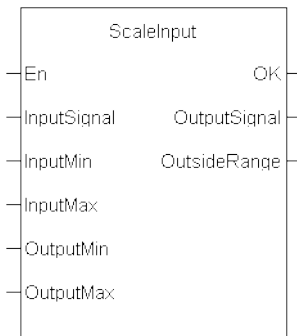


Figure 1-134: Scale Input

#### 5.2.2.13.2 Arguments

##### 5.2.2.14.3.1 Input

<b>InputSignal</b>	<b>Description</b>	Un-scaled input signal
	<b>Data type</b>	DINT
	<b>Range</b>	[0 , 4]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>InputMin</b>	<b>Description</b>	Minimum value of accepted input signal range
	<b>Data type</b>	DINT
	<b>Range</b>	[0 , 4]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>InputMax</b>	<b>Description</b>	Maximum value of accepted input signal range
	<b>Data type</b>	DINT
	<b>Range</b>	[0 , 4]
	<b>Unit</b>	n/a



	<b>Default</b>	—
<b>OutputMin</b>	<b>Description</b>	Output value mapped to the InputMin
	<b>Data type</b>	LREAL
	<b>Range</b>	[0 , 4]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>OutputMax</b>	<b>Description</b>	Output value mapped to the InputMax
	<b>Data type</b>	LREAL
	<b>Range</b>	[0 , 4]
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 5.2.2.15.4.2 Output

<b>OutputSignal</b>	<b>Description</b>	Scaled value of the Input Signal with type converted to LREAL. Stays within specified Min/Max output values
	<b>Data type</b>	LREAL
	<b>Unit</b>	n/a
<b>OutsideRange</b>	<b>Description</b>	True if InputSignal is outside range setup by min/max values, otherwise FALSE
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

#### 5.2.2.16.5 Usage

Scale an analog signal from a drive.

#### 5.2.2.17.6 Related Functions

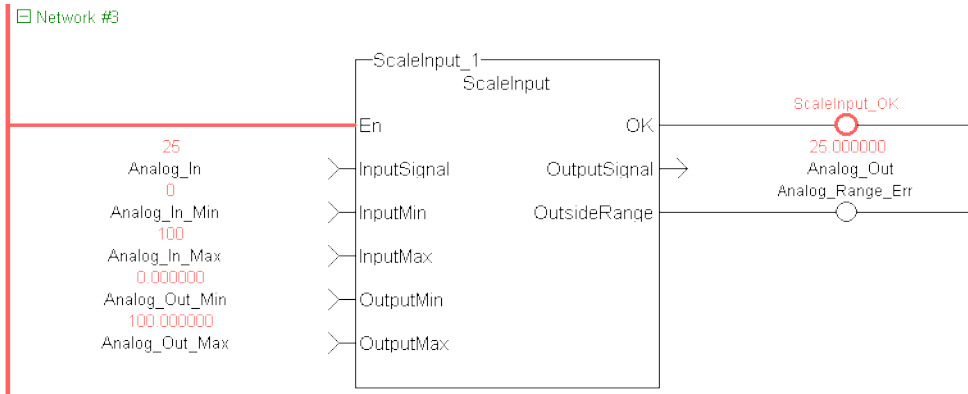
[UDFB ScaleOutput](#)

#### 5.2.2.18.7 Example

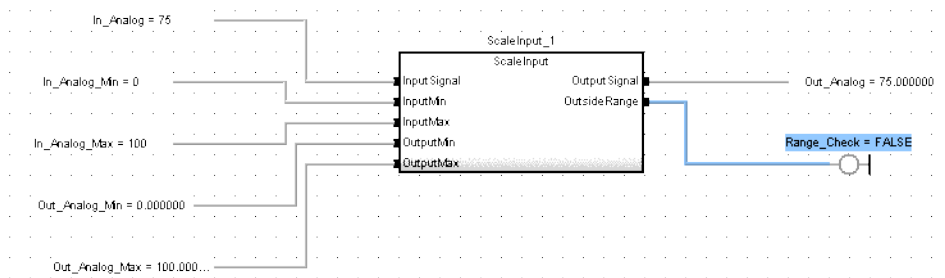
##### 5.2.2.19.8.1 Structured Text

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

##### 5.2.2.20.9.2 Ladder Diagram



### 5.2.2.21.10.3 Function Block Diagram



## 5.2.2.22 FB\_ScaleOutput

### 5.2.2.23.1 Description

Scale LREAL to DINT .

This Kollmorgen UDFB converts un-scaled LREAL values from a PLC Program into units of type DINT that can be mapped to an analog output. The input signal is converted based on a linear mapping automatically calculated by two points entered. InputMin is mapped to OutputMin, InputMax is mapped to OutputMax, and all values in between are scaled automatically. If an input value is not between the selected Min/Max, the Boolean output OutsideRange turns TRUE, and the OutputSignal is set to the corresponding OutputMin or OutputMax value.

The following figure shows the function block I/O:

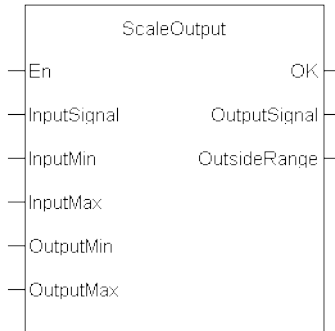


Figure 1-135: Scale Output

### 5.2.2.24.2 Arguments

#### 5.2.2.25.3.1 Input

InputSignal	Description	Un-scaled input signal
	Data type	LREAL
	Range	[0 , 4]
	Unit	n/a
	Default	—
InputMin	Description	Minimum value of accepted input signal range
	Data type	LREAL
	Range	[0 , 4]
	Unit	n/a
	Default	—
InputMax	Description	Maximum value of accepted input signal range
	Data type	LREAL
	Range	[0 , 4]
	Unit	n/a
	Default	—
OutputMin	Description	Output value mapped to the InputMin
	Data type	DINT
	Range	[0 , 4]
	Unit	n/a
	Default	—
OutputMax	Description	Output value mapped to the InputMax
	Data type	DINT
	Range	[0 , 4]

Unit n/a  
 Default —

**5.2.2.26.4.2 Output**

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

**5.2.2.27.5 Usage**

Scale an analog signal to a drive.

**5.2.2.28.6 Related Functions**

[UDFB ScaleInput](#)

**5.2.2.29.7 Example**

**5.2.2.30.8.1 Structured Text**

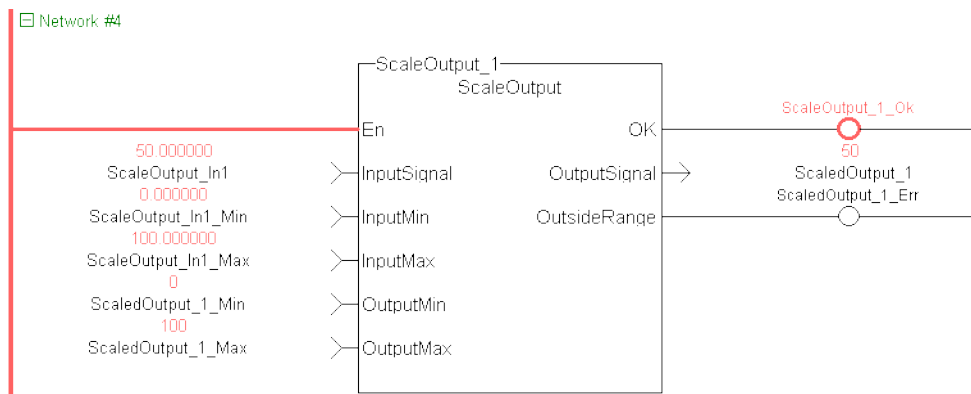
```

Inst_ScaleOutput1( ScaleOutput_In2, ScaleOutput_In2_Min,ScaleOutput_In2_Max, ScaledOutput_2_Min, ScaledOutput_2_Max
);

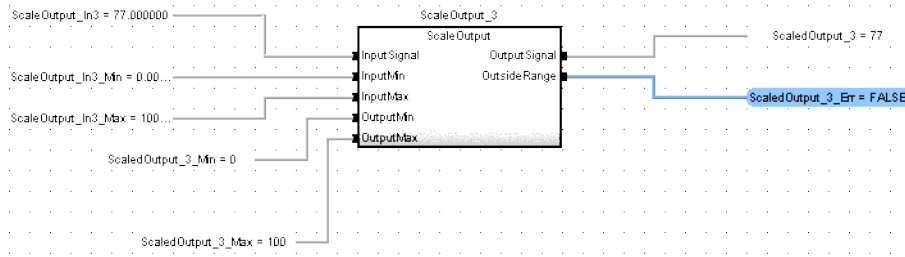
ScaledOutput_2:=Inst_ScaleOutput1.OutputSignal;

ScaledOutput_2_Err:=Inst_ScaleOutput1.OutsideRange;
    
```

**5.2.2.31.9.2 Ladder Diagram**



**5.2.2.32.10.3 Function Block Diagram**



### 5.2.3 FB\_ElapseTime

#### 5.2.3.1 Description

This Kollmorgen UDFB keeps track of the time ( oTotalOnTime) that a Boolean input variable is on. Once the iEN00 enable input is high the Kollmorgen UDFB will keep track of the total time iVariable is on. If iVariable changes to an off state while iEN00 is on, the oTotalOnTime will stop. oTotalOnTime will start to add again once iVariable changes state to high. As long as the iEN00 input is on, iVariable can change states many times. The oTotalOnTime will reflect only the total time that iVariable has been on. While iVariable is still TRUE, oInProcess will also be TRUE and oDone will be FALSE. Once iVariable is FALSE, oInProcess will be FALSE and oDone will be TRUE.

If the iEN00 input goes off, oTotalOnTime stops counting and the Kollmorgen UDFB execution stops. To restart the timer turn iEN00 on again. This will reset oTotalOnTime to zero and counting will begin once iVariable is also on.

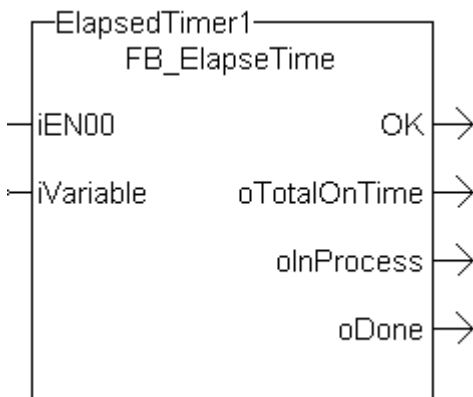


Figure 1-136: FB\_ElapseTime

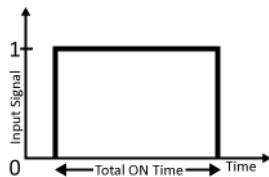


Figure 1-137: MFB\_ElapseTime – Time Diagram

#### 5.2.3.2 Arguments

##### 5.2.3.3.1 Input

<b>iEN00</b>	<b>Description</b>	Enable for the block
	<b>Data type</b>	Boolean
	<b>Range</b>	FALSE or TRUE
	<b>Unit</b>	n/a
	<b>Default</b>	FALSE
<b>iVariable</b>	<b>Description</b>	The variable to be tracked
	<b>Data type</b>	Boolean
	<b>Range</b>	FALSE or TRUE

<b>Unit</b>	n/a
<b>Default</b>	FALSE

#### 5.2.3.4.2 Output

<b>OK</b>	<b>Description</b>	Function Block OK. This output follows the state on iEN00 input
	<b>Data type</b>	Boolean
	<b>Range</b>	FALSE or TRUE
	<b>Unit</b>	n/a
<b>oTotalOnTime</b>	<b>Description</b>	The amount of time the iVariable is turned on.
	<b>Data type</b>	Time
	<b>Range</b>	0ms – 24h
	<b>Unit</b>	ms
<b>oTotalOnTime</b>	<b>Description</b>	The amount of time the iVariable is turned on.
	<b>Data type</b>	Time
	<b>Range</b>	0ms – 24h
	<b>Unit</b>	ms
<b>oInProcess</b>	<b>Description</b>	The state of block's execution whether or not it is still keeping track of time
	<b>Data type</b>	Boolean
	<b>Range</b>	FALSE or TRUE
	<b>Unit</b>	n/a
<b>oDone</b>	<b>Description</b>	The state of block's execution whether or not it is completed
	<b>Data type</b>	Boolean
	<b>Range</b>	FALSE or TRUE
	<b>Unit</b>	n/a

#### 5.2.3.5 Usage

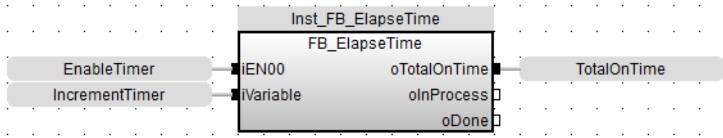
- Enable the block by setting iEN00 to TRUE
- Either manually set iVariable to TRUE or have the application set this variable to TRUE
- Once oDone returns TRUE, read the oTotalOnTime to find out how long iVariable was on.

#### 5.2.3.6 Example

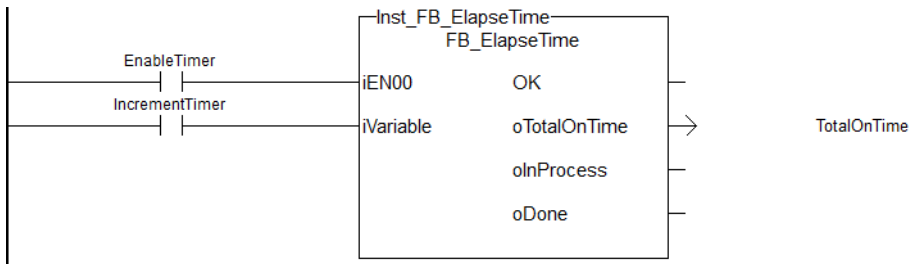
##### 5.2.3.7.1 Structured text

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

### 5.2.3.8.2 Function Block Diagram



### 5.2.3.9.3 Free Form Ladder Diagram





## 5.2.4 PipeNetwork\_FFLD Pipe Network ✓

### 5.2.4.1 Description

This function is used to call the PNCode Function Block in FFLD POU's. It starts and initializes the Pipe Network, based on the command specified by `cmdID`. Internally this function calls the Function Block PNCode.

This is a special function that should only be used in Pipe Network applications that contain FFLD POU's that call PNCode. Calling this function instead of PNCode in FFLD POU's will eliminate the following compile error that occurs after modifying the Pipe Network using the Pipe Network editor.

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

#### NOTE

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

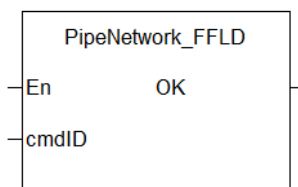


Figure 1-138: PipeNetwork\_FFLD

See also: [Design Motion with Pipe Network](#), [Initialize and Start up a Pipe Network](#), [PLCopen 2-Axes Template with FFLD](#)

### 5.2.4.2 Arguments

#### 5.2.4.3.1 Inputs

<b>En</b>	<b>Description</b>	Request to initialize the Pipe Network
	<b>Data type</b>	BOOL
	<b>Range</b>	0, 1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>cmdID</b>	<b>Description</b>	Commands used to start and initialize the Pipe Network <ul style="list-style-type: none"> <li>• <a href="#">MLPN_CREATE_OBJECTS</a> – Create Pipe Network</li> <li>• <a href="#">MLPN_POWER_ON</a> – Power on all axes</li> <li>• <a href="#">MLPN_POWER_OFF</a> – Power off all axes</li> <li>• <a href="#">MLPN_ACTIVATE</a> – Activate the pipes</li> <li>• <a href="#">MLPN_CONNECT</a> – Connect the axes to the pipes</li> <li>• <a href="#">MLPN_DEACTIVATE</a> – Deactivate the pipes</li> </ul>
	<b>Data type</b>	DINT
	<b>Range</b>	n/a

<b>Unit</b>	n/a
<b>Default</b>	—

#### 5.2.4.4.2 Outputs

<b>OK</b>	<b>Description</b>	Returns TRUE when the function has completed
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

#### 5.2.4.5 Usage

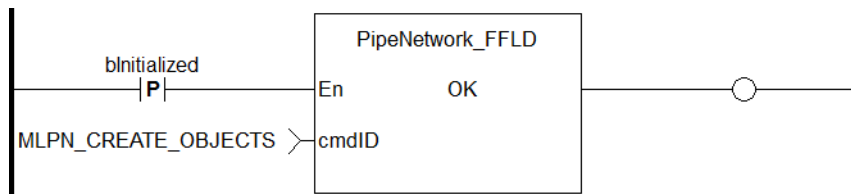
- This is a special function that should only be used in Pipe Network applications that contain FFLD POU's that call PNCode.
- To use this Function, PipeNetwork must be declared as a global variable in the dictionary.

#### **TIP**

The Pipe Network FFLD Application Template is a good example of how to use this Function. See [Pipe Network 2-Axes Template with FFLD only](#).

#### 5.2.4.6 Example

##### 5.2.4.7.1 FFLD



## 5.2.5 ProfilesCode\_FFLD

### 5.2.5.1 Description

This function is used to call the Profiles Code Function Block in FFLD POU's. Internally this function calls the Function Block ProfilesCode.

This is a special function which should only be used in applications that contain FFLD POU's that call ProfilesCode. Calling this function instead of ProfilesCode in FFLD POU's will eliminate the following compile error that occurs after adding a new Profile to the project tree.

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

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

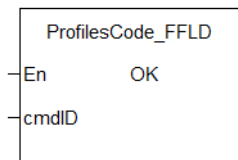


Figure 1-139: ProfilesCode\_FFLD

### 5.2.5.2 Arguments

#### 5.2.5.3.1 Inputs

<b>En</b>	<b>Description</b>	Request to initialize the Pipe Network
	<b>Data type</b>	BOOL
	<b>Range</b>	0,1
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>cmdID</b>	<b>Description</b>	Commands used to start and initialize the Pipe Network <ul style="list-style-type: none"> <li>• <a href="#">MLPR_CREATE_PROFILES</a> - Creation and initialization of profiles.</li> </ul>
	<b>Data type</b>	DINT
	<b>Range</b>	n/a
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 5.2.5.4.2 Outputs

<b>OK</b>	<b>Description</b>	Returns TRUE when the function has completed
	<b>Data type</b>	BOOL

<b>Unit</b>	n/a
-------------	-----

### 5.2.5.5 Usage

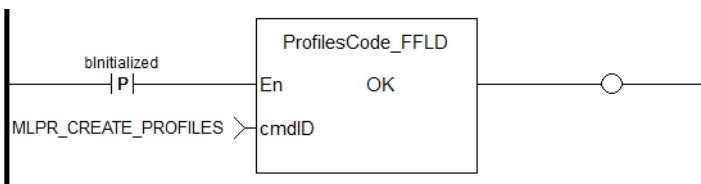
- This is a special function that should only be used in applications that contain FFLD POU's that call ProfilesCode.
- To use this function, Profiles must be declared as a global variable in the dictionary.

**TIP**

The Pipe Network and PLCopen 2 Axis FFLD Application Templates are two examples of how to use this function. See [Pipe Network 2-Axes Template with FFLD only](#) and [PLCopen 2-Axes Template with FFLD](#).

### 5.2.5.6 Example

#### 5.2.5.7.1 FFLD



## 5.2.6 FB\_TemperaturePID

### 5.2.6.1 Description

This function block provides PID temperature control with auto tuning.

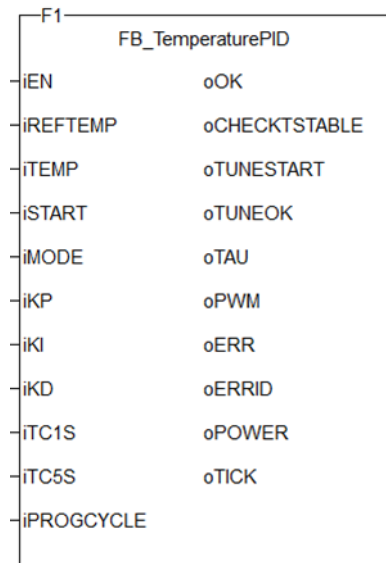


Figure 1-140: The TemperaturePID user-defined function block

### 5.2.6.2 Arguments

#### 5.2.6.3.1 Inputs

<b>iEN</b>	<b>BOOL</b>	Enable function
<b>iREFTEMP</b>	<b>LREAL</b>	Reference temperature [°C]
<b>iTEMP</b>	<b>LREAL</b>	Actual temperature [°C]
<b>iSTART</b>	<b>BOOL</b>	Start PID or auto tuning
<b>iMODE</b>	<b>BOOL</b>	FALSE-automatic, TRUE-tuning
<b>iKP</b>	<b>LREAL</b>	PID Proportional Gain
<b>iKI</b>	<b>LREAL</b>	PID Integral Gain
<b>iKD</b>	<b>LREAL</b>	PID Derivative Gain
<b>iTC1S</b>	<b>BOOL</b>	Sampling Time is 1s
<b>iTC5S</b>	<b>BOOL</b>	Sampling Time is 5s
<b>iPROGCYCLE</b>	<b>LREAL</b>	Execution time of the function [ms]

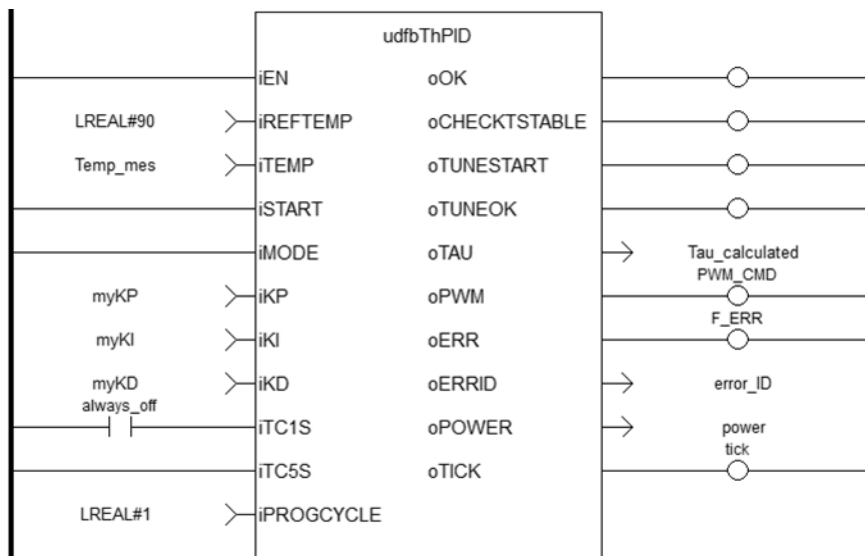
#### 5.2.6.4.2 Outputs

<b>oOK</b>	<b>BOOL</b>	Function enabled
<b>oCHECKSTABLE</b>	<b>BOOL</b>	TRUE when checking if ambient temperature is stable

<b>oTUNESTART</b>	<b>BOOL</b>	Tuning is started
<b>oTUNEOK</b>	<b>BOOL</b>	Tuning is completed
<b>oTAU</b>	<b>LREAL</b>	System Time Constant[s]
<b>oPWM</b>	<b>BOOL</b>	PWM command for heater
<b>oERR</b>	<b>BOOL</b>	Function error
<b>oERRID</b>	<b>INT</b>	Function ID error (in case of oERR=TRUE)
<b>oPOWER</b>	<b>LREAL</b>	% of power requested from heater (100%=full power)
<b>oTICK</b>	<b>BOOL</b>	Pulse every sampling time

### 5.2.6.5 Usage

#### 5.2.6.6.1 Tuning Process



Tuning consists of three steps.

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

After, PID gains are calculated as:

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

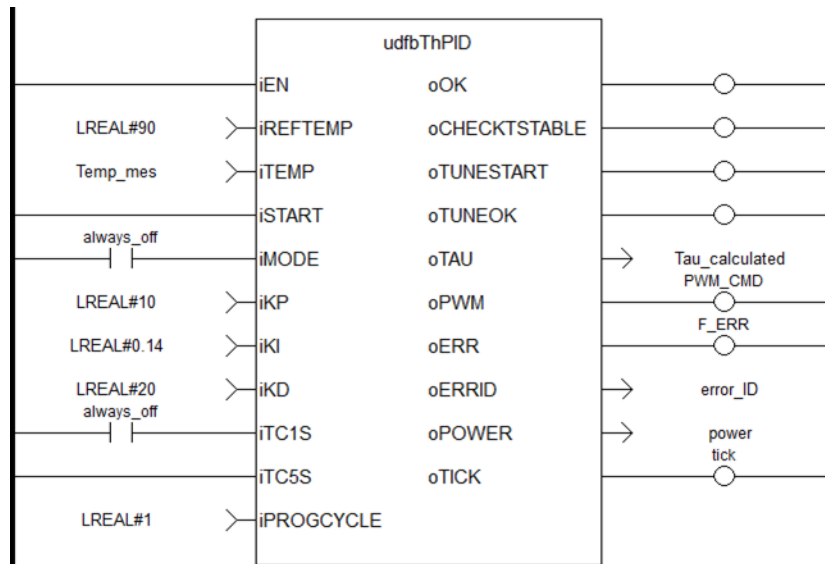
```
i:=SQRT(delta_time)*7
```

The tuning is completed.

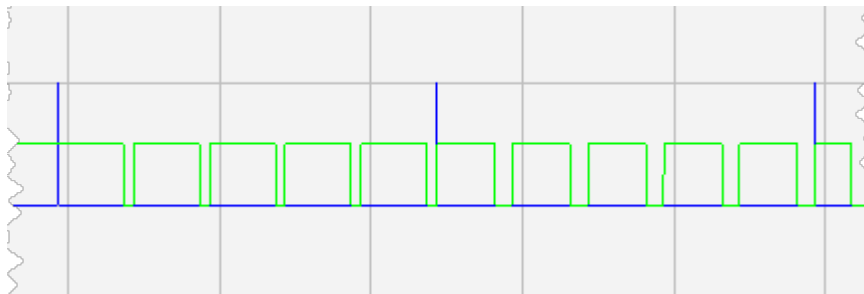
### **TIP**

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

#### 5.2.6.7.2 Start PID Controller



Upon starting the PID controller, the output **oPWM** is modulated 5 times within the sampling time (blue line is **oTICK**, green line is **oPWM**) and each pulse length depends on output **oPOWER** (100%=full length).

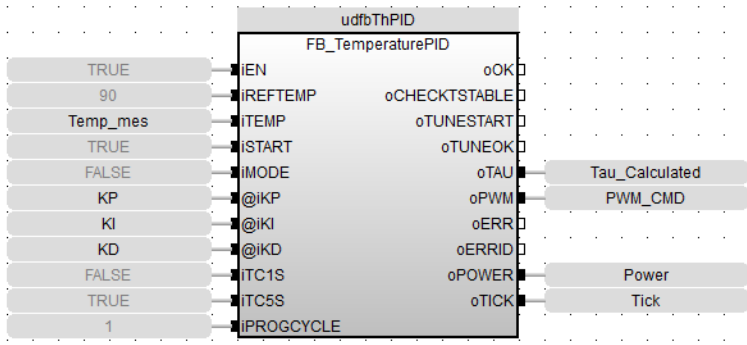


#### 5.2.6.8 Example

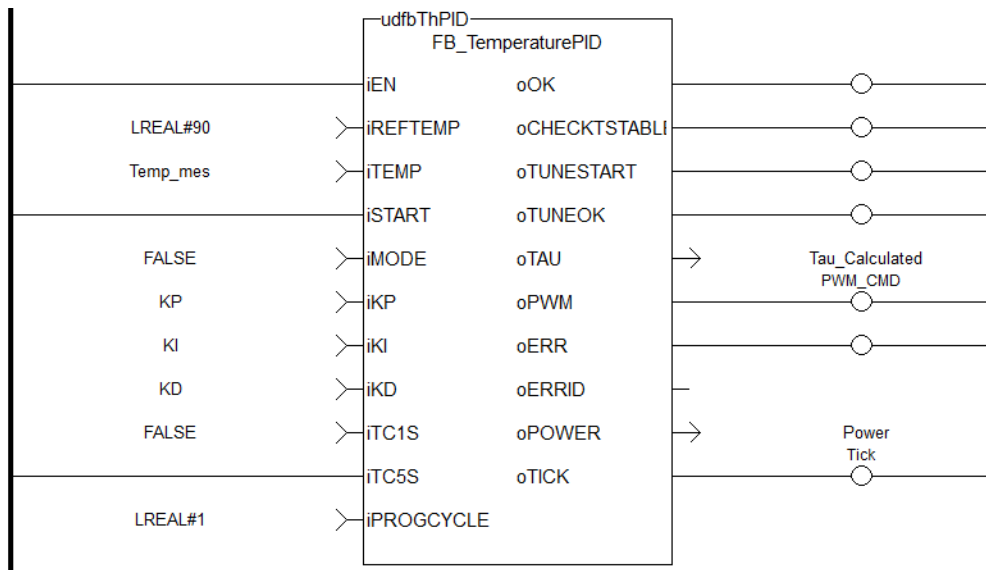
##### 5.2.6.9.1 ST

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

5.2.6.10.2 FBD



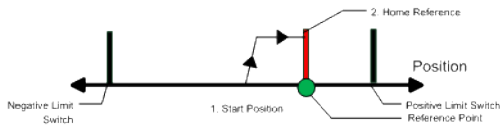
5.2.6.11.3 FFLD





### 5.2.6.12 gl MLFB\_HomeFindHomeInput Pipe Network ✓

#### 5.2.6.13.1 Description



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

#### 5.2.6.14.2 Arguments

##### 5.2.6.15.3.1 Input

<b>ibExecute</b>	<b>Description</b>	Start homing, edge-triggered
	<b>Data type</b>	BOOL
<b>iAxisID</b>	<b>Description</b>	ID of Axis block of Pipe Network
	<b>Data type</b>	DINT
<b>iPosition</b>	<b>Description</b>	Reference position
	<b>Data type</b>	LREAL
<b>ibDirection</b>	<b>Description</b>	0=positive, 1=negative
	<b>Data type</b>	BOOL
<b>iVelocity</b>	<b>Description</b>	Reference speed
	<b>Data type</b>	LREAL
<b>iAcceleration</b>	<b>Description</b>	Reference acceleration
	<b>Data type</b>	LREAL
<b>iDeceleration</b>	<b>Description</b>	Reference deceleration
	<b>Data type</b>	LREAL
<b>ibHomeInput</b>	<b>Description</b>	Home input, high-active
	<b>Data type</b>	BOOL
<b>ibPosLimitSwitch</b>	<b>Description</b>	Positive limit switch, high-active
	<b>Data type</b>	BOOL

<b>ibNegLimitSwitch</b>	<b>Description</b>	Negative limit switch, high-active
	<b>Data type</b>	BOOL
<b>iTimeout</b>	<b>Description</b>	Time monitoring (T#0ms: off)
	<b>Data type</b>	TIME

#### 5.2.6.16.4.2 Output

<b>obDone</b>	<b>Description</b>	Done bit												
	<b>Data type</b>	BOOL												
<b>obActive</b>	<b>Description</b>	Active bit												
	<b>Data type</b>	BOOL												
<b>obError</b>	<b>Description</b>	Error bit												
	<b>Data type</b>	BOOL												
<b>oErrorID</b>	<b>Description</b>	Error identifier, see list here												
		<table border="1"> <thead> <tr> <th>ErrorID</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Axis in error</td> </tr> <tr> <td>2</td> <td>Axis not enabled</td> </tr> <tr> <td>3</td> <td>Timeout expired</td> </tr> <tr> <td>4</td> <td>SDO read/write error</td> </tr> <tr> <td>5</td> <td>Input parameter out of range</td> </tr> </tbody> </table>	ErrorID	Description	1	Axis in error	2	Axis not enabled	3	Timeout expired	4	SDO read/write error	5	Input parameter out of range
ErrorID	Description													
1	Axis in error													
2	Axis not enabled													
3	Timeout expired													
4	SDO read/write error													
5	Input parameter out of range													
	<b>Data type</b>	DINT												

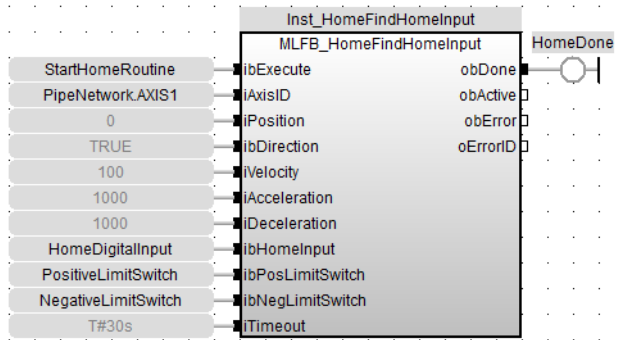
#### 5.2.6.17.5 Example

#### 5.2.6.18.6.1 ST

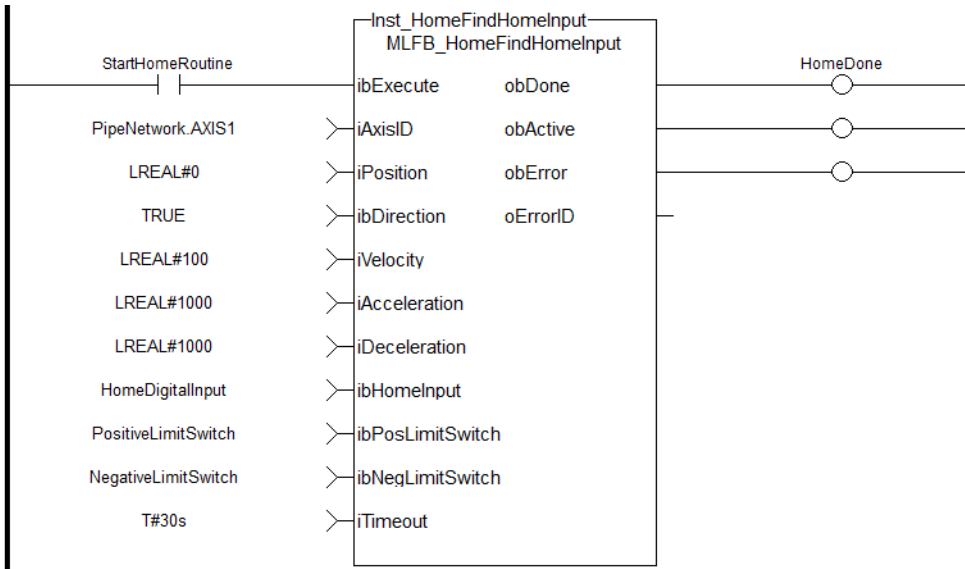
```
//Call homing function on Axis1 with preset velocity, accel, and
decel values
//Start in negative direction, change if limit switch seen
before home switch
//after seeing home switch, set axis position to zero
Inst_MLFB_HomeFindHomeInput( StartHomeRoutine,
    PipeNetwork.AXIS1,
    0,
    TRUE,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

```
HomeDone := Inst_MLFB_HomeFindHomeInput.obDone;
```

### 5.2.6.19.7.2 Function Block Diagram

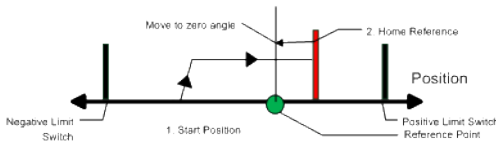


### 5.2.6.20.8.3 FFLD



**5.2.6.21 MLFB\_HomeFindHomeInputThenZeroAngle** Pipe Network ✓

**5.2.6.22.1 Description**



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

**5.2.6.23.2 Arguments**

**5.2.6.24.3.1 Input**

<b>ibExecute</b>	<b>Description</b>	Start homing, edge-triggered
	<b>Data type</b>	BOOL
<b>iAxisID</b>	<b>Description</b>	ID of Axis block of Pipe Network
	<b>Data type</b>	DINT
<b>iPosition</b>	<b>Description</b>	Reference position
	<b>Data type</b>	LREAL
<b>ibDirection</b>	<b>Description</b>	0=positive, 1=negative
	<b>Data type</b>	BOOL
<b>iVelocity</b>	<b>Description</b>	Reference speed
	<b>Data type</b>	LREAL
<b>iAcceleration</b>	<b>Description</b>	Reference acceleration
	<b>Data type</b>	LREAL
<b>iDeceleration</b>	<b>Description</b>	Reference deceleration
	<b>Data type</b>	LREAL
<b>ibHomeInput</b>	<b>Description</b>	Home input, high-active
	<b>Data type</b>	BOOL
<b>ibPosLimitSwitch</b>	<b>Description</b>	Positive limit switch, high-active
	<b>Data type</b>	BOOL
<b>ibNegLimitSwitch</b>	<b>Description</b>	Negative limit switch, high-active
	<b>Data type</b>	BOOL
<b>iTimeout</b>	<b>Description</b>	Time monitoring (T#0ms: off)
	<b>Data type</b>	TIME

## 5.2.6.25.4.2 Output

<b>obDone</b>	<b>Description</b>	Done bit												
	<b>Data type</b>	BOOL												
<b>obActive</b>	<b>Description</b>	Active bit												
	<b>Data type</b>	BOOL												
<b>obError</b>	<b>Description</b>	Error bit												
	<b>Data type</b>	BOOL												
<b>oErrorID</b>	<b>Description</b>	Error identifier, see list here												
		<table border="1"> <thead> <tr> <th>ErrorID</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Axis in error</td> </tr> <tr> <td>2</td> <td>Axis not enabled</td> </tr> <tr> <td>3</td> <td>Timeout expired</td> </tr> <tr> <td>4</td> <td>SDO read/write error</td> </tr> <tr> <td>5</td> <td>Input parameter out of range</td> </tr> </tbody> </table>	ErrorID	Description	1	Axis in error	2	Axis not enabled	3	Timeout expired	4	SDO read/write error	5	Input parameter out of range
ErrorID	Description													
1	Axis in error													
2	Axis not enabled													
3	Timeout expired													
4	SDO read/write error													
5	Input parameter out of range													
	<b>Data type</b>	DINT												

## 5.2.6.26.5 Example

## 5.2.6.27.6.1 ST

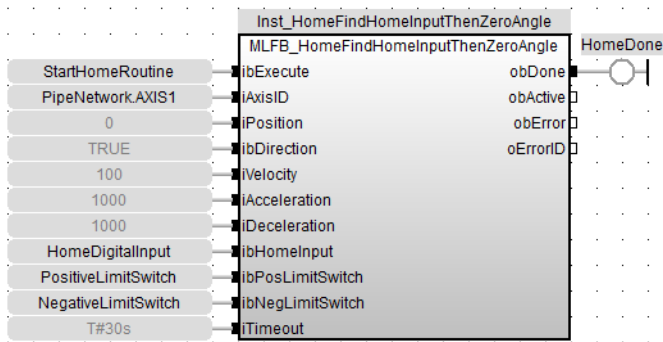
```

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

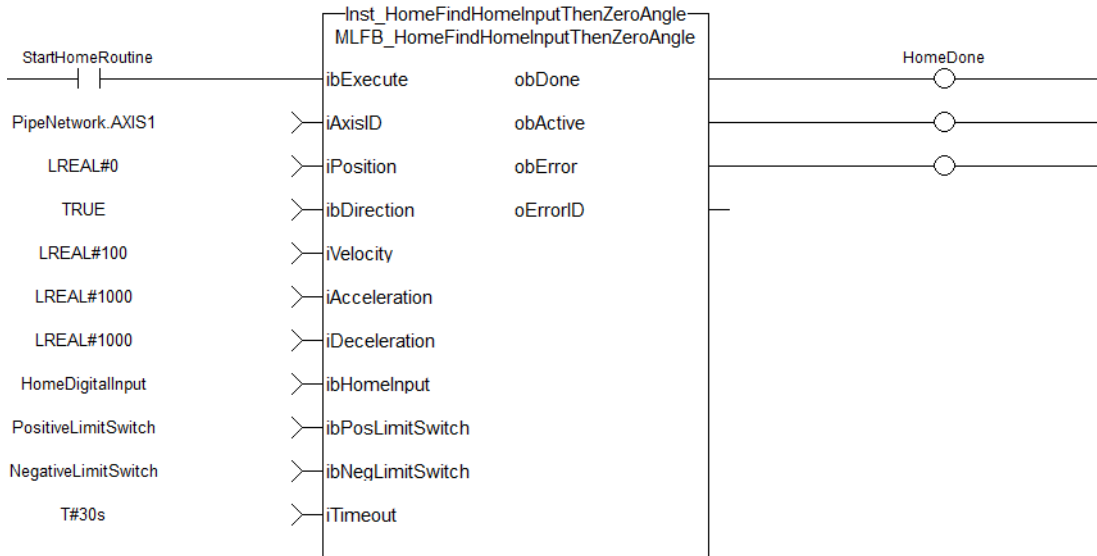
HomeDone := Inst_MLFB_HomeFindHomeInputThenZeroAngle.obDone;

```

## 5.2.6.28.7.2 Function Block Diagram

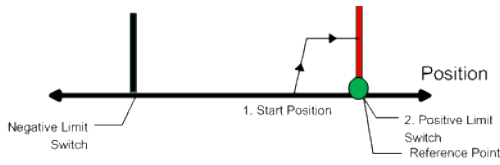


### 5.2.6.29.8.3 FFLD



**5.2.6.30 MLFB\_HomeFindLimitInput** Pipe Network ✓

**5.2.6.31.1 Description**



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

**5.2.6.32.2 Arguments**

**5.2.6.33.3.1 Input**

ibExecute	Description Data type	Start homing, edge-triggered BOOL
iAxisID	Description Data type	ID of Axis block of Pipe Network DINT
iPosition	Description Data type	Reference position LREAL
ibDirection	Description Data type	0=positive, 1=negative BOOL
iVelocity	Description Data type	Reference speed LREAL
iAcceleration	Description Data type	Reference acceleration LREAL
iDeceleration	Description Data type	Reference deceleration LREAL
ibLimitSwitch	Description Data type	Pos. or neg. limit switch, high-active (depends on ibDirection) BOOL
iTimeout	Description Data type	Time monitoring (T#0ms: off) TIME

**5.2.6.34.4.2 Output**

obDone	Description Data type	Done bit BOOL
obActive	Description Data type	Active bit BOOL
obError	Description Data type	Error bit BOOL

Error identifier, see list here

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

oErrorID                      Description

Data type DINT

5.2.6.35.5 Example

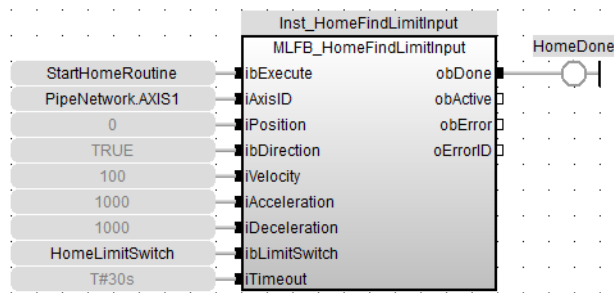
5.2.6.36.6.1 ST

```

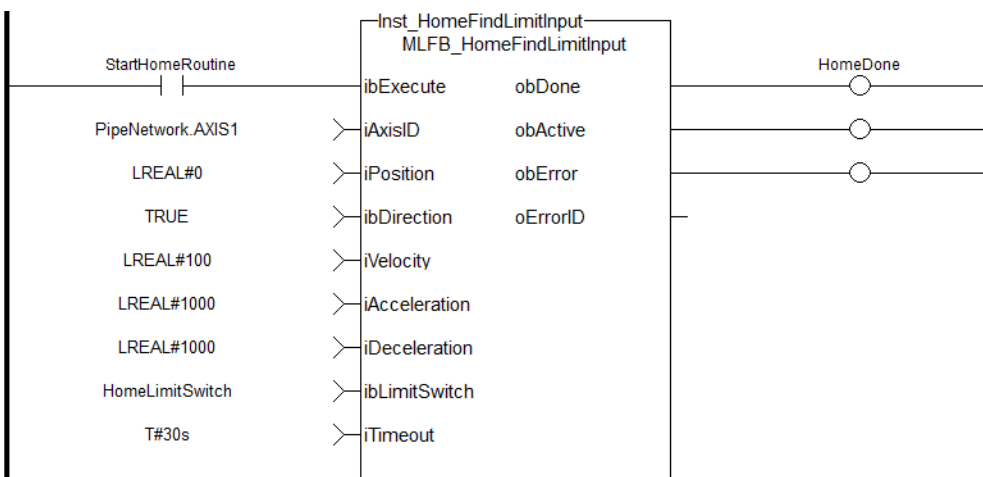
//Call homing function on Axis1 with preset velocity, accel, and
//decel values
//Start in negative direction and stop when axis hits limit
//switch or times out
//after seeing limit switch, set axis position to zero
Inst_MLFB_HomeFindLimitInput( StartHomeRoutine,
    PipeNetwork.AXIS1,
    0,
    TRUE,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

HomeDone := Inst_MLFB_HomeFindLimitInput.obDone;
    
```

5.2.6.37.7.2 Function Block Diagram



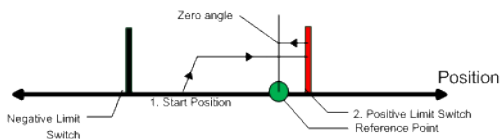
5.2.6.38.8.3 FFLD





## 5.2.6.39 MLFB\_HomeFindLimitInputThenZeroAngle Pipe Network ✓

### 5.2.6.40.1 Description



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

### 5.2.6.41.2 Arguments

#### 5.2.6.42.3.1 Input

ibExecute	Description Data type	Start homing, edge-triggered BOOL
iAxisID	Description Data type	ID of Axis block of Pipe Network BOOL
iPosition	Description Data type	Reference position BOOL
ibDirection	Description Data type	0=positive, 1=negative BOOL
iVelocity	Description Data type	Reference speed BOOL
iAcceleration	Description Data type	Reference acceleration BOOL
iDeceleration	Description Data type	Reference deceleration BOOL
ibLimitSwitch	Description Data type	Pos. or neg. limit switch, high-active (depends on ibDirection) BOOL
iTimeout	Description Data type	Time monitoring (T#0ms: off) BOOL

#### 5.2.6.43.4.2 Output

obDone	Description Data type	Done bit BOOL
obActive	Description Data type	Active bit BOOL
obError	Description Data type	Error bit BOOL

Error identifier, see list here

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

Data type DINT

**5.2.6.44.5 Example**

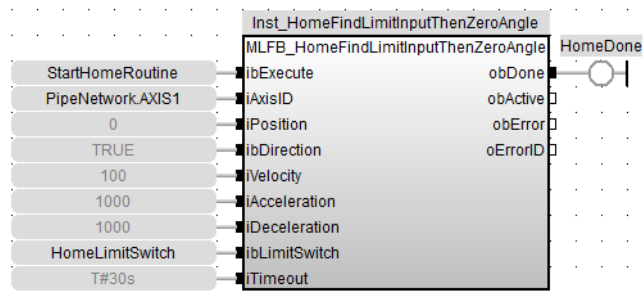
**5.2.6.45.6.1 ST**

```

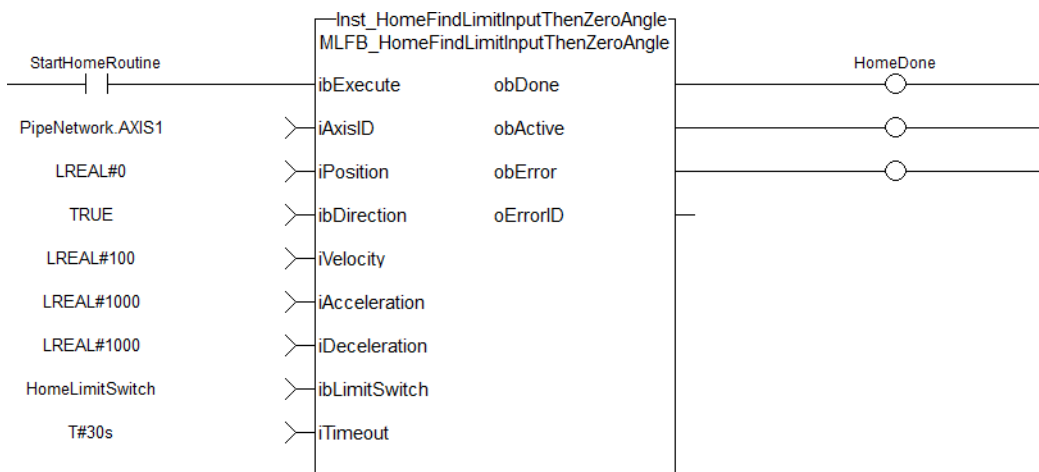
//Call homing function on Axis1 with preset velocity, accel, and
//decel values
//Start in negative direction and stop when axis hits limit
//switch or times out
//after seeing limit switch, moves to zero angle and set axis
//position to zero
Inst_MLFB_HomeFindLimitInputThenZeroAngle( StartHomeRoutine,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

HomeDone := Inst_MLFB_HomeFindLimitInputThenZeroAngle.obDone;
    
```

**5.2.6.46.7.2 Function Block Diagram**

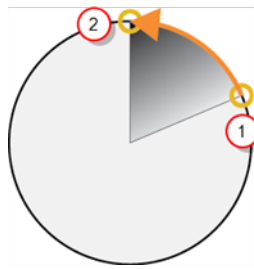


**5.2.6.47.8.3 FFLD**



### 5.2.6.48 MLFB\_HomeFindZeroAngle Pipe Network ✓

#### 5.2.6.49.1 Description



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

Figure 1-141: Mode to find the zero angle reference of the motor.

#### NOTE

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

#### 5.2.6.50.2 Arguments

##### 5.2.6.51.3.1 Input

<b>ibExecute</b>	<b>Description</b>	Start homing, edge-triggered
	<b>Data type</b>	BOOL
<b>iAxisID</b>	<b>Description</b>	ID of Axis block of Pipe Network
	<b>Data type</b>	DINT
<b>iPosition</b>	<b>Description</b>	Reference position
	<b>Data type</b>	LREAL
<b>iDirectionType</b>	<b>Description</b>	0=positive, 1=negative, 2=shortest
	<b>Data type</b>	DINT
<b>iVelocity</b>	<b>Description</b>	Reference speed
	<b>Data type</b>	LREAL
<b>iAcceleration</b>	<b>Description</b>	Reference acceleration
	<b>Data type</b>	LREAL
<b>iDeceleration</b>	<b>Description</b>	Reference deceleration
	<b>Data type</b>	LREAL
<b>iTimeout</b>	<b>Description</b>	Time monitoring (T#0ms: off)
	<b>Data type</b>	TIME

##### 5.2.6.52.4.2 Output

<b>obDone</b>	<b>Description</b>	Done bit
---------------	--------------------	----------

	<b>Data type</b>	BOOL												
<b>obActive</b>	<b>Description</b>	Active bit												
	<b>Data type</b>	BOOL												
<b>obError</b>	<b>Description</b>	Error bit												
	<b>Data type</b>	BOOL												
<b>oErrorID</b>	<b>Description</b>	Error identifier, see list here												
		<table border="1"> <thead> <tr> <th>ErrorID</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Axis in error</td> </tr> <tr> <td>2</td> <td>Axis not enabled</td> </tr> <tr> <td>3</td> <td>Timeout expired</td> </tr> <tr> <td>4</td> <td>SDO read/write error</td> </tr> <tr> <td>5</td> <td>Input parameter out of range</td> </tr> </tbody> </table>	ErrorID	Description	1	Axis in error	2	Axis not enabled	3	Timeout expired	4	SDO read/write error	5	Input parameter out of range
ErrorID	Description													
1	Axis in error													
2	Axis not enabled													
3	Timeout expired													
4	SDO read/write error													
5	Input parameter out of range													
	<b>Data type</b>	DINT												

**5.2.6.53.5 Example**

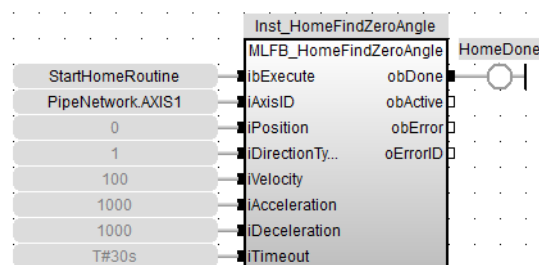
**5.2.6.54.6.1 ST**

```

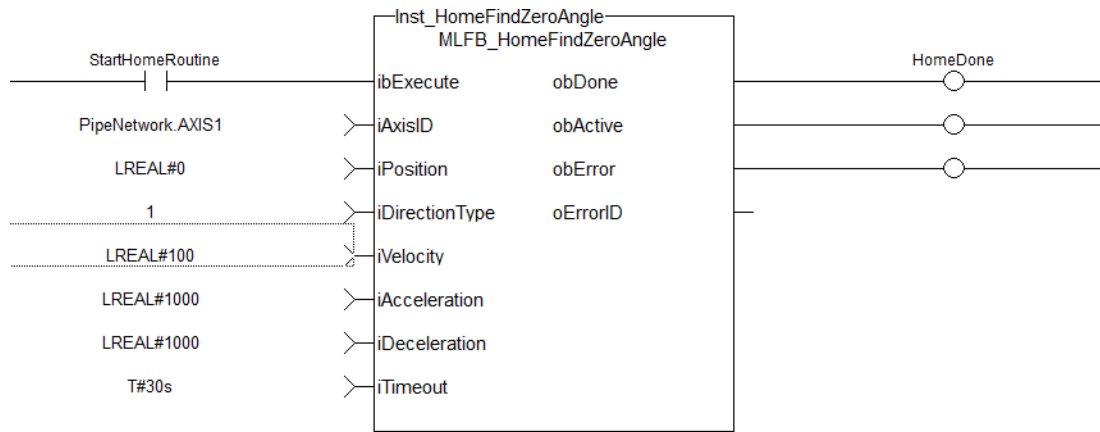
//Call homing function on Axis1 with preset velocity, accel, and
decel values
//Start in negative direction and go to zero angle or time out
//after reaching zero angle set axis position to zero
Inst_MLFB_HomeFindZeroAngle( StartHomeRoutine,
PipeNetwork.AXIS1,
0,
1,
100,
1000,
1000,
T#30s );

HomeDone := Inst_MLFB_HomeFindZeroAngle.obDone;
    
```

**5.2.6.55.7.2 Function Block Diagram**

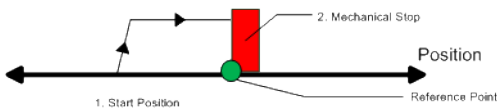


**5.2.6.56.8.3 FFLD**



**5.2.6.57 MLFB\_HomeMoveUntilPosErrExceeded** Pipe Network ✓

**5.2.6.58.1 Description**



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

**5.2.6.59.2 Arguments**

**5.2.6.60.3.1 Input**

ibExecute	Description Data type	Start homing, edge-triggered BOOL
iAxisID	Description Data type	ID of Axis block of Pipe Network DINT
iPosition	Description Data type	Reference position LREAL
ibDirection	Description Data type	0=positive, 1=negative BOOL
iVelocity	Description Data type	Reference speed LREAL
iAcceleration	Description Data type	Reference acceleration LREAL
iDeceleration	Description Data type	Reference deceleration LREAL
iMaxPositionError	Description Data type	Maximum position error LREAL
iPeakCurrent	Description Data type	Peak current in mA DINT
iTimeout	Description Data type	Time monitoring (T#0ms: off) TIME

**5.2.6.61.4.2 Output**

obDone	Description Data type	Done bit BOOL
obActive	Description Data type	Active bit BOOL
obError	Description Data type	Error bit BOOL

		Error identifier, see list here	
		ErrorID	Description
oErrorID	Description	1	Axis in error
		2	Axis not enabled
		3	Timeout expired
		4	SDO read/write error
		5	Input parameter out of range
	Data type	DINT	

**5.2.6.62.5 Example**

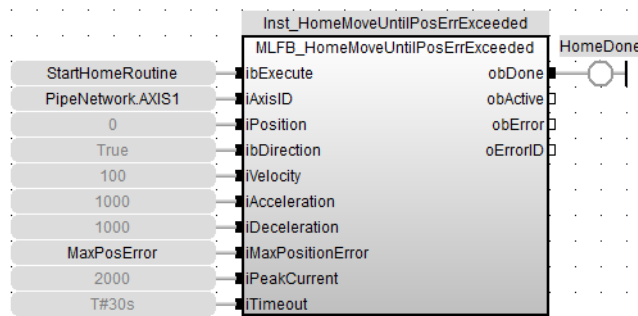
**5.2.6.63.6.1 ST**

```

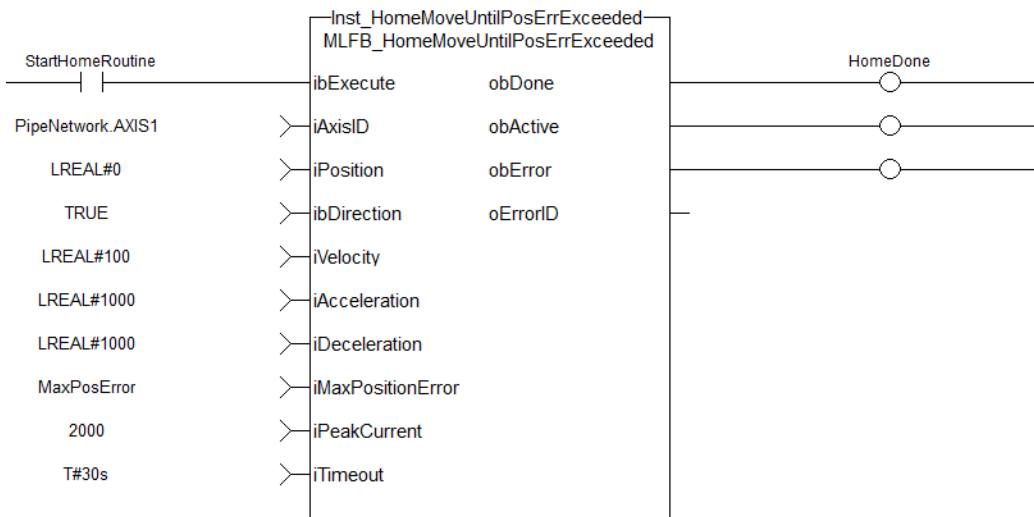
//Call homing function on Axis1 with preset velocity, accel, and
//decel values
//Start in negative direction and go until position error
//exceeds input value or time out
//afterwards set axis position to zero
//function block temporarily writes new max current value to 2
//Amp while home routine active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

HomeDone := Inst_MLFB_HomeMoveUntilPosErrExceeded.obDone;
    
```

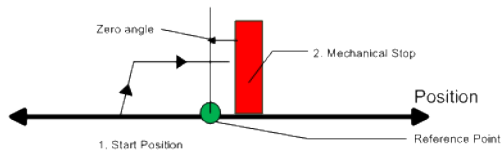
**5.2.6.64.7.2 Function Block Diagram**



**5.2.6.65.8.3 FFLD**





**5.2.6.66 MLFB\_HomeMoveUntilPosErrExceededThenZeroAngle** Pipe Network ✓**5.2.6.67.1 Description**

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

**5.2.6.68.2 Arguments****5.2.6.69.3.1 Input**

ibExecute	Description Data type	Start homing, edge-triggered BOOL
iAxisID	Description Data type	ID of Axis block of Pipe Network DINT
iPosition	Description Data type	Reference position LREAL
ibDirection	Description Data type	0=positive, 1=negative BOOL
iVelocity	Description Data type	Reference speed LREAL
iAcceleration	Description Data type	Reference acceleration LREAL
iDeceleration	Description Data type	Reference deceleration LREAL
iMaxPositionError	Description Data type	Maximum position error LREAL
iPeakCurrent	Description Data type	Peak current in mA DINT
iTimeout	Description Data type	Time monitoring (T#0ms: off) TIME

**5.2.6.70.4.2 Output**

obDone	Description Data type	Done bit BOOL
obActive	Description Data type	Active bit BOOL
obError	Description Data type	Error bit BOOL

		Error identifier, see list here	
oErrorID	Description	ErrorID	Description
		1	Axis in error
		2	Axis not enabled
		3	Timeout expired
		4	SDO read/write error
		5	Input parameter out of range
	Data type	DINT	

**5.2.6.71.5 Example**

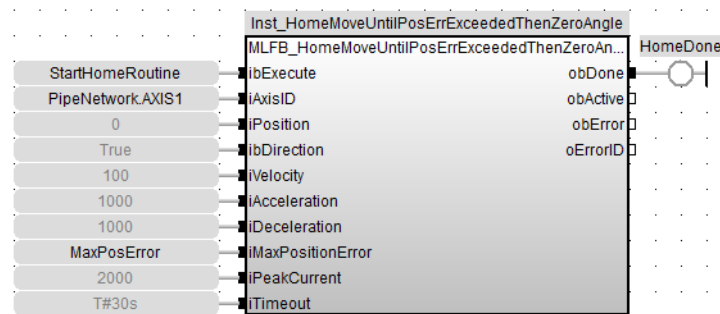
**5.2.6.72.6.1 ST**

```

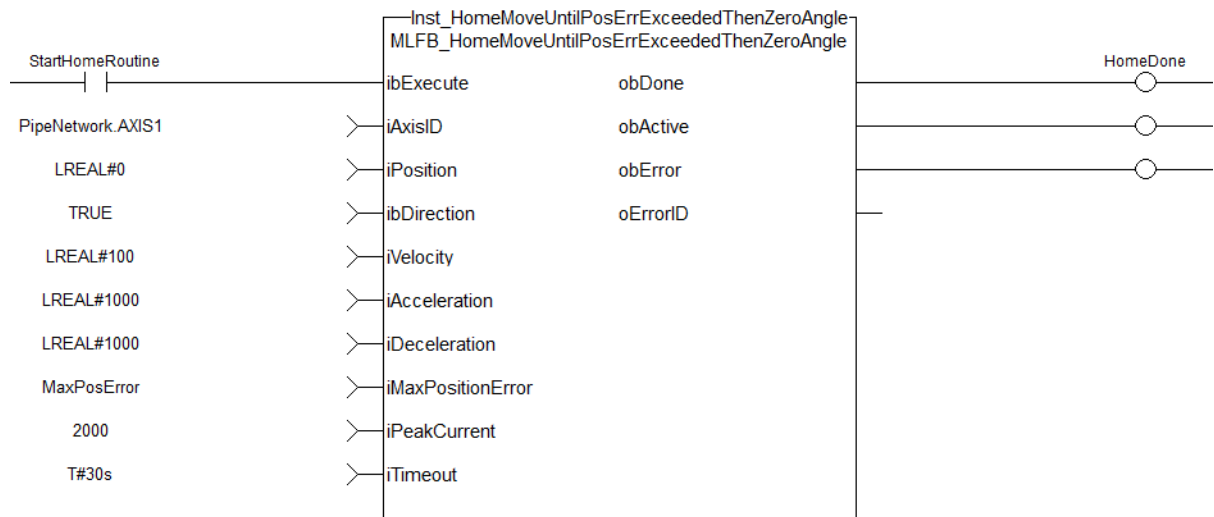
//Call homing function on Axis1 with preset velocity, accel, and
//decel values
//Start in negative direction and go until position error
//exceeds input value or time out
//afterterwards moves to zero angle and sets axis position to
//zero
//function block temporarily writes new max current value to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

HomeDone := Inst_MLFB_
HomeMoveUntilPosErrExceededThenZeroAngle.obDone;
    
```

**5.2.6.73.7.2 Function Block Diagram**



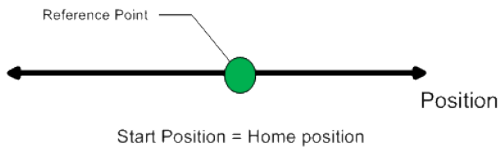
**5.2.6.74.8.3 FFLD**



### 5.2.6.75

MLFB\_HomeUsingCurrentPosition Pipe Network ✓

#### 5.2.6.76.1 Description



Using the current position is the most basic homing method. This method simply uses the current position of the motor as the home point reference.

You can use this parameter to set the value of the home position other than zero. This allows you to offset your home reference away from zero.

#### 5.2.6.77.2 Arguments

##### 5.2.6.78.3.1 Input

ibExecute	Description Data type	Start homing, edge-triggered BOOL
iAxisID	Description Data type	ID of Axis block of Pipe Network DINT
iPosition	Description Data type	Reference position LREAL

##### 5.2.6.79.4.2 Output

obDone	Description Data type	Done bit BOOL
obActive	Description Data type	Active bit BOOL
obError	Description Data type	Error bit BOOL

Error identifier, see list here

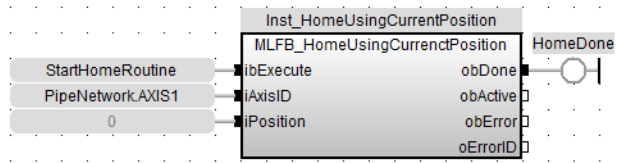
ErrorID	Description
1	Axis in error
2	Axis not enabled
3	Timeout expired
4	SDO read/write error
5	Input parameter out of range

#### 5.2.6.80.5 Example

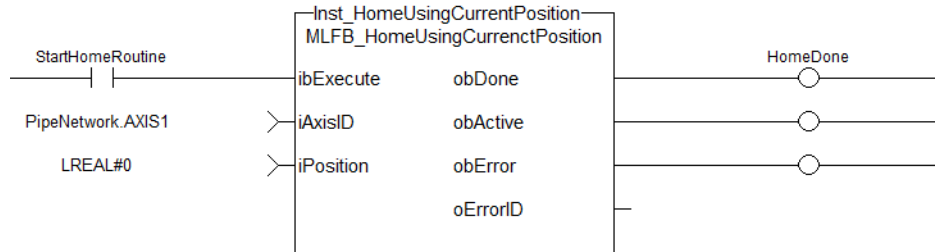
##### 5.2.6.81.6.1 ST

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

### 5.2.6.82.7.2 Function Block Diagram



### 5.2.6.83.8.3 FFLD



**5.2.6.84 MLFB\_HomeFindHomeFastInput** Pipe Network ✓

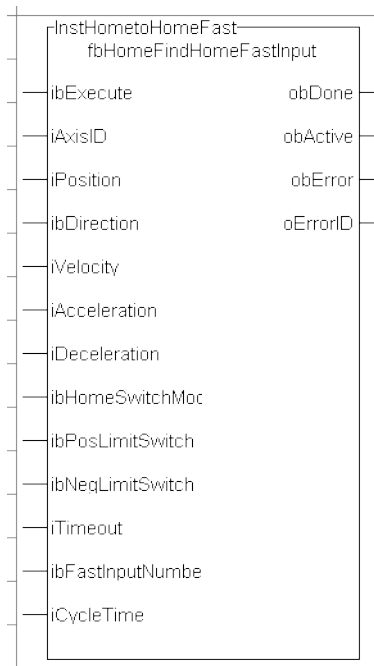
**5.2.6.85.1 Description**

This function block performs a single-axis home to a limit switch connected to a High Speed Input. The motor starts to move according to the direction setting. The home position has been found as soon as the fast input selected is triggered on the edge selected.

An absolute move is made to the triggered position, and then the position value is set. The hardware limit switches are monitored during the homing procedure.

The drive behaves as follows in case that a hardware limit switch is active before the home-switch has been activated: The motor changes the direction until the home switch is crossed.

The following figure shows the function block I/O:



**Figure 1-142:** MLFB HomeFindHomeFastInput

**5.2.6.86.2 Arguments**

**5.2.6.87.3.1 Input**

<b>ibExecute</b>	<b>Description</b>	Request the homing step procedure at rising edge
	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>iAxisID</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1 , 256]
	<b>Unit</b>	n/a

	<b>Default</b>	—						
<b>iPosition</b>	<b>Description</b>	Offset Position Applied After Home Switch is found						
	<b>Data type</b>	LREAL						
	<b>Range</b>	—						
	<b>Unit</b>	User unit						
	<b>Default</b>	—						
<b>ibDirection</b>	<b>Description</b>	Define the axis homing direction						
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>clockwise rotation</td> </tr> <tr> <td>1</td> <td>counterclockwise rotation</td> </tr> </tbody> </table>	Value	Description	0	clockwise rotation	1	counterclockwise rotation
Value	Description							
0	clockwise rotation							
1	counterclockwise rotation							
	<b>Data type</b>	BOOL						
	<b>Range</b>	[0, 1]						
	<b>Unit</b>	n/a						
	<b>Default</b>	—						
<b>iVelocity</b>	<b>Description</b>	Commanded velocity for the homing move						
	<b>Data type</b>	LREAL						
	<b>Range</b>	—						
	<b>Unit</b>	User unit/sec						
	<b>Default</b>	—						
<b>iAcceleration</b>	<b>Description</b>	Commanded acceleration for the homing move						
	<b>Data type</b>	LREAL						
	<b>Range</b>	—						
	<b>Unit</b>	User unit/sec <sup>2</sup>						
	<b>Default</b>	—						
<b>iDeceleration</b>	<b>Description</b>	Commanded deceleration for the homing move						
	<b>Data type</b>	LREAL						
	<b>Range</b>	—						
	<b>Unit</b>	User unit/sec <sup>2</sup>						
	<b>Default</b>	—						

<b>ibHomeSwitchMode</b>	<b>Description</b>	Limit switch state to complete homing						
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Rising edge of switch</td> </tr> <tr> <td>1</td> <td>Falling edge of switch</td> </tr> </tbody> </table>	Value	Description	0	Rising edge of switch	1	Falling edge of switch
	Value	Description						
	0	Rising edge of switch						
	1	Falling edge of switch						
<b>Data type</b>	BOOL							
<b>Range</b>	[0 , 1]							
<b>Unit</b>	n/a							
	<b>Default</b>	—						
<b>ibPosLimitSwitch</b>	<b>Description</b>	The positive direction limit switch input I/O point						
	<b>Data type</b>	BOOL						
	<b>Range</b>	[0 , 1]						
	<b>Unit</b>	n/a						
	<b>Default</b>	—						
<b>ibNegLimitSwitch</b>	<b>Description</b>	The negative direction limit switch input I/O point						
	<b>Data type</b>	BOOL						
	<b>Range</b>	[0 , 1]						
	<b>Unit</b>	n/a						
	<b>Default</b>	—						
<b>iTimeout</b>	<b>Description</b>	Maximum time for homing move to complete. If exceeded the homing procedure will error out. 0= no time limit						
	<b>Data type</b>	TIME						
	<b>Range</b>	—						
	<b>Unit</b>	sec						
	<b>Default</b>	—						
<b>ibFastInputNumber</b>	<b>Description</b>	Limit switch state to complete homing.						
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Fast Input Number 1</td> </tr> <tr> <td>1</td> <td>Fast Input Number 2</td> </tr> </tbody> </table>	Value	Description	0	Fast Input Number 1	1	Fast Input Number 2
	Value	Description						
	0	Fast Input Number 1						
	1	Fast Input Number 2						
<b>Data type</b>	BOOL							
<b>Range</b>	[0 , 1]							
<b>Unit</b>	n/a							



	<b>Default</b>	—
<b>iCycleTime</b>	<b>Description</b>	Ethercat Cycle Time 250, 500 or 1000
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	microseconds
	<b>Default</b>	—

#### 5.2.6.88.4.2 Output

<b>obDone</b>	<b>Description</b>	Indicates the move completed successfully. The Command Position has reached the endpoint												
	<b>Data type</b>	BOOL												
	<b>Unit</b>	n/a												
<b>obActive</b>	<b>Description</b>	Indicates this move is the active move												
	<b>Data type</b>	BOOL												
	<b>Unit</b>	n/a												
<b>obError</b>	<b>Description</b>	Indicates an invalid input was specified or the move was terminated due to an error												
	<b>Data type</b>	BOOL												
	<b>Unit</b>	n/a												
<b>oErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE												
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Axis in Error State</td> </tr> <tr> <td>2</td> <td>Axis is Not Enabled</td> </tr> <tr> <td>3</td> <td>Timeout Exceeded</td> </tr> <tr> <td>4</td> <td>SDO Read/Write Error</td> </tr> <tr> <td>5</td> <td>Input Parameter out of Range</td> </tr> </tbody> </table>	Value	Description	1	Axis in Error State	2	Axis is Not Enabled	3	Timeout Exceeded	4	SDO Read/Write Error	5	Input Parameter out of Range
Value	Description													
1	Axis in Error State													
2	Axis is Not Enabled													
3	Timeout Exceeded													
4	SDO Read/Write Error													
5	Input Parameter out of Range													
	<b>Data type</b>	DINT												
	<b>Unit</b>	n/a												

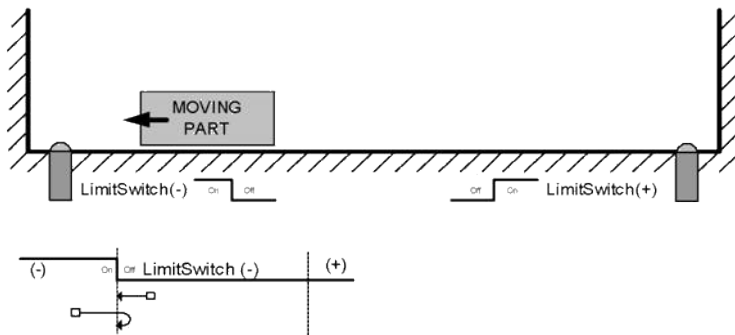
#### 5.2.6.89.5 Usage

This homing procedure performs a homing function searching for sensor using only High Speed Input Switches. (A High Speed Limit Switch has 1 “Off” (or “On”) area).

- Home is commanded by user in the desired homing direction at the selected Velocity
- If LimitSwitch is found ‘On’ on rising ‘Execute’, then the process is started in the opposite direction as specified, LimitSwitch is search for ‘Off’ (or On, depending on LimitSwitchMode setting) Edge

(released), and process is restarted again in original direction. This ensures that the end conditions are always the same

- The Timeout can cause an error if exceeded



### 5.2.6.90.6 Related Functions

[MLFB\\_HomeFindHomeFastInputModulo](#)

[MLFB\\_HomeFindLimitFastInput](#)

[MLFB\\_HomeFindLimitFastInputModulo](#)

### 5.2.6.91.7 Example

### 5.2.6.92.8.1 Structured Text

```

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

```

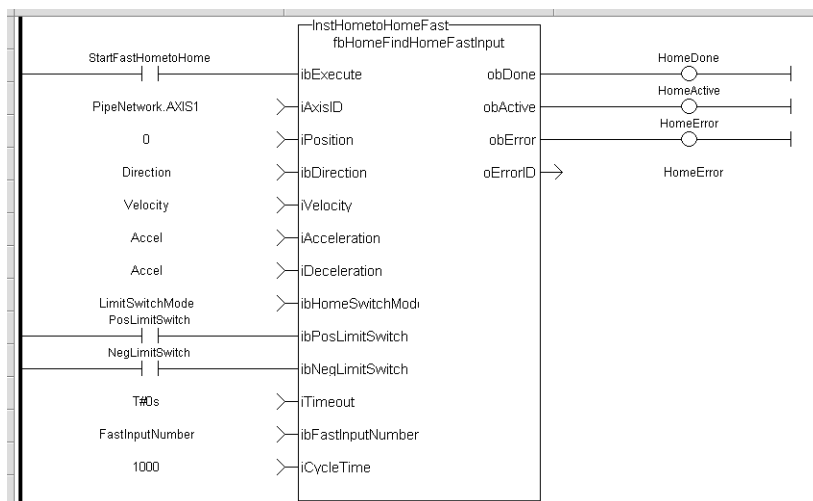
inst_fbHomeFindHomeFastInput(True, Axis1, Position, Direction,
Velocity, Acceleration, Deceleration, HomeSwitchMode,
PosLimitSwitch, NegLimitSwitch, Timeout, FastInputNumber,
CycleTime);
    
```

```

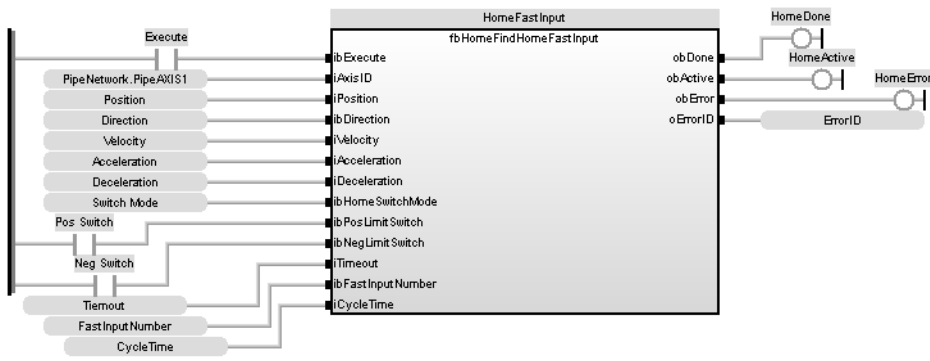
HomeComplete :=inst_fbHomeFindHomeFastInput.Done;
HomeActive :=inst_fbHomeFindHomeFastInput.Active;
HomeError :=inst_fbHomeFindHomeFastInput.Error;
HomeErrorID :=inst_fbHomeFindHomeFastInput.ErrorID;
    
```

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

### 5.2.6.93.9.2 Ladder Diagram



### 5.2.6.94.10.3 Function Block Diagram



**5.2.6.95 MLFB\_HomeFindHomeFastInputModulo** Pipe Network ✓

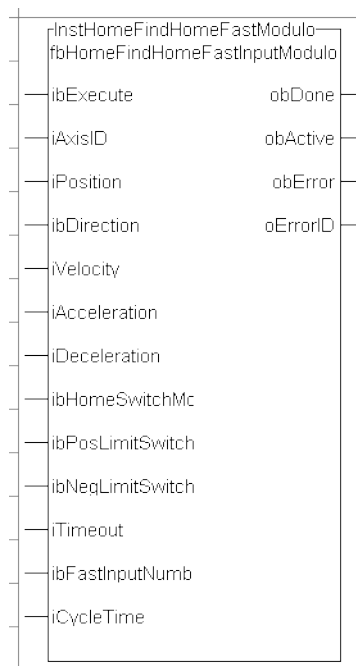
**5.2.6.96.1 Description**

This Application Specific Function function block performs a single-axis home to a limit switch connected to a High Speed Input. The motor starts to move according to the direction setting. The home position has been found as soon as the fast input selected is triggered on the edge selected.

An absolute move is made to the triggered position, and then the position value is set. The hardware limit switches are monitored during the homing procedure.

The drive behaves as follows in case that a hardware limit switch is active before the home-switch has been activated: The motor changes the direction until the home switch is crossed. This function is to be used when the axis is set-up in Modulo mode.

The following figure shows the function block I/O:



**Figure 1-143:** MLFB HomeFindHomeFastInputModulo

**5.2.6.97.2 Arguments**

**5.2.6.98.3.1 Input**

<b>ibExecute</b>	<b>Description</b>	Request the homing step procedure at rising edge
	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>iAxisID</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function
	<b>Data type</b>	AXIS_REF

	<b>Range</b>	[1 , 256]						
	<b>Unit</b>	n/a						
	<b>Default</b>	—						
<b>iPosition</b>	<b>Description</b>	Offset Position Applied After Home Switch is found						
	<b>Data type</b>	LREAL						
	<b>Range</b>	—						
	<b>Unit</b>	User unit						
	<b>Default</b>	—						
<b>ibDirection</b>	<b>Description</b>	Define the axis homing direction						
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>clockwise rotation</td> </tr> <tr> <td>1</td> <td>counterclockwise rotation</td> </tr> </tbody> </table>	Value	Description	0	clockwise rotation	1	counterclockwise rotation
Value	Description							
0	clockwise rotation							
1	counterclockwise rotation							
	<b>Data type</b>	BOOL						
	<b>Range</b>	[0 , 1]						
	<b>Unit</b>	n/a						
	<b>Default</b>	—						
<b>iVelocity</b>	<b>Description</b>	Commanded velocity for the homing move						
	<b>Data type</b>	LREAL						
	<b>Range</b>	—						
	<b>Unit</b>	User unit/sec						
	<b>Default</b>	—						
<b>iAcceleration</b>	<b>Description</b>	Commanded acceleration for the homing move						
	<b>Data type</b>	LREAL						
	<b>Range</b>	—						
	<b>Unit</b>	User unit/sec <sup>2</sup>						
	<b>Default</b>	—						
<b>iDeceleration</b>	<b>Description</b>	Commanded deceleration for the homing move						
	<b>Data type</b>	LREAL						
	<b>Range</b>	—						

	<b>Unit</b>	User unit/sec <sup>2</sup>						
	<b>Default</b>	—						
<b>ibLimitSwitchMode</b>	<b>Description</b>	Limit switch state to complete homing						
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Rising edge of switch</td> </tr> <tr> <td>1</td> <td>Falling edge of switch</td> </tr> </tbody> </table>	Value	Description	0	Rising edge of switch	1	Falling edge of switch
Value	Description							
0	Rising edge of switch							
1	Falling edge of switch							
	<b>Data type</b>	BOOL						
	<b>Range</b>	[0, 1]						
	<b>Unit</b>	n/a						
	<b>Default</b>	—						
<b>ibPosLimitSwitch</b>	<b>Description</b>	The positive direction limit switch input I/O point						
	<b>Data type</b>	BOOL						
	<b>Range</b>	[0, 1]						
	<b>Unit</b>	n/a						
	<b>Default</b>	—						
<b>ibNegLimitSwitch</b>	<b>Description</b>	The negative direction limit switch input I/O point						
	<b>Data type</b>	BOOL						
	<b>Range</b>	[0, 1]						
	<b>Unit</b>	n/a						
	<b>Default</b>	—						
<b>iTimeout</b>	<b>Description</b>	Maximum time for homing move to complete. If exceeded the homing procedure will error out. 0= no time limit						
	<b>Data type</b>	TIME						
	<b>Range</b>	—						
	<b>Unit</b>	sec						
	<b>Default</b>	—						
<b>ibFastInputNumber</b>	<b>Description</b>	Limit switch state to complete homing.						
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Fast Input Number 1</td> </tr> <tr> <td>1</td> <td>Fast Input Number 2</td> </tr> </tbody> </table>	Value	Description	0	Fast Input Number 1	1	Fast Input Number 2
Value	Description							
0	Fast Input Number 1							
1	Fast Input Number 2							
	<b>Data type</b>	BOOL						

	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>iCycleTime</b>	<b>Description</b>	Ethercat Cycle Time 250, 500 or 1000
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	microseconds
	<b>Default</b>	—

#### 5.2.6.99.4.2 Output

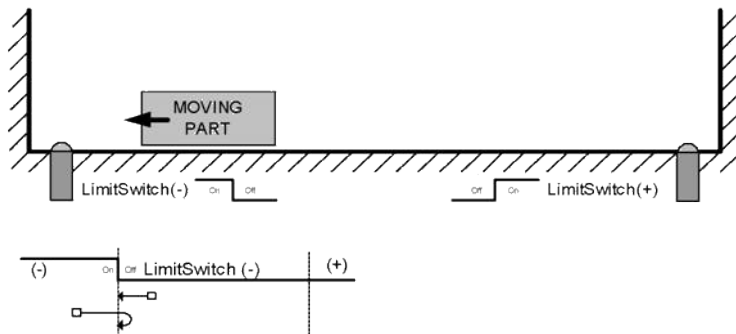
<b>obDone</b>	<b>Description</b>	Indicates the move completed successfully. The Command Position has reached the endpoint												
	<b>Data type</b>	BOOL												
	<b>Unit</b>	n/a												
<b>obActive</b>	<b>Description</b>	Indicates this move is the active move												
	<b>Data type</b>	BOOL												
	<b>Unit</b>	n/a												
<b>obError</b>	<b>Description</b>	Indicates an invalid input was specified or the move was terminated due to an error												
	<b>Data type</b>	BOOL												
	<b>Unit</b>	n/a												
<b>oErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE												
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Axis in Error State</td> </tr> <tr> <td>2</td> <td>Axis is Not Enabled</td> </tr> <tr> <td>3</td> <td>Timeout Exceeded</td> </tr> <tr> <td>4</td> <td>SDO Read/Write Error</td> </tr> <tr> <td>5</td> <td>Input Parameter out of Range</td> </tr> </tbody> </table>	Value	Description	1	Axis in Error State	2	Axis is Not Enabled	3	Timeout Exceeded	4	SDO Read/Write Error	5	Input Parameter out of Range
Value	Description													
1	Axis in Error State													
2	Axis is Not Enabled													
3	Timeout Exceeded													
4	SDO Read/Write Error													
5	Input Parameter out of Range													
	<b>Data type</b>	DINT												
	<b>Unit</b>	n/a												

#### 5.2.6.100.5 Usage



This homing procedure performs a homing function searching for sensor using only High Speed Input Switches. (A High Speed Limit Switch has 1 “Off” (or “On”) area).

- Home is commanded by user in the desired homing direction at the selected Velocity
- If LimitSwitch is found ‘On’ on rising ‘Execute’, then the process is started in the opposite direction as specified, LimitSwitch is search for ‘Off’ (or On, depending on LimitSwitchMode setting) Edge (released), and process is restarted again in original direction. This ensures that the end conditions are always the same
- The Timeout can cause an error if exceeded



#### 5.2.6.101.6 Related Functions

[MLFB\\_HomeFindHomeFastInput](#)

[MLFB\\_HomeFindLimitFastInput](#)

[MLFB\\_HomeFindLimitFastInputModulo](#)

#### 5.2.6.102.7 Example

#### 5.2.6.103.8.1 Structured Text

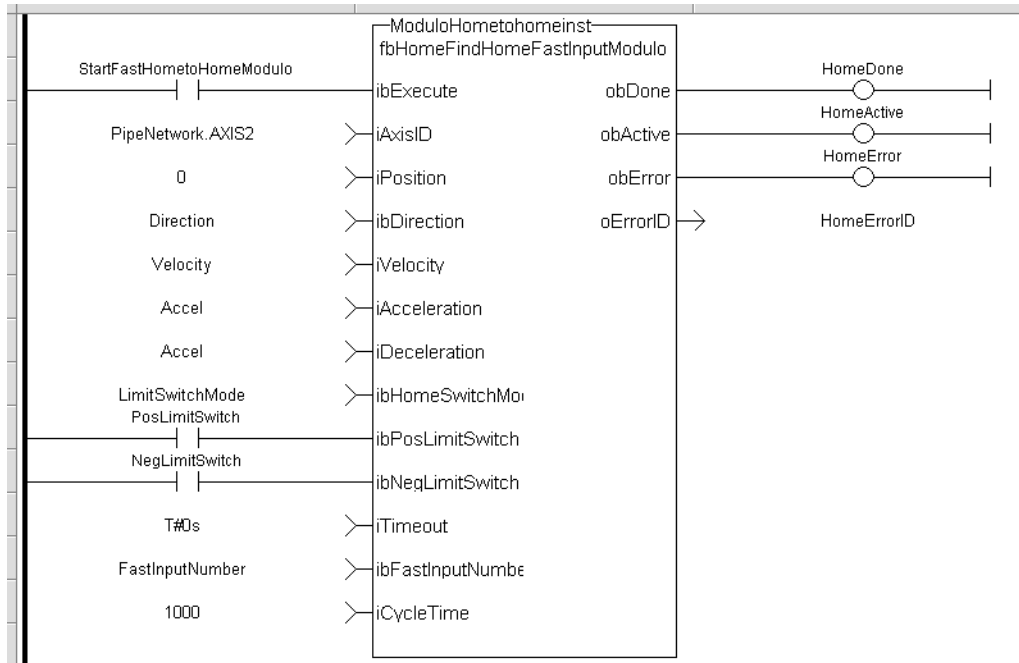
```
Direction:= 0;
Position:=1000;
Velocity:=1000;
Acceleration:=10000;
Deceleration:=10000;
SwitchMode:=0;
Timeout:=T#100;
FastInputNumber:=0;
CycleTime:=1000;

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

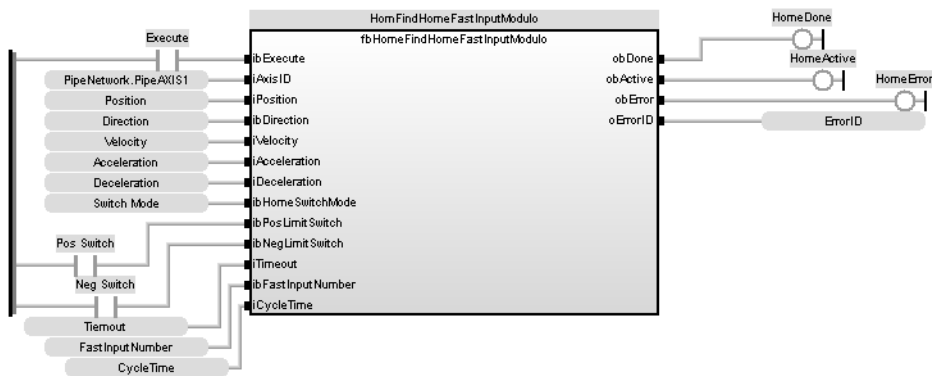
HomeComplete :=inst_fbHomeFindHomeFastInputModulo.Done;
HomeActive :=inst_fbHomeFindHomeFastInputModulo.Active;
HomeError :=inst_fbHomeFindHomeFastInputModulo.Error;
HomeErrorID :=inst_fbHomeFindHomeFastInputModulo.ErrorID;

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

#### 5.2.6.104.9.2 Ladder Diagram



### 5.2.6.105.10.3 Function Block Diagram



5.2.6.106

MLFB\_HomeFindLimitFastInput 

5.2.6.107.1 Description

This function block performs a single-axis home to a limit switch connected to a High Speed Input. The motor starts to move according to the direction setting. The home position has been found as soon as the fast input selected is triggered on the edge selected.

An absolute move is made to the triggered position, and then the position value is set.

The following figure shows the function block I/O:

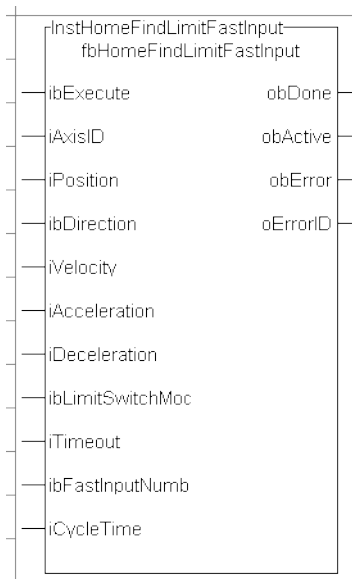


Figure 1-144: MLFB HomeFindLimitFastInput

5.2.6.108.2 Arguments

5.2.6.109.3.1 Input

<b>ibExecute</b>	<b>Description</b>	Request the homing step procedure at rising edge
	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>iAxisID</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function
	<b>Data type</b>	AXIS_REF
	<b>Range</b>	[1 , 256]
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>iPosition</b>	<b>Description</b>	Offset Position Applied After Home Switch is found						
	<b>Data type</b>	LREAL						
	<b>Range</b>	—						
	<b>Unit</b>	User unit						
	<b>Default</b>	—						
<b>ibDirection</b>	<b>Description</b>	Define the axis homing direction						
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>clockwise rotation</td> </tr> <tr> <td>1</td> <td>counterclockwise rotation</td> </tr> </tbody> </table>	Value	Description	0	clockwise rotation	1	counterclockwise rotation
	Value	Description						
	0	clockwise rotation						
	1	counterclockwise rotation						
<b>Data type</b>	BOOL							
<b>Range</b>	[0, 1]							
<b>Unit</b>	n/a							
<b>Default</b>	—							
<b>iVelocity</b>	<b>Description</b>	Commanded velocity for the homing move						
	<b>Data type</b>	LREAL						
	<b>Range</b>	—						
	<b>Unit</b>	User unit/sec						
	<b>Default</b>	—						
<b>iAcceleration</b>	<b>Description</b>	Commanded acceleration for the homing move						
	<b>Data type</b>	LREAL						
	<b>Range</b>	—						
	<b>Unit</b>	User unit/sec <sup>2</sup>						
	<b>Default</b>	—						
<b>ibLimitSwitchMode</b>	<b>Description</b>	Limit switch state to complete homing						
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Rising edge of switch</td> </tr> <tr> <td>1</td> <td>Falling edge of switch</td> </tr> </tbody> </table>	Value	Description	0	Rising edge of switch	1	Falling edge of switch
	Value	Description						
	0	Rising edge of switch						
	1	Falling edge of switch						
<b>Data type</b>	BOOL							
<b>Range</b>	[0, 1]							
<b>Unit</b>	n/a							

	<b>Default</b>	—						
<b>ibFastInputNumber</b>	<b>Description</b>	Limit switch state to complete homing.						
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Fast Input Number 1</td> </tr> <tr> <td>1</td> <td>Fast Input Number 2</td> </tr> </tbody> </table>	Value	Description	0	Fast Input Number 1	1	Fast Input Number 2
	Value	Description						
	0	Fast Input Number 1						
1	Fast Input Number 2							
<b>Data type</b>	BOOL							
<b>Range</b>	[0, 1]							
	<b>Unit</b>	n/a						
	<b>Default</b>	—						
<b>iCycleTime</b>	<b>Description</b>	Ethercat Cycle Time 250, 500 or 1000						
	<b>Data type</b>	LREAL						
	<b>Range</b>	—						
	<b>Unit</b>	microseconds						
	<b>Default</b>	—						

#### 5.2.6.110.4.2 Output

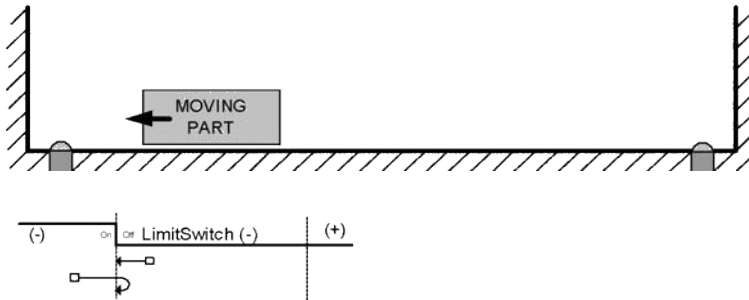
<b>obDone</b>	<b>Description</b>	Indicates the move completed successfully. The Command Position has reached the endpoint
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>obActive</b>	<b>Description</b>	Indicates this move is the active move
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>obError</b>	<b>Description</b>	Indicates an invalid input was specified or the move was terminated due to an error
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

<b>oErrorID</b>	<b>Description</b>												
	Indicates the error if Error output is set to TRUE												
	<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Axis in Error State</td> </tr> <tr> <td>2</td> <td>Axis is Not Enabled</td> </tr> <tr> <td>3</td> <td>Timeout Exceeded</td> </tr> <tr> <td>4</td> <td>SDO Read/Write Error</td> </tr> <tr> <td>5</td> <td>Input Parameter out of Range</td> </tr> </tbody> </table>	Value	Description	1	Axis in Error State	2	Axis is Not Enabled	3	Timeout Exceeded	4	SDO Read/Write Error	5	Input Parameter out of Range
Value	Description												
1	Axis in Error State												
2	Axis is Not Enabled												
3	Timeout Exceeded												
4	SDO Read/Write Error												
5	Input Parameter out of Range												
	<b>Data type</b> DINT												
	<b>Unit</b> n/a												

### 5.2.6.111.5 Usage

This homing procedure performs a homing function searching for sensor using only High Speed Input Switches. (A High Speed Limit Switch has 1 "Off" (or "On") area).

- Home is commanded by user in the desired homing direction at the selected Velocity.
- The Timeout can cause an error if exceeded



### 5.2.6.112.6 Related Functions

[MLFB\\_HomeFindHomeFastInput](#)

[MLFB\\_HomeFindHomeFastInputModulo](#)

[MLFB\\_HomeFindLimitFastInputModulo](#)

### 5.2.6.113.7 Example

#### 5.2.6.114.8.1 Structured Text

```

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

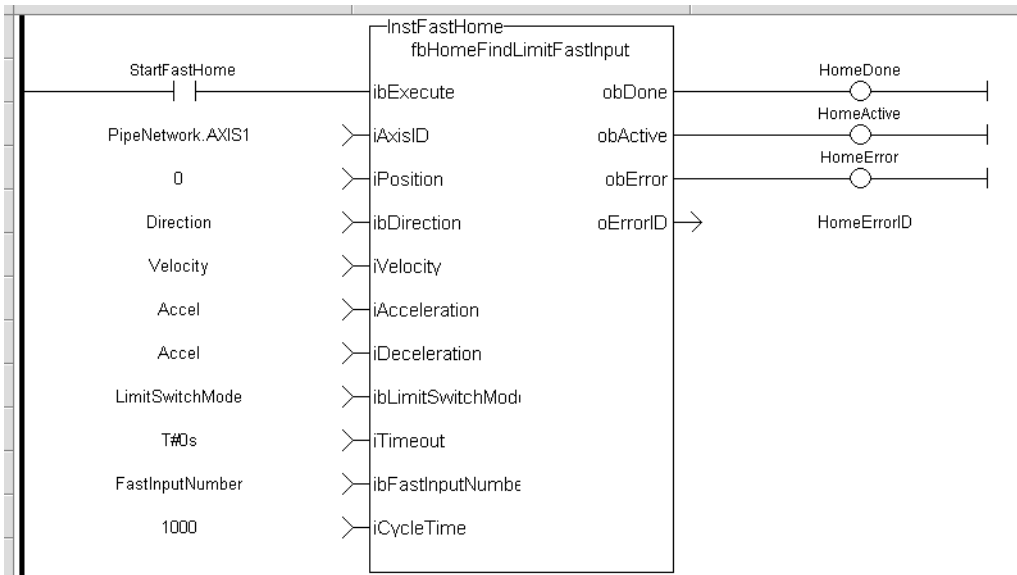
```

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

```

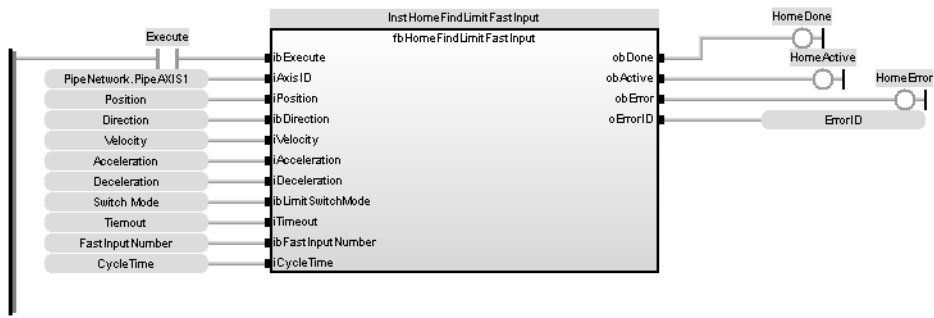
HomeComplete :=inst_fbHomeFindLimitFastInput.Done;
HomeActive :=inst_fbHomeFindLimitFastInput.Active;
HomeError :=inst_fbHomeFindLimitFastInput.Error;
HomeErrorID :=inst_fbHomeFindLimitFastInput.ErrorID;
    
```

### 5.2.6.115.9.2 Ladder Diagram



### 5.2.6.116.10.3 Function Block Diagram





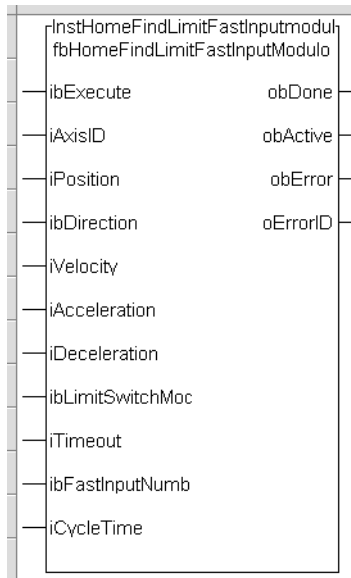
**5.2.6.117 MLFB\_HomeFindLimitFastInputModulo** Pipe Network ✓

**5.2.6.118.1 Description**

This function block performs a single-axis home to a limit switch connected to a High Speed Input. The motor starts to move according to the direction setting. The home position has been found as soon as the fast input selected is triggered on the edge selected.

An absolute move is made to the triggered position, and then the position value is set. This function is to be used when the axis is set-up in Modulo mode.

The following figure shows the function block I/O:



**Figure 1-145:** MLFB HomeFindLimitFastInputModulo

**5.2.6.119.2 Arguments**

**5.2.6.120.3.1 Input**

ibExecute	Description Data type Range Unit Default	Request the homing step procedure at rising edge BOOL [0 , 1] n/a —
iAxisID	Description Data type Range Unit Default	Name of a declared instance of the AXIS_REF library function AXIS_REF [1 , 256] n/a —
iPosition	Description Data type Range Unit Default	Offset Position Applied After Home Switch is found LREAL — User unit —

Define the axis homing direction

ibDirection	Description	<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>clockwise rotation</td> </tr> <tr> <td>1</td> <td>counterclockwise rotation</td> </tr> </tbody> </table>	Value	Description	0	clockwise rotation	1	counterclockwise rotation
	Value	Description						
	0	clockwise rotation						
	1	counterclockwise rotation						
	Data type	BOOL						
Range	[0 , 1]							
Unit	n/a							
Default	—							
iVelocity	Description	Commanded velocity for the homing move						
	Data type	LREAL						
	Range	—						
	Unit	User unit/sec						
	Default	—						
iAcceleration	Description	Commanded acceleration for the homing move						
	Data type	LREAL						
	Range	—						
	Unit	User unit/sec <sup>2</sup>						
	Default	—						
iDeceleration	Description	Commanded deceleration for the homing move						
	Data type	LREAL						
	Range	—						
	Unit	User unit/sec <sup>2</sup>						
	Default	—						
Limit switch state to complete homing								
ibLimitSwitchMode	Description	<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Rising edge of switch</td> </tr> <tr> <td>1</td> <td>Falling edge of switch</td> </tr> </tbody> </table>	Value	Description	0	Rising edge of switch	1	Falling edge of switch
	Value	Description						
	0	Rising edge of switch						
	1	Falling edge of switch						
	Data type	BOOL						
Range	[0 , 1]							
Unit	n/a							
Default	—							
iTimeout	Description	Maximum time for homing move to complete. If exceeded the homing procedure will error out. 0= no time limit						
	Data type	TIME						
	Range	—						
	Unit	sec						
	Default	—						
Limit switch state to complete homing.								
ibFastInputNumber	Description	<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Fast Input Number 1</td> </tr> <tr> <td>1</td> <td>Fast Input Number 2</td> </tr> </tbody> </table>	Value	Description	0	Fast Input Number 1	1	Fast Input Number 2
	Value	Description						
	0	Fast Input Number 1						
	1	Fast Input Number 2						
	Data type	BOOL						
Range	[0 , 1]							
Unit	n/a							
Default	—							

iCycleTime	Description	Ethercat Cycle Time 250, 500 or 1000
	Data type	LREAL
	Range	—
	Unit	microseconds
	Default	—

**5.2.6.121.4.2 Output**

obDone	Description	Indicates the move completed successfully. The Command Position has reached the endpoint
	Data type	BOOL
	Unit	n/a
obActive	Description	Indicates this move is the active move
	Data type	BOOL
	Unit	n/a
obError	Description	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
	Unit	n/a

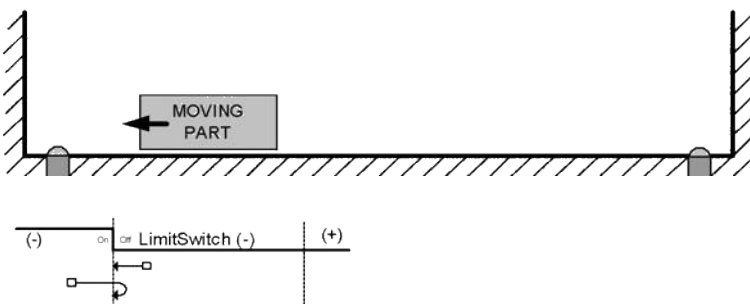
Indicates the error if Error output is set to TRUE

oErrorID	Description	Value	Description
		1	Axis in Error State
		2	Axis is Not Enabled
		3	Timeout Exceeded
		4	SDO Read/Write Error
	5	Input Parameter out of Range	
	Data type	DINT	
	Unit	n/a	

**5.2.6.122.5 Usage**

This homing procedure performs a homing function searching for sensor using only High Speed Input Switches. (A High Speed Limit Switch has 1 “Off” (or “On”) area).

- Home is commanded by user in the desired homing direction at the selected Velocity.
- The Timeout can cause an error if exceeded



**5.2.6.123.6 Related Functions**

- [MLFB\\_HomeFindHomeFastInput](#)
- [MLFB\\_HomeFindHomeFastInputModulo](#)
- [MLFB\\_HomeFindLimitFastInput](#)

**5.2.6.124.7 Example****5.2.6.125.8.1 Structured Text**

```

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

```

```

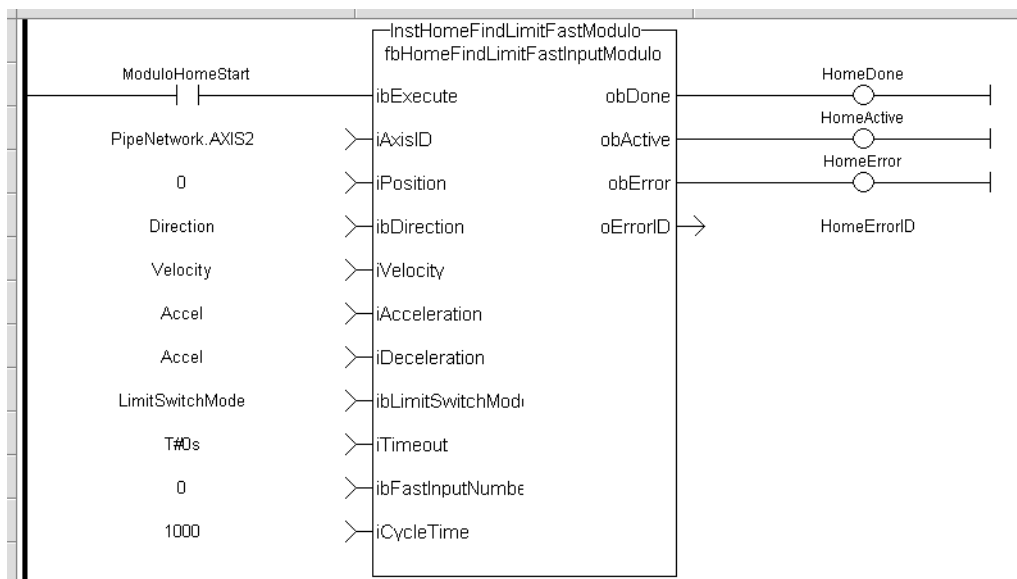
inst_fbHomeFindLimitFastInputModulo(True, Axis1, Position,
Direction, Velocity, Acceleration, Deceleration,
LimitSwitchMode, Timeout, FastInputNumber, CycleTime);

```

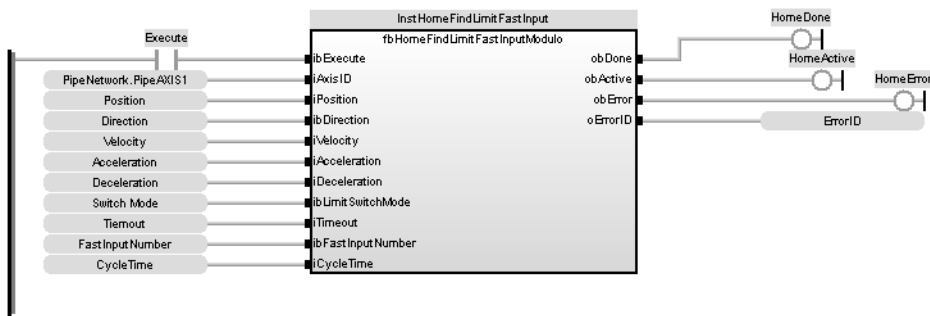
```

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

```

**5.2.6.126.9.2 Ladder Diagram**

### 5.2.6.127.10.3 Function Block Diagram



## 5.2.7 Jog for Pipe Network

### 5.2.7.1 Description

This function is defined to jog an axis in the selected direction at a defined speed. The En input (FFLD editor only) must be high. Typically wired to the rail. The AxisID selects the axis to jog. The JogPlus and JogMinus inputs select the direction the motion will occur in. Only one of these inputs should be enabled at a given time. If both are selected the motion will stop. If other motion is active when the jog is requested that motion will be aborted and the jog will start.

The following figure shows the function I/O

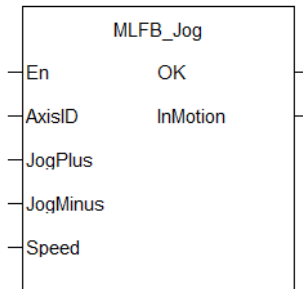


Figure 1-146: Kollmorgen UDFB Jog for PipeNetwork

### 5.2.7.2 Arguments

#### 5.2.7.3.1 Input

<b>En</b>	<b>Description</b>	Enables execution (FFLD only )
	<b>Data type</b>	BOOL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxisID</b>	<b>Description</b>	ID Name of the Axis
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>JogPlus</b>	<b>Description</b>	Enables a Jog in the plus direction
	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>JogMinus</b>	<b>Description</b>	Enables a Jog in the Minus direction

	<b>Data type</b>	BOOL
	<b>Range</b>	[0, 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Speed</b>	<b>Description</b>	Rate at which the axis will move
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	—

#### 5.2.7.4.2 Output

<b>InMotion</b>	<b>Description</b>	Jogging is active when TRUE
	<b>Data type</b>	BOOL
	<b>Range</b>	n/a

#### 5.2.7.5 Usage

This function is used to command motion in a designated direction at a defined rate. This may be used where continuous motion required as in a conveyor system, or in a setup mode for manually jogging the axis. Motion will start when the JogPlus or JogMinus input is true. It will stop when the input goes false. This function is used with the Pipe Network motion engine.

#### 5.2.7.6 Related Functions

[MLAxisMoveVel](#)

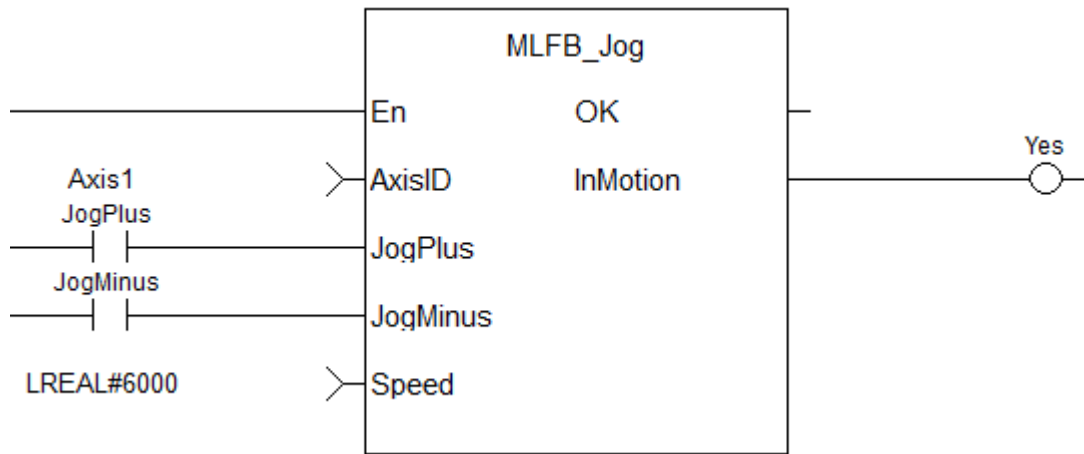
#### 5.2.7.7 Example

##### 5.2.7.8.1 Structured Text

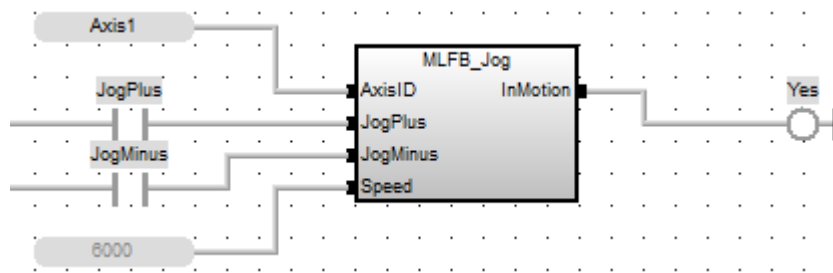
```
//Jog Axis1 at 6000 user units a second when JogPlus or JogMinus
variables are TRUE
//Stop motion on falling edge of either variable
MLFB_Jog( PipeNetwork.AXIS1, JogPlus, JogMinus, 6000 );
```

#### 5.2.7.9.2 Ladder Diagram





**5.2.7.10.3 Function Block Diagram**



### 5.2.7.11 MLFB\_PlsPosFw Pipe Network ✓

#### 5.2.7.12.1 Description

This function block should be used in the command position path with ascending position. A dedicated comparator pipe block is needed. The Boolean output oPLS is set to TRUE if the position of the comparator has crossed the start position and is set to FALSE if the position has crossed the end position. The function block is executed cyclically. The modulo position is considered. The function block has the possibility to compensate a delay time of the connected device, e.g. glue nozzles.

#### 5.2.7.13.2 Arguments

##### 5.2.7.14.3.1 Input

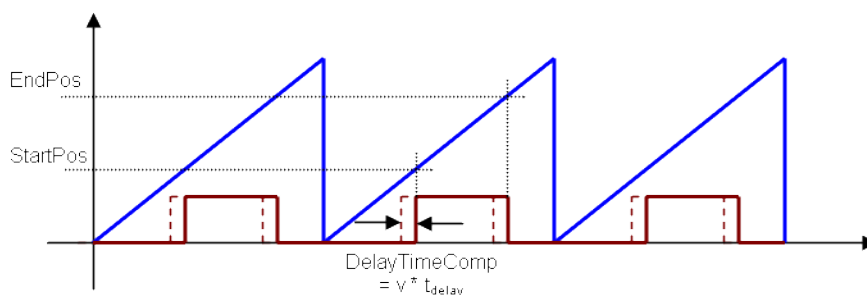
ibExecute	Description Data type	Enable PLS BOOL
iDedicatedCmplID	Description Data type	ID of dedicated comparator DINT
iStartPos	Description Data type	Start position of PLS LREAL
iEndPos	Description Data type	End position of PLS LREAL
iDelayTime	Description Data type	Delay time for compensation TIME
ibForce	Description Data type	Force PLS BOOL

##### 5.2.7.15.4.2 Output

oPLS	Description Data type	Position limit switch BOOL
------	--------------------------	-------------------------------

#### 5.2.7.16.5 Example

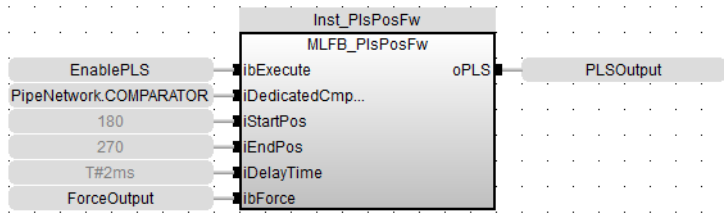
#### 5.2.7.17.6 Timing



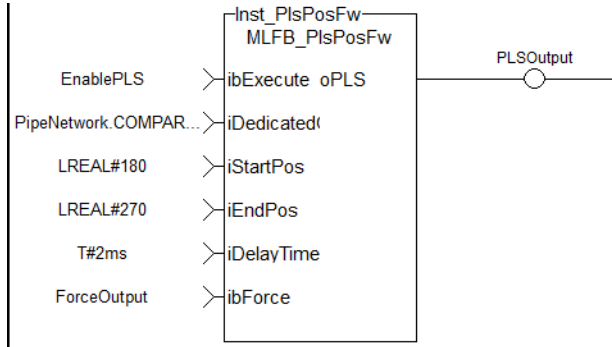
#### 5.2.7.18.7.1 ST

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

### 5.2.7.19.8.2 FBD



### 5.2.7.20.9.3 FFLD



**5.2.7.21 MLFB\_PlsPosFwBw** Pipe Network ✓

**5.2.7.22.1 Description**

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

**5.2.7.23.2 Arguments**

**5.2.7.24.3.1 Input**

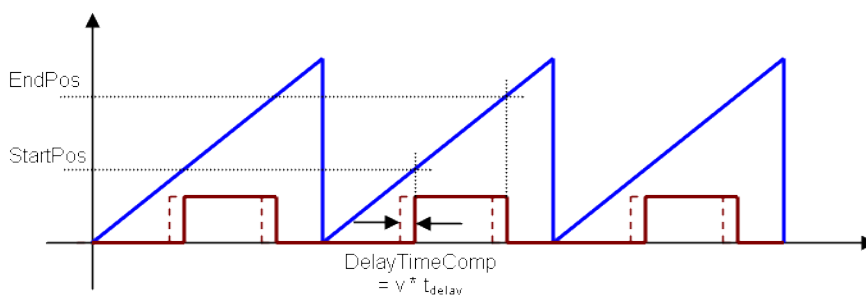
ibExecute	Description Data type	Enable PLS BOOL
iAnyModuloBkID	Description Data type	Any modulo pipe network block ID DINT
iStartPos	Description Data type	Start position of PLS LREAL
iEndPos	Description Data type	End position of PLS LREAL
iDelayTime	Description Data type	Delay time for compensation TIME
iHysteresis	Description Data type	Hysteresis LREAL
ibForce	Description Data type	Force PLS BOOL

**5.2.7.25.4.2 Output**

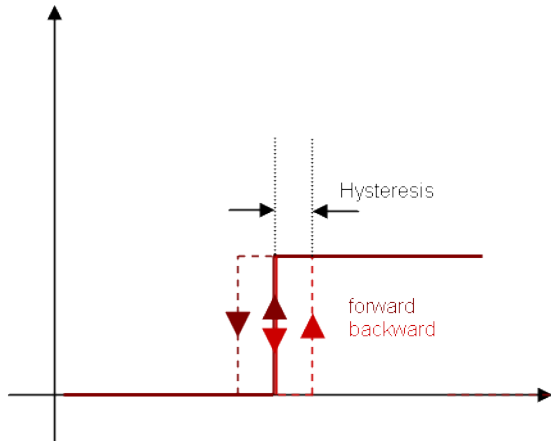
oPLS	Description Data type	Position limit switch BOOL
------	--------------------------	-------------------------------

**5.2.7.26.5 Example**

**5.2.7.27.6 Timing**



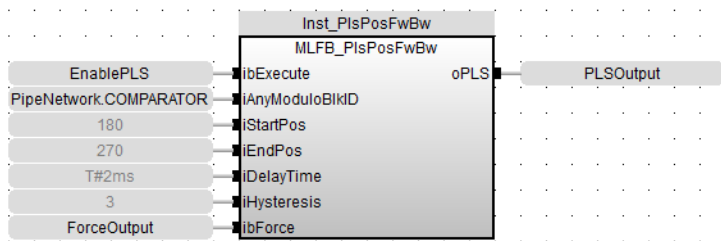
**5.2.7.28.7 Hysteresis**



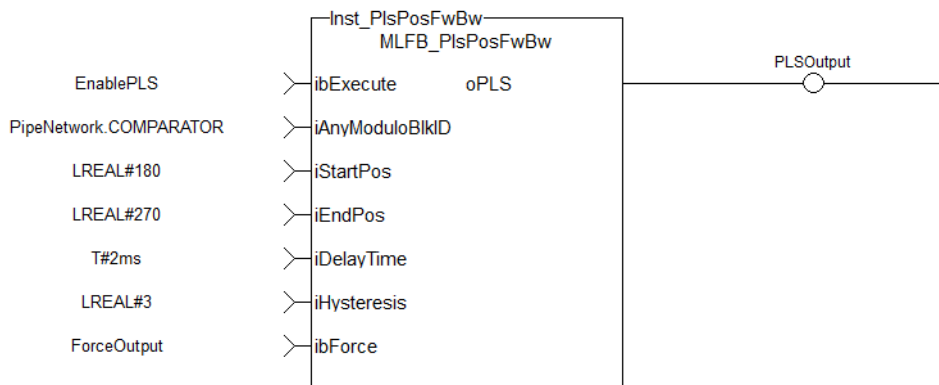
**5.2.7.29.8.1 ST**

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

**5.2.7.30.9.2 FBD**



**5.2.7.31.10.3 FFLD**



**5.2.7.32 MLFB\_PlsTimeFw** Pipe Network ✓

**5.2.7.33.1 Description**

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

**5.2.7.34.2 Arguments**

**5.2.7.35.3.1 Input**

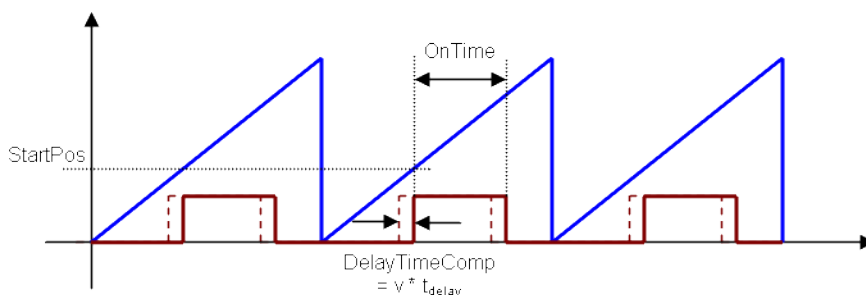
<b>ibExecute</b>	<b>Description</b>	Enable PLS
	<b>Data type</b>	BOOL
<b>iDedicatedCmplID</b>	<b>Description</b>	ID of dedicated comparator
	<b>Data type</b>	DINT
<b>iStartPos</b>	<b>Description</b>	Start position of PLS
	<b>Data type</b>	LREAL
<b>iOnTime</b>	<b>Description</b>	Time PLS is on
	<b>Data type</b>	TIME
<b>iDelayTime</b>	<b>Description</b>	Delay time for compensation
	<b>Data type</b>	TIME
<b>ibForce</b>	<b>Description</b>	Force PLS
	<b>Data type</b>	BOOL

**5.2.7.36.4.2 Output**

<b>oPLS</b>	<b>Description</b>	Position limit switch
	<b>Data type</b>	BOOL

**5.2.7.37.5 Example**

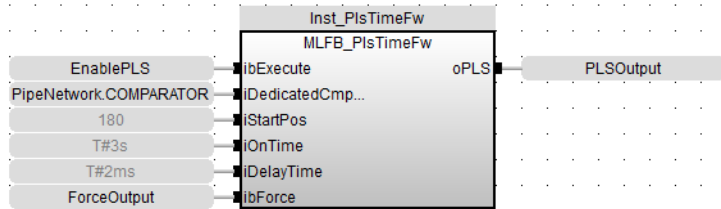
**5.2.7.38.6 Timing**



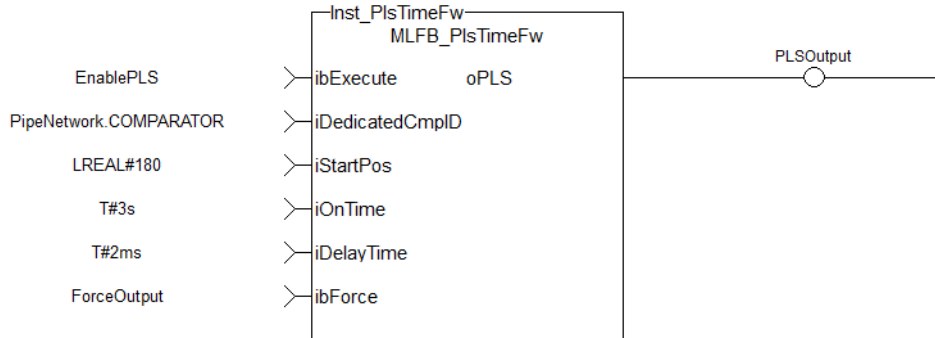
**5.2.7.39.7.1 ST**

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

### 5.2.7.40.8.2 FBD



### 5.2.7.41.9.3 FFLD



**5.2.7.42 MCFB\_StepAbsolutes** PLCopen ✓

**5.2.7.43.1 Description**

This function block performs a static homing function by setting Actual Position to the position of an absolute encoder. No physical motion is performed in this mode. Equivalent to MC\_SetPosition is performed with SetPosition coming from absolute encoder reading, but with the option of using the once per rev feedback value.

The following figure shows the function block I/O:

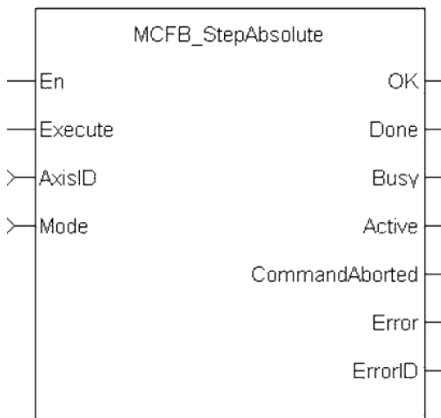


Figure 1-147: MCFB StepAbsolute

**5.2.7.44.2 Arguments**

**5.2.7.45.3.1 Input**

<b>En</b>	<b>Description</b>	Enables execution (FFLD only )
	<b>Data type</b>	BOOL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Execute</b>	<b>Description</b>	Request the homing step procedure at rising edge
	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxisID</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function
	<b>Data type</b>	<a href="#">AXIS_REF</a>
	<b>Range</b>	[1 , 256]
	<b>Unit</b>	n/a



	<b>Default</b>	—						
<b>Mode</b>	<b>Description</b>	Define the actual position assignment source						
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>use drive feedback position for actual position</td> </tr> <tr> <td>1</td> <td>use once per rev feedback position</td> </tr> </tbody> </table>	Value	Description	0	use drive feedback position for actual position	1	use once per rev feedback position
	Value	Description						
	0	use drive feedback position for actual position						
1	use once per rev feedback position							
<b>Data type</b>	BOOL							
<b>Range</b>	[0 , 1]							
	<b>Unit</b>	n/a						
	<b>Default</b>	—						

#### 5.2.7.46.4.2 Output

<b>Done</b>	<b>Description</b>	Indicates the move completed successfully. The Command Position has reached the endpoint
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Busy</b>	<b>Description</b>	High from the moment the Execute input is one-shot to the time the move is ended
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Active</b>	<b>Description</b>	Indicates this move is the active move
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>CommandAborted</b>	<b>Description</b>	Indicates the move was aborted
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified or the move was terminated due to an error
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified or the move was terminated due to an error
	<b>Data type</b>	BOOL

	<b>Unit</b>	n/a		
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE		
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Desired SetPosition is outside of Rollover period</td> </tr> </tbody> </table>	Value	Description
Value	Description			
1	Desired SetPosition is outside of Rollover period			
	<b>Data type</b>	INT		
	<b>Unit</b>	n/a		

**5.2.7.47.5 Related Functions**

[MCFB\\_StepAbsSwitch](#)

[MCFB\\_StepRefPulse](#)

[MCFB\\_StepBlock](#)

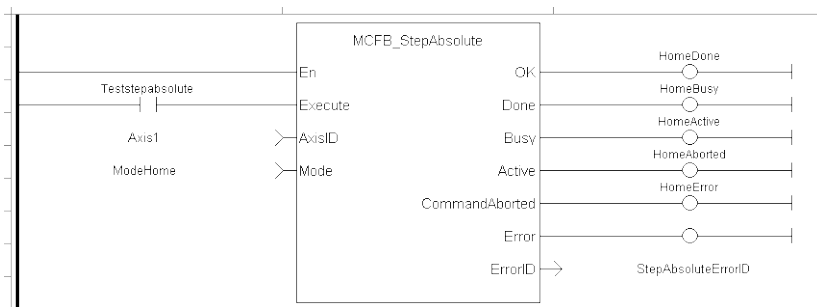
[MCFB\\_StepLimitSwitch](#)

**5.2.7.48.6 Example**

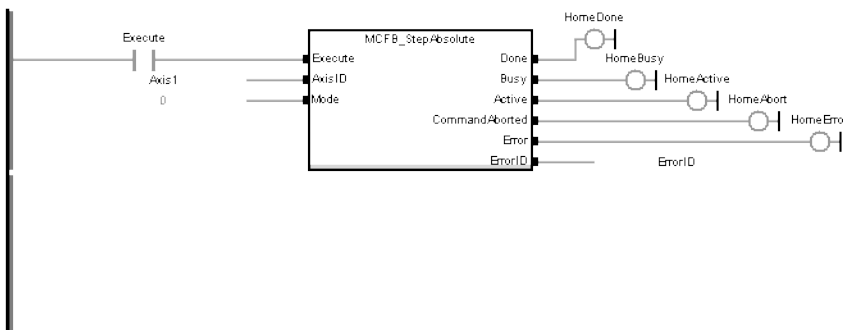
**5.2.7.49.7.1 Structured Text**

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

**5.2.7.50.8.2 Ladder Diagram**



**5.2.7.51.9.3 Function Block Diagram**

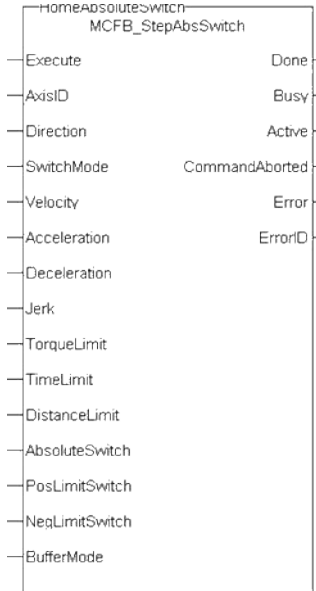


**5.2.7.52 MCFB\_StepAbsSwitch** PLCopen

**5.2.7.53.1 Description**

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

The following figure shows the function block I/O:



**Figure 1-148:** MCFB StepAbsSwitch

**5.2.7.54.2 Arguments**

**5.2.7.55.3.1 Input**

<b>Execute</b>	<b>Description</b>	Request the homing step procedure at rising edge. Outputs are reset when execute input is false.
	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxisID</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function
	<b>Data type</b>	<a href="#">AXIS_REF</a>
	<b>Range</b>	[1 , 256]
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>Direction</b>	<b>Description</b>	Define the axis homing direction										
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>clockwise rotation</td> </tr> <tr> <td>1</td> <td>counterclockwise rotation</td> </tr> <tr> <td>2</td> <td>clockwise if AbsoluteSwitch starts Off and negative if switch starts On</td> </tr> <tr> <td>3</td> <td>counter clockwise if AbsoluteSwitch starts On and positive if switch starts Off</td> </tr> </tbody> </table>	Value	Description	0	clockwise rotation	1	counterclockwise rotation	2	clockwise if AbsoluteSwitch starts Off and negative if switch starts On	3	counter clockwise if AbsoluteSwitch starts On and positive if switch starts Off
	Value	Description										
	0	clockwise rotation										
	1	counterclockwise rotation										
2	clockwise if AbsoluteSwitch starts Off and negative if switch starts On											
3	counter clockwise if AbsoluteSwitch starts On and positive if switch starts Off											
<b>Data type</b>	DINT											
<b>Range</b>	[0 , 3]											
<b>Unit</b>	n/a											
	<b>Default</b>	—										
<b>SwitchMode</b>	<b>Description</b>	Switch state to complete homing										
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>switch is on</td> </tr> <tr> <td>1</td> <td>switch if off</td> </tr> <tr> <td>2</td> <td>rising edge of switch</td> </tr> <tr> <td>3</td> <td>falling edge of switch</td> </tr> </tbody> </table>	Value	Description	0	switch is on	1	switch if off	2	rising edge of switch	3	falling edge of switch
	Value	Description										
	0	switch is on										
	1	switch if off										
2	rising edge of switch											
3	falling edge of switch											
<b>Data type</b>	DINT											
<b>Range</b>	[0 , 3]											
<b>Unit</b>	n/a											
	<b>Default</b>	—										
<b>Velocity</b>	<b>Description</b>	Commanded velocity for the homing move										
	<b>Data type</b>	LREAL										
	<b>Range</b>	—										
	<b>Unit</b>	User unit/sec										
	<b>Default</b>	—										
<b>Acceleration</b>	<b>Description</b>	Commanded acceleration for the homing move										
	<b>Data type</b>	LREAL										
	<b>Range</b>	—										
	<b>Unit</b>	User unit/sec <sup>2</sup>										
	<b>Default</b>	—										

<b>Deceleration</b>	<b>Description</b>	Commanded deceleration for the homing move
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Commanded jerk for the homing move (if zero, then trapezoidal acc/dec is used)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—
<b>TorqueLimit</b>	<b>Description</b>	Maximum torque applied for the homing move entered in thousandths of maximum torque, e.g. "250" is 250/1000, or 25%.
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>TimeLimit</b>	<b>Description</b>	Maximum time for homing move to complete. If exceeded the homing procedure will error out. 0= no time limit
	<b>Data type</b>	TIME
	<b>Range</b>	—
	<b>Unit</b>	sec
	<b>Default</b>	—
<b>DistanceLimit</b>	<b>Description</b>	Maximum distance for homing move to complete. If exceeded the homing procedure will error out. 0= no distance limit
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>AbsoluteSwitch</b>	<b>Description</b>	The absolute switch input I/O point

	<b>Data type</b>	BOOL														
	<b>Range</b>	[0 , 1]														
	<b>Unit</b>	n/a														
	<b>Default</b>	—														
<b>PosLimitSwitch</b>	<b>Description</b>	The positive direction limit switch input I/O point														
	<b>Data type</b>	BOOL														
	<b>Range</b>	[0 , 1]														
	<b>Unit</b>	n/a														
	<b>Default</b>	—														
<b>NegLimitSwitch</b>	<b>Description</b>	The negative direction limit switch input I/O point														
	<b>Data type</b>	BOOL														
	<b>Range</b>	[0 , 1]														
	<b>Unit</b>	n/a														
	<b>Default</b>	—														
<b>SwitchMode</b>	<b>Description</b>	Switch state to complete homing														
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>abort</td> </tr> <tr> <td>1</td> <td>buffer</td> </tr> <tr> <td>2</td> <td>Blend to active</td> </tr> <tr> <td>3</td> <td>blend to next</td> </tr> <tr> <td>4</td> <td>blend to low velocity</td> </tr> <tr> <td>5</td> <td>blend to high velocity</td> </tr> </tbody> </table>	Value	Description	0	abort	1	buffer	2	Blend to active	3	blend to next	4	blend to low velocity	5	blend to high velocity
Value	Description															
0	abort															
1	buffer															
2	Blend to active															
3	blend to next															
4	blend to low velocity															
5	blend to high velocity															
	<b>Data type</b>	SINT														
	<b>Range</b>	[0 , 5]														
	<b>Unit</b>	n/a														
	<b>Default</b>	—														

#### 5.2.7.56.4.2 Output

<b>Done</b>	<b>Description</b>	Indicates the move completed successfully. The Command Position has reached the endpoint
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

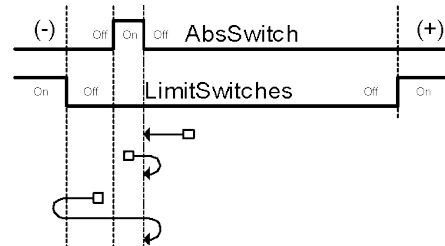
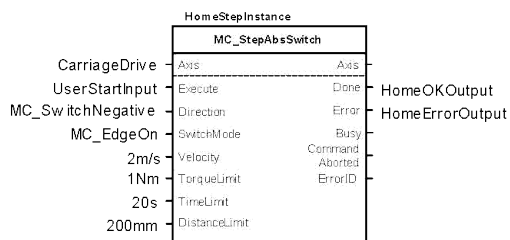
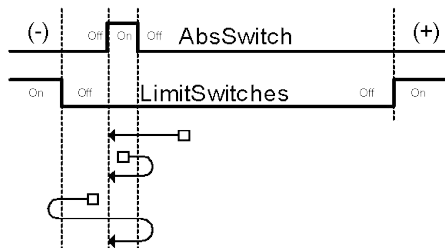
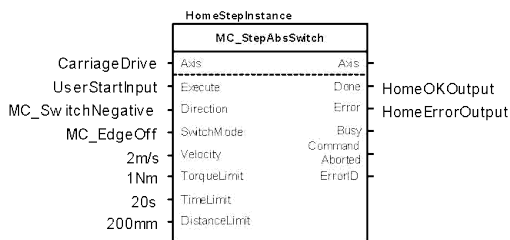
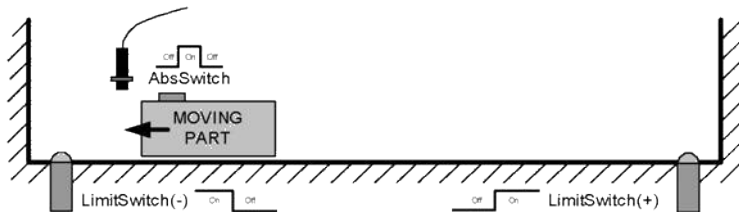
<b>Busy</b>	<b>Description</b>	High from the moment the Execute input is one-shot to the time the move is ended														
	<b>Data type</b>	BOOL														
	<b>Unit</b>	n/a														
<b>Active</b>	<b>Description</b>	Indicates this move is the active move														
	<b>Data type</b>	BOOL														
	<b>Unit</b>	n/a														
<b>CommandAborted</b>	<b>Description</b>	Indicates the move was aborted														
	<b>Data type</b>	BOOL														
	<b>Unit</b>	n/a														
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified or the move was terminated due to an error														
	<b>Data type</b>	BOOL														
	<b>Unit</b>	n/a														
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE														
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>TimeLimit exceeded</td> </tr> <tr> <td>2</td> <td>DistanceLimit exceeded</td> </tr> <tr> <td>3</td> <td>TorqueLimit exceeded</td> </tr> <tr> <td>4</td> <td>axis error stop state</td> </tr> <tr> <td>5</td> <td>axis not enabled</td> </tr> <tr> <td>6</td> <td>invalid inputs for Velocity-Acceleration-Deceleration</td> </tr> </tbody> </table>	Value	Description	1	TimeLimit exceeded	2	DistanceLimit exceeded	3	TorqueLimit exceeded	4	axis error stop state	5	axis not enabled	6	invalid inputs for Velocity-Acceleration-Deceleration
Value	Description															
1	TimeLimit exceeded															
2	DistanceLimit exceeded															
3	TorqueLimit exceeded															
4	axis error stop state															
5	axis not enabled															
6	invalid inputs for Velocity-Acceleration-Deceleration															
	<b>Data type</b>	INT														
	<b>Unit</b>	n/a														

### 5.2.7.57.5 Usage

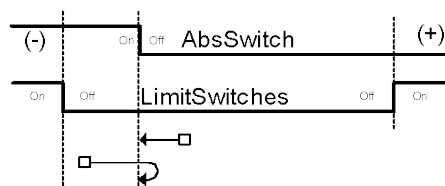
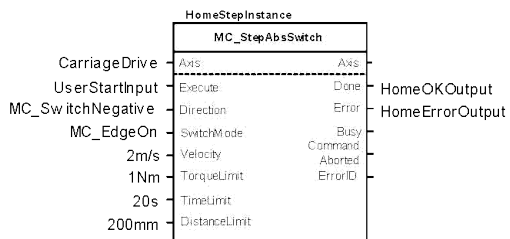
This physical layout has the risk that homing is started in the wrong direction (escaping the switch). To support such case, it implements a special behavior when Limit Switches are found (or the AbsSwitch itself is "On" at Execute):

- The homing is commanded in the most likely direction were the sensor can be found. In this example (-).
- The velocity is defined by the input.
- The torque is limited.
- Both Time and Distance Limits can cause an error if exceeded
- If any LimitSwitch is found during Homing (any of them), then a special process is started in the opposite direction, the AbsSwitch is searched to switch off (or On, depending on SwitchMode setting). The Edge (passed by), and homing process is restarted in the original direction and with the same conditions. This ensures that the end conditions are always same

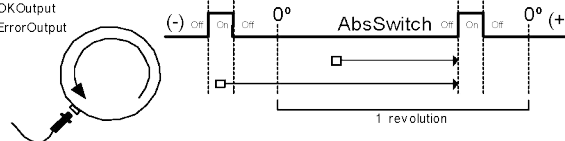
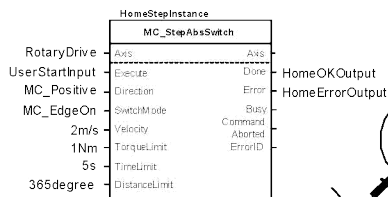
- If the SwitchMode is either MC\_SwitchNegative or MC\_SwitchPositive, then the special process is also started in opposite direction depending from the switch state at 'execute'.
- The direction changes only when the specified Velocity is reached (InVelocity).
- This Function Block doesn't modify the actual position



An overlapping switch configuration is also possible. This has same the behavior as working on the limit switches:



If the input Direction is set to a fixed direction (MC\_Positive or MC\_Negative), then the initial switch state is ignored (used for example in rotary axis where only one sense of rotation is allowed):



### 5.2.7.58.6 Related Functions

[MCFB\\_StepAbsolute](#)



[MCFB\\_StepRefPulse](#)[MCFB\\_StepBlock](#)[MCFB\\_StepLimitSwitch](#)

### 5.2.7.59.7 Example

#### 5.2.7.60.8.1 Structured Text

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

```
Inst_MCFB_StepAbsSwitch( True, Axis1, NegativeDirection,
RisingEdge, Velocity, 1000, 1000, 0, TorqueLimit, TimeLimit,
DistanceLimit, AbsoluteSwitch, PosLimitSwitch, NegLimitSwitch,
0 );
```

```
HomeComplete :=Inst_MCFB_StepAbsSwitch.Done;
```

```
HomeBusy :=Inst_MCFB_StepAbsSwitch.Busy;
```

```
HomeActive :=Inst_MCFB_StepAbsSwitch.Active;
```

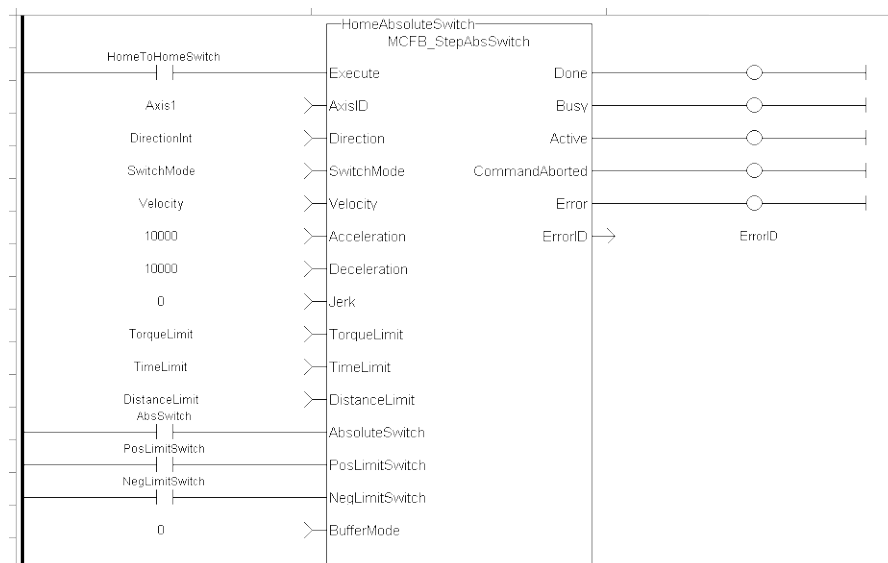
```
HomeAborted :=Inst_MCFB_StepAbsSwitch.CommandAborted;
```

```
HomeError :=Inst_MCFB_StepAbsSwitch.Error;
```

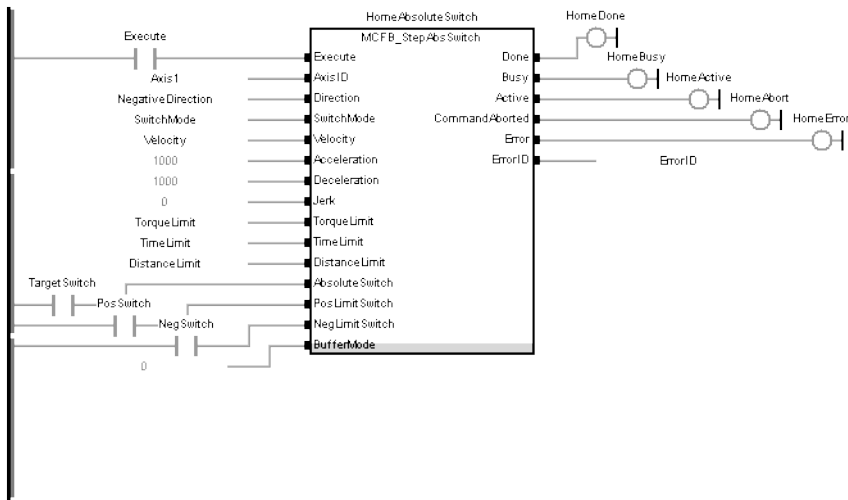
```
HomeErrorID :=Inst_MCFB_StepAbsSwitch.ErrorID;
```

```
(* AbsoluteSwitch, PosLimitSwitch, NegLimitSwitch are declared
I/O points *)
```

#### 5.2.7.61.9.2 Ladder Diagram



### 5.2.7.62.10.3 Function Block Diagram



**5.2.7.63 MCFB\_StepBlock** PLCopen

**5.2.7.64.1 Description**

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

The following figure shows the function block I/O:

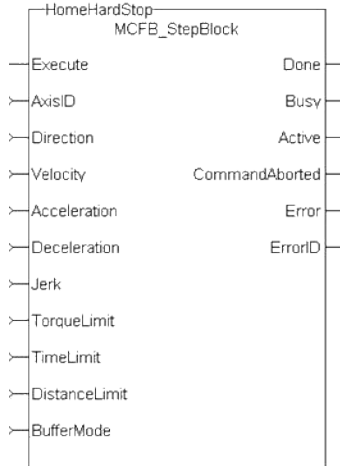


Figure 1-149: MCFB StepBlock

**5.2.7.65.2 Arguments**

**5.2.7.66.3.1 Input**

<b>Execute</b>	<b>Description</b>	Request the homing step procedure at rising edge. Outputs are reset when execute input is false.					
	<b>Data type</b>	BOOL					
	<b>Range</b>	[0 , 1]					
	<b>Unit</b>	n/a					
	<b>Default</b>	—					
<b>AxisID</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function					
	<b>Data type</b>	<a href="#">AXIS_REF</a>					
	<b>Range</b>	[1 , 256]					
	<b>Unit</b>	n/a					
	<b>Default</b>	—					
<b>Direction</b>	<b>Description</b>	Define the axis homing direction					
	<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>clockwise rotation</td> </tr> <tr> <td>1</td> <td>counterclockwise rotation</td> </tr> </tbody> </table>		Value	Description	0	clockwise rotation	1
Value	Description						
0	clockwise rotation						
1	counterclockwise rotation						

	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Velocity</b>	<b>Description</b>	Commanded velocity for the homing move
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	—
<b>Acceleration</b>	<b>Description</b>	Commanded acceleration for the homing move
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Deceleration</b>	<b>Description</b>	Commanded deceleration for the homing move
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Commanded jerk for the homing move (if zero, then trapezoidal acc/dec is used)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—
<b>TorqueLimit</b>	<b>Description</b>	Maximum torque applied for the homing move entered in thousandths of maximum torque, e.g. "250" is 250/1000, or 25%.
	<b>Data type</b>	LREAL
	<b>Range</b>	—

	<b>Unit</b>	User unit														
	<b>Default</b>	—														
<b>TimeLimit</b>	<b>Description</b>	Maximum time for homing move to complete. If exceeded the homing procedure will error out. 0= no time limit														
	<b>Data type</b>	TIME														
	<b>Range</b>	—														
	<b>Unit</b>	sec														
	<b>Default</b>	—														
<b>DistanceLimit</b>	<b>Description</b>	Maximum distance for homing move to complete. If exceeded the homing procedure will error out. 0= no distance limit														
	<b>Data type</b>	LREAL														
	<b>Range</b>	—														
	<b>Unit</b>	User unit														
	<b>Default</b>	—														
<b>BufferMode</b>	<b>Description</b>	Define the homing move start action														
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>abort</td> </tr> <tr> <td>1</td> <td>buffer</td> </tr> <tr> <td>2</td> <td>Blend to active</td> </tr> <tr> <td>3</td> <td>blend to next</td> </tr> <tr> <td>4</td> <td>blend to low velocity</td> </tr> <tr> <td>5</td> <td>blend to high velocity</td> </tr> </tbody> </table>	Value	Description	0	abort	1	buffer	2	Blend to active	3	blend to next	4	blend to low velocity	5	blend to high velocity
Value	Description															
0	abort															
1	buffer															
2	Blend to active															
3	blend to next															
4	blend to low velocity															
5	blend to high velocity															
	<b>Data type</b>	SINT														
	<b>Range</b>	[0 , 5]														
	<b>Unit</b>	n/a														
	<b>Default</b>	—														

#### 5.2.7.67.4.2 Output

<b>Done</b>	<b>Description</b>	Indicates the move completed successfully. The Command Position has reached the endpoint
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Busy</b>	<b>Description</b>	High from the moment the Execute input is one-shot to the time the move is ended

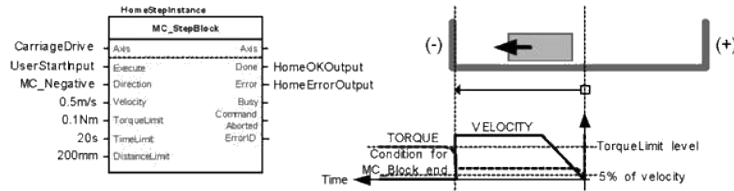
	<b>Data type</b>	BOOL														
	<b>Unit</b>	n/a														
<b>Active</b>	<b>Description</b>	Indicates this move is the active move														
	<b>Data type</b>	BOOL														
	<b>Unit</b>	n/a														
<b>CommandAborted</b>	<b>Description</b>	Indicates the move was aborted														
	<b>Data type</b>	BOOL														
	<b>Unit</b>	n/a														
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified or the move was terminated due to an error														
	<b>Data type</b>	BOOL														
	<b>Unit</b>	n/a														
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE														
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>TimeLimit exceeded</td> </tr> <tr> <td>2</td> <td>DistanceLimit exceeded</td> </tr> <tr> <td>3</td> <td></td> </tr> <tr> <td>4</td> <td>axis error stop state</td> </tr> <tr> <td>5</td> <td>axis not enabled</td> </tr> <tr> <td>6</td> <td>invalid inputs for Velocity-Acceleration-Deceleration</td> </tr> </tbody> </table>	Value	Description	1	TimeLimit exceeded	2	DistanceLimit exceeded	3		4	axis error stop state	5	axis not enabled	6	invalid inputs for Velocity-Acceleration-Deceleration
Value	Description															
1	TimeLimit exceeded															
2	DistanceLimit exceeded															
3																
4	axis error stop state															
5	axis not enabled															
6	invalid inputs for Velocity-Acceleration-Deceleration															
	<b>Data type</b>	INT														
	<b>Unit</b>	n/a														

**5.2.7.68.5 Usage**

Homing against a physical object, mechanically blocking the movement require adequate torque limits for not damaging mechanics during homing process. The StepBlock condition is that we have reached the torque limit and real velocity falls below 5% of demanded.

- Home is commanded by user in the desired homing direction at the selected Velocity
- Torque is limited.
- Time and Distance Limits can cause error if exceeded
- Process is finished when Torque is in limit condition and real velocity is below 5% of selected velocity.
- This Function Block doesn't modify actual position





### 5.2.7.69.6 Related Functions

[MCFB\\_StepAbsolute](#)

[MCFB\\_StepRefPulse](#)

[MCFB\\_StepAbsSwitch](#)

[MCFB\\_StepLimitSwitch](#)

### 5.2.7.70.7 Example

#### 5.2.7.71.8.1 Structured Text

```

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

```

```

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

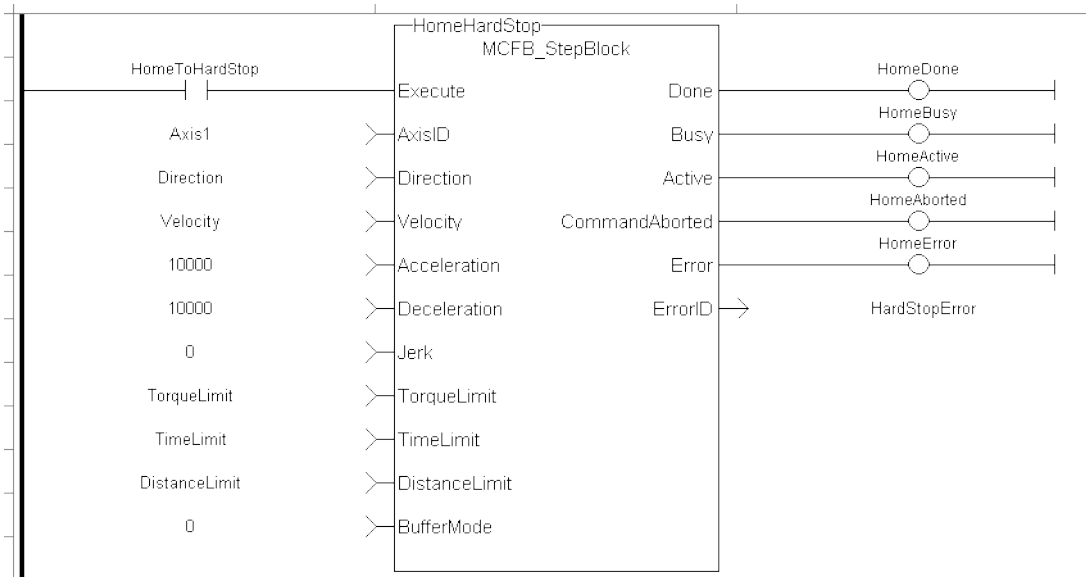
```

```

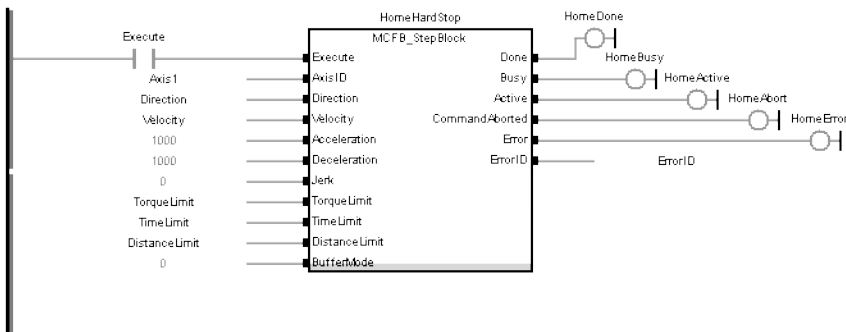
HomeComplete :=Inst_MCFB_StepBlock.Done;
HomeBusy :=Inst_MCFB_StepBlock.Busy;
HomeActive :=Inst_MCFB_StepBlock.Active;
HomeAborted :=Inst_MCFB_StepBlock.CommandAborted;
HomeError :=Inst_MCFB_StepBlock.Error;
HomeErrorID :=Inst_MCFB_StepBlock.ErrorID;

```

#### 5.2.7.72.9.2 Ladder Diagram



### 5.2.7.73.10.3 Function Block Diagram



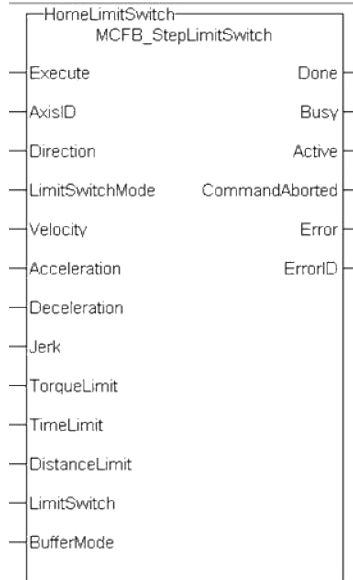


**5.2.7.74 MCFB\_StepLimitSwitch** PLCopen

**5.2.7.75.1 Description**

This function block performs a single-axis home to a limit switch. In this case the limit switches (always active once moving part working area has been surpassed) are used for homing procedure.

The following figure shows the function block I/O:



**Figure 1-150:** MCFB StepLimitSwitch

**5.2.7.76.2 Arguments**

**5.2.7.77.3.1 Input**

<b>Execute</b>	<b>Description</b>	Request the homing step procedure at rising edge. Outputs are reset when execute input is false.
	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxisID</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function
	<b>Data type</b>	<a href="#">AXIS_REF</a>
	<b>Range</b>	[1 , 256]
	<b>Unit</b>	n/a
	<b>Default</b>	—

<b>Direction</b>	<b>Description</b>	Define the axis homing direction										
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>clockwise rotation</td> </tr> <tr> <td>1</td> <td>counterclockwise rotation</td> </tr> </tbody> </table>	Value	Description	0	clockwise rotation	1	counterclockwise rotation				
	Value	Description										
	0	clockwise rotation										
	1	counterclockwise rotation										
<b>Data type</b>	BOOL											
<b>Range</b>	[0 , 1]											
<b>Unit</b>	n/a											
	<b>Default</b>	—										
<b>LimitSwitchMode</b>	<b>Description</b>	Limit switch state to complete homing										
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>switch is on</td> </tr> <tr> <td>1</td> <td>switch if off</td> </tr> <tr> <td>2</td> <td>rising edge of switch</td> </tr> <tr> <td>3</td> <td>falling edge of switch</td> </tr> </tbody> </table>	Value	Description	0	switch is on	1	switch if off	2	rising edge of switch	3	falling edge of switch
	Value	Description										
	0	switch is on										
	1	switch if off										
2	rising edge of switch											
3	falling edge of switch											
<b>Data type</b>	DINT											
<b>Range</b>	[0 , 3]											
<b>Unit</b>	n/a											
	<b>Default</b>	—										
<b>Velocity</b>	<b>Description</b>	Commanded velocity for the homing move										
	<b>Data type</b>	LREAL										
	<b>Range</b>	—										
	<b>Unit</b>	User unit/sec										
	<b>Default</b>	—										
<b>Acceleration</b>	<b>Description</b>	Commanded acceleration for the homing move										
	<b>Data type</b>	LREAL										
	<b>Range</b>	—										
	<b>Unit</b>	User unit/sec <sup>2</sup>										
	<b>Default</b>	—										
<b>Deceleration</b>	<b>Description</b>	Commanded deceleration for the homing move										
	<b>Data type</b>	LREAL										
	<b>Range</b>	—										

	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Commanded jerk for the homing move (if zero, then trapezoidal acc/dec is used)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—
<b>TorqueLimit</b>	<b>Description</b>	Maximum torque applied for the homing move entered in thousandths of maximum torque, e.g. "250" is 250/1000, or 25%.
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>TimeLimit</b>	<b>Description</b>	Maximum time for homing move to complete. If exceeded the homing procedure will error out. 0= no time limit
	<b>Data type</b>	TIME
	<b>Range</b>	—
	<b>Unit</b>	sec
	<b>Default</b>	—
<b>DistanceLimit</b>	<b>Description</b>	Maximum distance for homing move to complete. If exceeded the homing procedure will error out. 0= no distance limit
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>LimitSwitch</b>	<b>Description</b>	The limit switch input I/O point
	<b>Data type</b>	BOOL
	<b>Range</b>	[0, 1]
	<b>Unit</b>	n/a

	<b>Default</b>	—														
<b>BufferMode</b>	<b>Description</b>	Define the homing move start action														
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>abort</td> </tr> <tr> <td>1</td> <td>buffer</td> </tr> <tr> <td>2</td> <td>Blend to active</td> </tr> <tr> <td>3</td> <td>blend to next</td> </tr> <tr> <td>4</td> <td>blend to low velocity</td> </tr> <tr> <td>5</td> <td>blend to high velocity</td> </tr> </tbody> </table>	Value	Description	0	abort	1	buffer	2	Blend to active	3	blend to next	4	blend to low velocity	5	blend to high velocity
	Value	Description														
	0	abort														
	1	buffer														
	2	Blend to active														
	3	blend to next														
	4	blend to low velocity														
5	blend to high velocity															
<b>Data type</b>	SINT															
<b>Range</b>	[0 , 5]															
<b>Unit</b>	n/a															
<b>Default</b>	—															

#### 5.2.7.78.4.2 Output

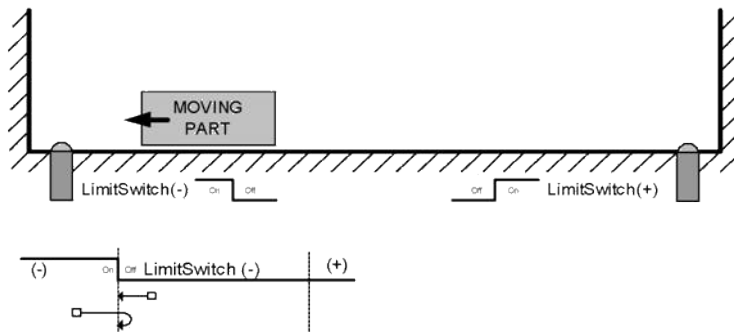
<b>Done</b>	<b>Description</b>	Indicates the move completed successfully. The Command Position has reached the endpoint
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Busy</b>	<b>Description</b>	High from the moment the Execute input is one-shot to the time the move is ended
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Active</b>	<b>Description</b>	Indicates this move is the active move
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>CommandAborted</b>	<b>Description</b>	Indicates the move was aborted
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified or the move was terminated due to an error
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

ErrorID	Description	Indicates the error if Error output is set to TRUE	
		Value	Description
		1	TimeLimit exceeded
		2	DistanceLimit exceeded
		3	TorqueLimit exceeded
		4	axis error stop state
		5	axis not enabled
		6	invalid inputs for Velocity-Acceleration-Deceleration
	<b>Data type</b>	INT	
	<b>Unit</b>	n/a	

### 5.2.7.79.5 Usage

This homing procedure performs a homing function searching for sensor using only LimitSwitches. (A LimitSwitch has 1 "Off" (or "On") area).

- Home is commanded by user in the desired homing direction at the selected Velocity.
- If LimitSwitch is found 'On' on rising 'Execute', then the process is started in the opposite direction as specified, LimitSwitch is search for 'Off' (or On, depending on LimitSwitchMode setting) Edge (released), and process is restarted again in original direction. This ensures that the end conditions are always the same.
- The torque is limited.
- The Time and Distance Limits can cause error if exceeded
- The Direction changes only when the specified Velocity is reached, this ensures acceleration and deceleration spaces are fixed
- This Function Block doesn't modify actual position



### 5.2.7.80.6 Related Functions

[MCFB\\_StepAbsolute](#)

[MCFB\\_StepRefPulse](#)

[MCFB\\_StepBlock](#)

[MCFB\\_StepAbsSwitch](#)

### 5.2.7.81.7 Example

### 5.2.7.82.8.1 Structured Text

```

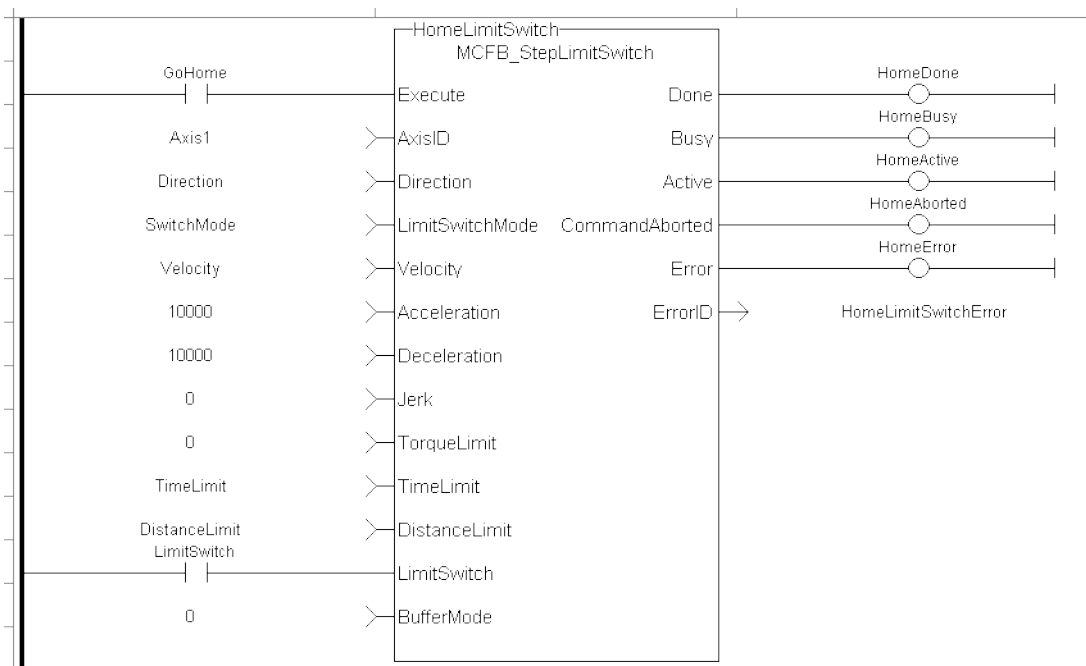
PositiveDirection :=0;
RisingEdge :=2;
Velocity :=10000.0;
TorqueLimit :=50.0;
TimeLimit :=T#10s;
DistanceLimit :=10000.0;

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

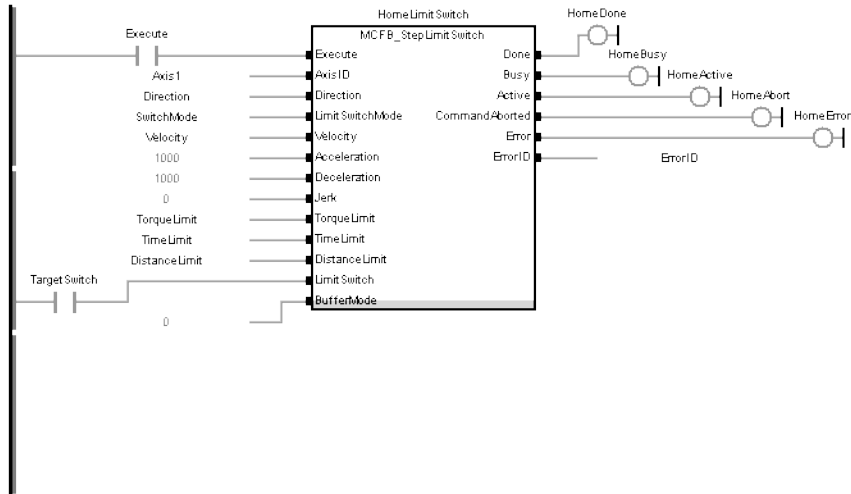
HomeComplete :=Inst_MCFB_StepLimitSwitch.Done;
HomeBusy :=Inst_MCFB_StepLimitSwitch.Busy;
HomeActive :=Inst_MCFB_StepLimitSwitch.Active;
HomeAborted :=Inst_MCFB_StepLimitSwitch.CommandAborted;
HomeError :=Inst_MCFB_StepLimitSwitch.Error;
HomeErrorID :=Inst_MCFB_StepLimitSwitch.ErrorID;

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

**5.2.7.83.9.2 Ladder Diagram**



**5.2.7.84.10.3 Function Block Diagram**

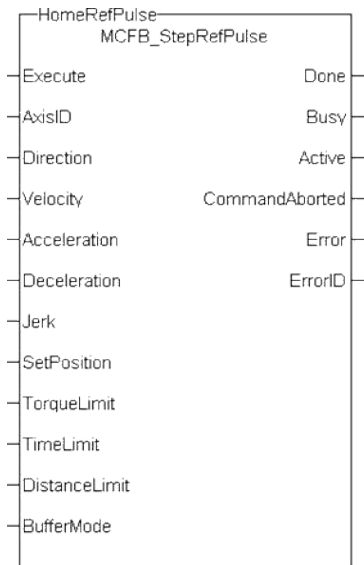


**5.2.7.85 MCFB\_StepRefPulse** PLCopen

**5.2.7.86.1 Description**

This function block performs homing by searching for Zero pulse (also called Marker or reference pulse) in encoder. The reference pulse appears once per encoder revolution. The advantage in using Reference Pulse for homing is the higher accuracy and precision that can be achieved compared to traditional optical, mechanical or magnetic sensors.

The following figure shows the function block I/O:



**Figure 1-151:** MCFB StepRefPulse

**5.2.7.87.2 Arguments**

**5.2.7.88.3.1 Input**

<b>Execute</b>	<b>Description</b>	Request the homing step procedure at rising edge
	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxisID</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function
	<b>Data type</b>	<a href="#">AXIS_REF</a>
	<b>Range</b>	[1 , 256]
	<b>Unit</b>	n/a
	<b>Default</b>	—



<b>Direction</b>	<b>Description</b>	Define the axis homing direction										
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>clockwise rotation</td> </tr> <tr> <td>1</td> <td>counterclockwise rotation</td> </tr> </tbody> </table>	Value	Description	0	clockwise rotation	1	counterclockwise rotation				
	Value	Description										
	0	clockwise rotation										
	1	counterclockwise rotation										
	<b>Data type</b>	BOOL										
<b>Range</b>	[0 , 1]											
<b>Unit</b>	n/a											
<b>Default</b>	—											
<b>SwitchMode</b>	<b>Description</b>	Switch state to complete homing										
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>switch is on</td> </tr> <tr> <td>1</td> <td>switch if off</td> </tr> <tr> <td>2</td> <td>rising edge of switch</td> </tr> <tr> <td>3</td> <td>falling edge of switch</td> </tr> </tbody> </table>	Value	Description	0	switch is on	1	switch if off	2	rising edge of switch	3	falling edge of switch
	Value	Description										
	0	switch is on										
	1	switch if off										
	2	rising edge of switch										
3	falling edge of switch											
<b>Data type</b>	DINT											
<b>Range</b>	[0 , 3]											
<b>Unit</b>	n/a											
<b>Default</b>	—											
<b>Velocity</b>	<b>Description</b>	Commanded velocity for the homing move										
	<b>Data type</b>	LREAL										
	<b>Range</b>	—										
	<b>Unit</b>	User unit/sec										
	<b>Default</b>	—										
<b>Acceleration</b>	<b>Description</b>	Commanded acceleration for the homing move										
	<b>Data type</b>	LREAL										
	<b>Range</b>	—										
	<b>Unit</b>	User unit/sec <sup>2</sup>										
	<b>Default</b>	—										
<b>Deceleration</b>	<b>Description</b>	Commanded deceleration for the homing move										
	<b>Data type</b>	LREAL										
	<b>Range</b>	—										

	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Commanded jerk for the homing move (if zero, then trapezoidal acc/dec is used)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—
<b>SetPosition</b>	<b>Description</b>	Value of the absolute position to be set when the homing move is done
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>TorqueLimit</b>	<b>Description</b>	Maximum torque applied for the homing move
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>TimeLimit</b>	<b>Description</b>	Maximum time for homing move to complete. If exceeded the homing procedure will error out. 0= no time limit
	<b>Data type</b>	TIME
	<b>Range</b>	—
	<b>Unit</b>	sec
	<b>Default</b>	—
<b>DistanceLimit</b>	<b>Description</b>	Maximum distance for homing move to complete. If exceeded the homing procedure will error out. 0= no distance limit
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—

<b>BufferMode</b>	<b>Description</b>	Define the homing move start action														
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>abort</td> </tr> <tr> <td>1</td> <td>buffer</td> </tr> <tr> <td>2</td> <td>Blend to active</td> </tr> <tr> <td>3</td> <td>blend to next</td> </tr> <tr> <td>4</td> <td>blend to low velocity</td> </tr> <tr> <td>5</td> <td>blend to high velocity</td> </tr> </tbody> </table>	Value	Description	0	abort	1	buffer	2	Blend to active	3	blend to next	4	blend to low velocity	5	blend to high velocity
	Value	Description														
	0	abort														
	1	buffer														
	2	Blend to active														
	3	blend to next														
4	blend to low velocity															
5	blend to high velocity															
<b>Data type</b>	SINT															
<b>Range</b>	[0 , 5]															
<b>Unit</b>	n/a															
<b>Default</b>	—															

#### 5.2.7.89.4.2 Output

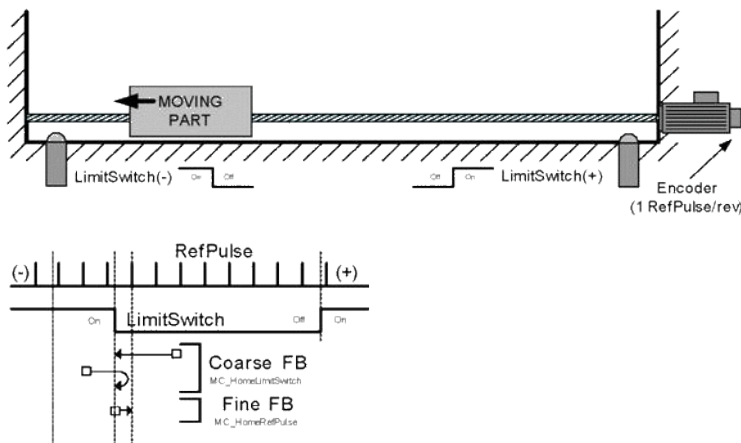
<b>Done</b>	<b>Description</b>	Indicates the move completed successfully. The Command Position has reached the endpoint
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Busy</b>	<b>Description</b>	High from the moment the Execute input is one-shot to the time the move is ended
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Active</b>	<b>Description</b>	Indicates this move is the active move
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>CommandAborted</b>	<b>Description</b>	Indicates the move was aborted
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Error</b>	<b>Description</b>	Indicates an invalid input was specified or the move was terminated due to an error
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

ErrorID	Description
	Indicates the error if Error output is set to TRUE
Value	Description
1	TimeLimit exceeded
2	DistanceLimit exceeded
3	TorqueLimit exceeded
4	axis error stop state
5	axis not enabled
6	invalid inputs for Velocity-Acceleration-Deceleration
<b>Data type</b>	INT
<b>Unit</b>	n/a

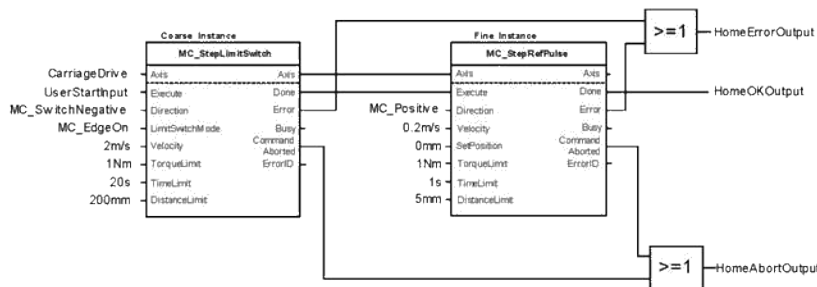
### 5.2.7.90.5 Usage

This function Block performs homing by searching for Zero pulse (also called Marker or reference pulse) in encoder. The reference pulse appears once per encoder revolution.

- Home is commanded by user in the desired homing direction at the programmed velocity.
- First occurrence of the Reference Pulse, Homing is finished
- Torque is limited. Time and Distance Limits can cause error if exceeded
- This Function modifies actual position and sets to the "SetPosition" input value at the end



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



### 5.2.7.91.6 Related Functions

[MCFB\\_StepAbsolute](#)

[MCFB\\_StepAbsSwitch](#)[MCFB\\_StepBlock](#)[MCFB\\_StepLimitSwitch](#)**5.2.7.92.7 Example****5.2.7.93.8.1 Structured Text**

```

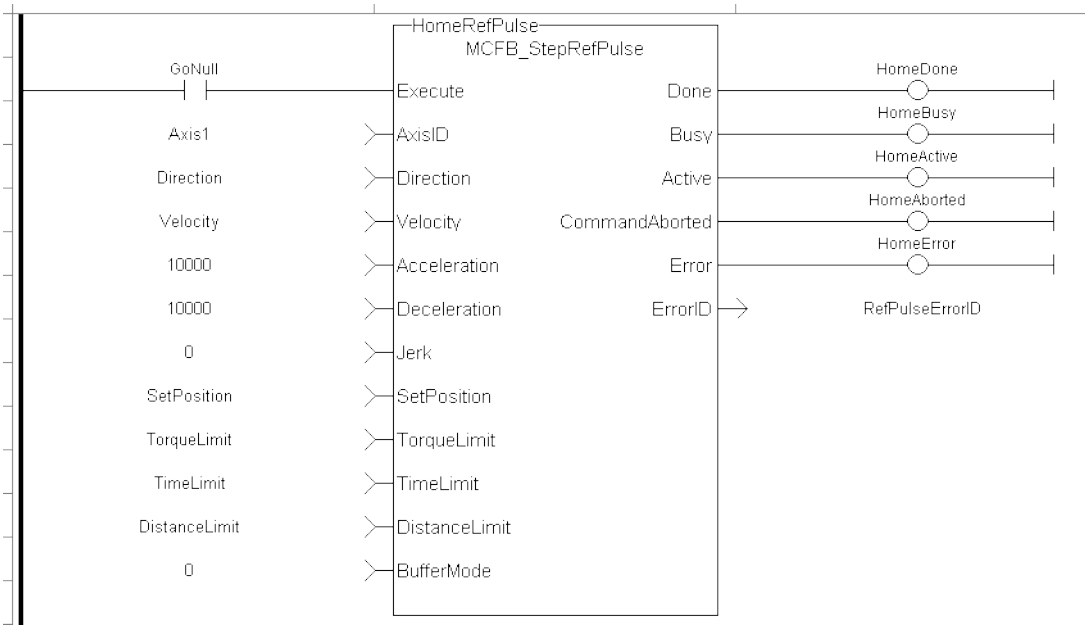
PositiveDirection :=0;
Velocity :=10000.0;
SetPosition :=0.0;
TorqueLimit :=50.0;
TimeLimit :=T#10s;
DistanceLimit :=10000.0;

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

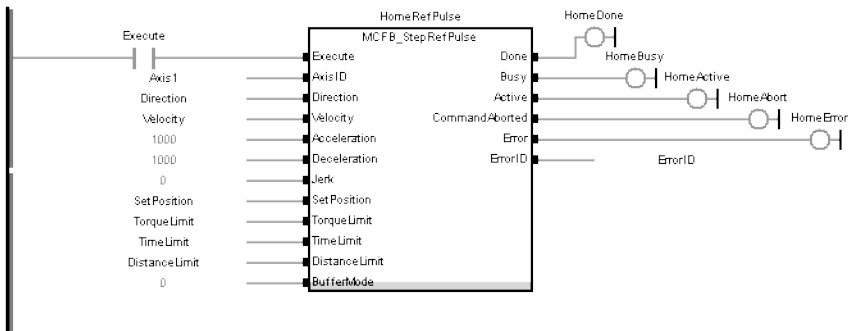
HomeComplete :=Inst_MCFB_StepRefPulse.Done;
HomeBusy :=Inst_MCFB_StepRefPulse.Busy;
HomeActive :=Inst_MCFB_StepRefPulse.Active;
HomeAborted :=Inst_MCFB_StepRefPulse.CommandAborted;
HomeError :=Inst_MCFB_StepRefPulse.Error;
HomeErrorID :=Inst_MCFB_StepRefPulse.ErrorID;

```

**5.2.7.94.9.2 Ladder Diagram**



### 5.2.7.95.10.3 Function Block Diagram

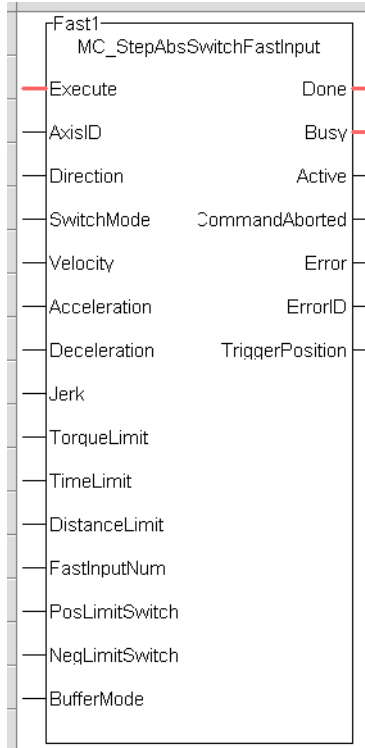


**5.2.7.96 MCFB\_StepAbsSwitchFastInput** PLCopen ✔

**5.2.7.97.1 Description**

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

The following figure shows the function block I/O:



**Figure 1-152:** MCFB StepAbsSwitchFastInput

**5.2.7.98.2.1 Input**

<b>Execute</b>	<b>Description</b>	Request the homing step procedure at rising edge. Outputs are reset when execute input is false.
	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxisID</b>	<b>Description</b>	Structure for specified Axis desired to home
	<b>Data type</b>	<a href="#">AXIS_REF</a>
	<b>Range</b>	[1 , 256]
	<b>Unit</b>	n/a

	<b>Default</b>	—										
<b>Direction</b>	<b>Description</b>	Define the axis homing direction										
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>clockwise rotation</td> </tr> <tr> <td>1</td> <td>counterclockwise rotation</td> </tr> </tbody> </table>	Value	Description	0	clockwise rotation	1	counterclockwise rotation				
	Value	Description										
	0	clockwise rotation										
	1	counterclockwise rotation										
<b>Data type</b>	BOOL											
<b>Range</b>	[0 , 1]											
<b>Unit</b>	n/a											
	<b>Default</b>	—										
<b>SwitchMode</b>	<b>Description</b>	Switch state to complete homing										
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>when rising edge of sensor</td> </tr> <tr> <td>1</td> <td>when falling edge</td> </tr> <tr> <td>2</td> <td>rising edge when traveling in positive direction but falling edge in negative direction</td> </tr> <tr> <td>3</td> <td>falling edge when traveling in negative direction but rising edge in positive direction</td> </tr> </tbody> </table>	Value	Description	0	when rising edge of sensor	1	when falling edge	2	rising edge when traveling in positive direction but falling edge in negative direction	3	falling edge when traveling in negative direction but rising edge in positive direction
	Value	Description										
	0	when rising edge of sensor										
	1	when falling edge										
	2	rising edge when traveling in positive direction but falling edge in negative direction										
3	falling edge when traveling in negative direction but rising edge in positive direction											
<b>Data type</b>	DINT											
<b>Range</b>	[0 , 3]											
<b>Unit</b>	n/a											
	<b>Default</b>	—										
<b>Velocity</b>	<b>Description</b>	Commanded velocity for the homing move										
	<b>Data type</b>	LREAL										
	<b>Range</b>	—										
	<b>Unit</b>	User unit/sec										
	<b>Default</b>	—										
<b>Acceleration</b>	<b>Description</b>	Commanded acceleration for the homing move										
	<b>Data type</b>	LREAL										
	<b>Range</b>	—										
	<b>Unit</b>	User unit/sec <sup>2</sup>										
	<b>Default</b>	—										
<b>Deceleration</b>	<b>Description</b>	Commanded deceleration for the homing move										



	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Commanded jerk for the homing move (if zero, then trapezoidal acc/dec is used)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—
<b>TorqueLimit</b>	<b>Description</b>	Maximum torque applied for the homing move entered in thousandths of maximum torque, e.g. "250" is 250/1000, or 25%.
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>TimeLimit</b>	<b>Description</b>	Maximum time for homing move to complete. If exceeded the homing procedure will error out. 0= no time limit
	<b>Data type</b>	TIME
	<b>Range</b>	—
	<b>Unit</b>	sec
	<b>Default</b>	—
<b>DistanceLimit</b>	<b>Description</b>	Maximum distance for homing move to complete. If exceeded the homing procedure will error out. 0= no distance limit
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>FastInputNum</b>	<b>Description</b>	0 for first fast input (X7 Pin 10), 1 for second fast input (X7 pin 9)
	<b>Data type</b>	BOOL

	<b>Range</b>	[0 , 1]														
	<b>Unit</b>	n/a														
	<b>Default</b>	—														
<b>PosLimitSwitch</b>	<b>Description</b>	The positive direction limit switch input I/O point														
	<b>Data type</b>	BOOL														
	<b>Range</b>	[0 , 1]														
	<b>Unit</b>	n/a														
	<b>Default</b>	—														
<b>NegLimitSwitch</b>	<b>Description</b>	The negative direction limit switch input I/O point														
	<b>Data type</b>	BOOL														
	<b>Range</b>	[0 , 1]														
	<b>Unit</b>	n/a														
	<b>Default</b>	—														
<b>BufferMode</b>	<b>Description</b>	Define the homing move start action														
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>abort</td> </tr> <tr> <td>1</td> <td>buffer</td> </tr> <tr> <td>2</td> <td>Blend to active</td> </tr> <tr> <td>3</td> <td>blend to next</td> </tr> <tr> <td>4</td> <td>blend to low velocity</td> </tr> <tr> <td>5</td> <td>blend to high velocity</td> </tr> </tbody> </table>	Value	Description	0	abort	1	buffer	2	Blend to active	3	blend to next	4	blend to low velocity	5	blend to high velocity
Value	Description															
0	abort															
1	buffer															
2	Blend to active															
3	blend to next															
4	blend to low velocity															
5	blend to high velocity															
	<b>Data type</b>	SINT														
	<b>Range</b>	[0 , 5]														
	<b>Unit</b>	n/a														
	<b>Default</b>	—														

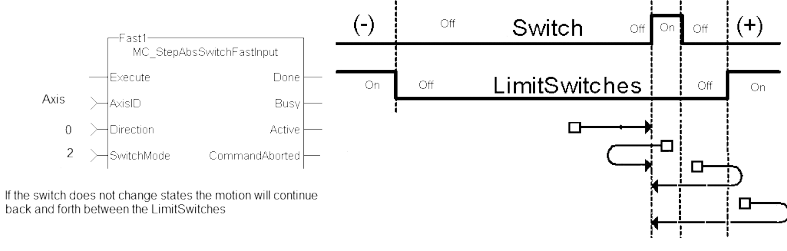
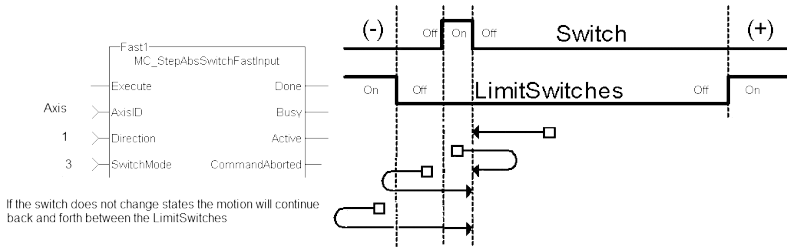
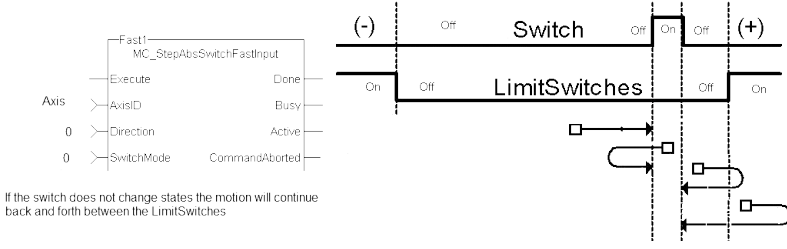
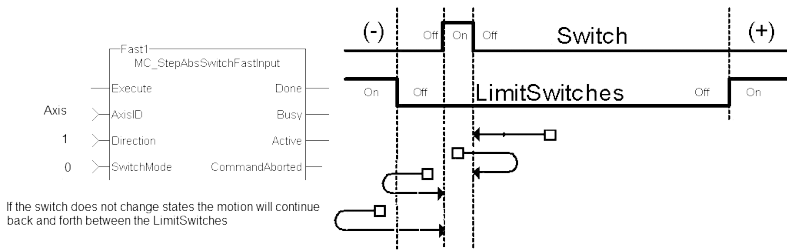
#### 5.2.7.99.3.2 Output

<b>Done</b>	<b>Description</b>	Indicates the move completed successfully. The Command Position has reached the endpoint
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Busy</b>	<b>Description</b>	High from the moment the Execute input is one-shot to the time the move is ended

	<b>Data type</b>	BOOL														
	<b>Unit</b>	n/a														
<b>Active</b>	<b>Description</b>	Set when the function block is active														
	<b>Data type</b>	BOOL														
	<b>Unit</b>	n/a														
<b>CommandAborted</b>	<b>Description</b>	Indicates the move was aborted														
	<b>Data type</b>	BOOL														
	<b>Unit</b>	n/a														
<b>Error</b>	<b>Description</b>	Signals that an error has occurred within the function block														
	<b>Data type</b>	BOOL														
	<b>Unit</b>	n/a														
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE														
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>TimeLimit exceeded</td> </tr> <tr> <td>2</td> <td>DistanceLimit exceeded</td> </tr> <tr> <td>3</td> <td>TorqueLimit exceeded</td> </tr> <tr> <td>4</td> <td>axis error stop state</td> </tr> <tr> <td>5</td> <td>axis not enabled</td> </tr> <tr> <td>6</td> <td>invalid inputs for Velocity-Accel-Decel</td> </tr> </tbody> </table>	Value	Description	1	TimeLimit exceeded	2	DistanceLimit exceeded	3	TorqueLimit exceeded	4	axis error stop state	5	axis not enabled	6	invalid inputs for Velocity-Accel-Decel
Value	Description															
1	TimeLimit exceeded															
2	DistanceLimit exceeded															
3	TorqueLimit exceeded															
4	axis error stop state															
5	axis not enabled															
6	invalid inputs for Velocity-Accel-Decel															
	<b>Data type</b>	INT														
	<b>Unit</b>	n/a														
<b>TriggerPosition</b>	<b>Data type</b>	LREAL														
	<b>Range</b>	-														
	<b>Unit</b>	User units														
	<b>Default</b>	-														

#### 5.2.7.100.4 Usage

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



**5.2.7.101.5 Related Functions**

[MCFB\\_StepLimitSwitchFastInput](#)

**5.2.7.102.6 Example**

**5.2.7.103.7.1 Structured Text**

```
Execute_1 :=1;
```

```
(*Positive_Switch and Negitive_Switch are physical hardware in Dictionary. *)
```

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

```
HomeComplete := Inst_MC_StepAbsSwitchFastInput.Done;
```

```
HomeBusy := Inst_MC_StepAbsSwitchFastInput.Busy;
```

```
HomeActive := Inst_MC_StepAbsSwitchFastInput.Active;
```

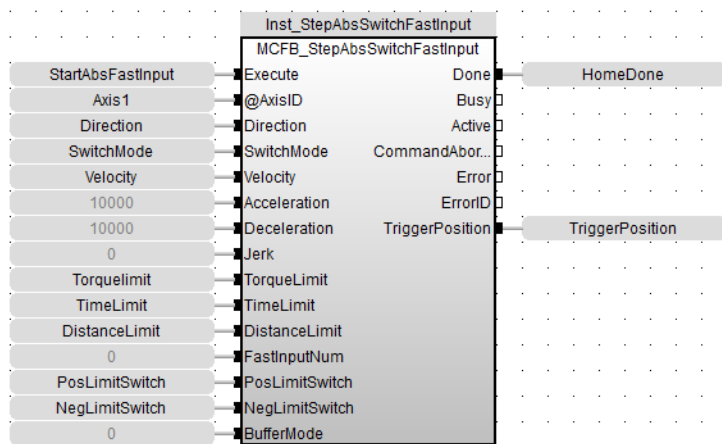
```
HomeAborted := Inst_MC_StepAbsSwitchFastInput.CommandAborted;
```

```
HomeError := Inst_MC_StepAbsSwitchFastInput.Error;
```

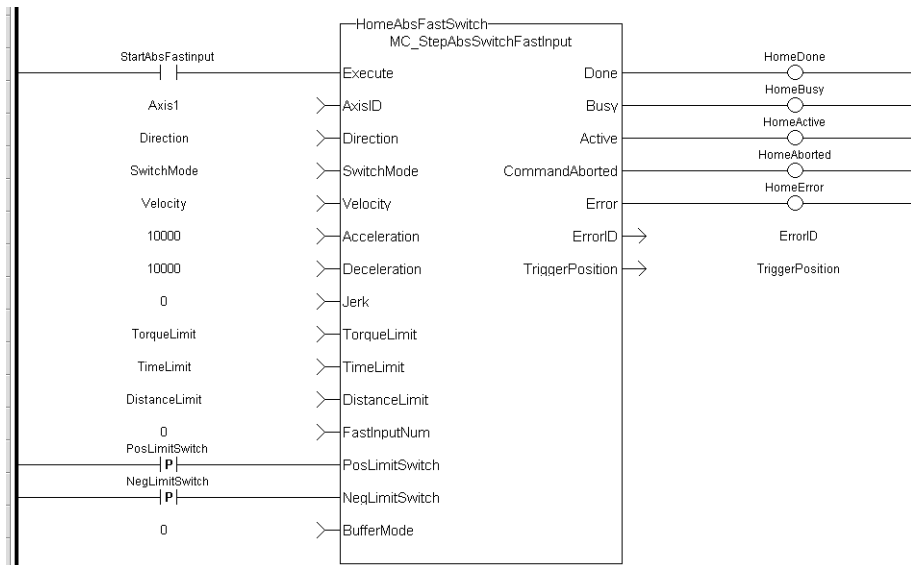
```
HomeErrorID := Inst_MC_StepAbsSwitchFastInput.ErrorID;
```

```
HomeTriggerPosition := Inst_MC_
StepAbsSwitchFastInput.TriggerPosition;
```

### 5.2.7.104.8.2 FBD



### 5.2.7.105.9.3 Ladder Diagram



(\* PosLimitSwitch, NegLimitSwitch are declared I/O points \*)

**5.2.7.106 MCFB\_StepLimitSwitchFastInput** PLCopen ✓

**5.2.7.107.1 Description**

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

The following figure shows the function block I/O:

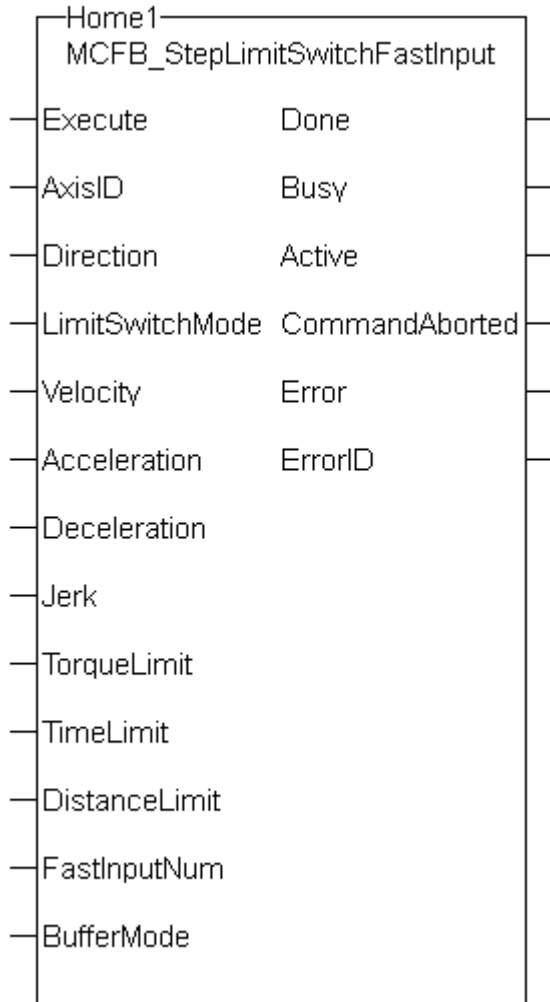


Figure 1-153: MCFB StepLimitSwitchFastInput

**5.2.7.108.2.1 Input**

<b>Execute</b>	<b>Description</b>	Request the homing step procedure at rising edge. Outputs are reset when execute input is false.
	<b>Data type</b>	BOOL
	<b>Range</b>	[0, 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxisID</b>	<b>Description</b>	Structure for specified Axis desired to home

	<b>Data type</b>	<a href="#">AXIS_REF</a>						
	<b>Range</b>	[1 , 256]						
	<b>Unit</b>	n/a						
	<b>Default</b>	—						
<b>Direction</b>	<b>Description</b>	Define the axis homing direction						
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>clockwise rotation</td> </tr> <tr> <td>1</td> <td>counterclockwise rotation</td> </tr> </tbody> </table>	Value	Description	0	clockwise rotation	1	counterclockwise rotation
Value	Description							
0	clockwise rotation							
1	counterclockwise rotation							
	<b>Data type</b>	BOOL						
	<b>Range</b>	[0 , 1]						
	<b>Unit</b>	n/a						
	<b>Default</b>	—						
<b>LimitSwitchMode</b>	<b>Description</b>	Limit switch state to complete homing						
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>when rising edge of sensor</td> </tr> <tr> <td>1</td> <td>when falling edge</td> </tr> </tbody> </table>	Value	Description	0	when rising edge of sensor	1	when falling edge
Value	Description							
0	when rising edge of sensor							
1	when falling edge							
	<b>Data type</b>	DINT						
	<b>Range</b>	[0 , 1]						
	<b>Unit</b>	n/a						
	<b>Default</b>	—						
<b>Velocity</b>	<b>Description</b>	Commanded velocity for the homing move						
	<b>Data type</b>	LREAL						
	<b>Range</b>	—						
	<b>Unit</b>	User unit/sec						
	<b>Default</b>	—						
<b>Acceleration</b>	<b>Description</b>	Commanded acceleration for the homing move						
	<b>Data type</b>	LREAL						
	<b>Range</b>	—						
	<b>Unit</b>	User unit/sec <sup>2</sup>						
	<b>Default</b>	—						



<b>Deceleration</b>	<b>Description</b>	Commanded deceleration for the homing move
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—
<b>Jerk</b>	<b>Description</b>	Commanded jerk for the homing move (if zero, then trapezoidal acc/dec is used)
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>3</sup>
	<b>Default</b>	—
<b>TorqueLimit</b>	<b>Description</b>	Maximum torque applied for the homing move
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit entered in thousandths of maximum torque, e.g. "250" is 250/1000, or 25%.
	<b>Default</b>	—
<b>TimeLimit</b>	<b>Description</b>	Maximum time for homing move to complete. If exceeded the homing procedure will error out. 0= no time limit
	<b>Data type</b>	TIME
	<b>Range</b>	—
	<b>Unit</b>	sec
	<b>Default</b>	—
<b>DistanceLimit</b>	<b>Description</b>	Maximum distance for homing move to complete. If exceeded the homing procedure will error out. 0= no distance limit
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit
	<b>Default</b>	—
<b>FastInputNum</b>	<b>Description</b>	0 for first fast input (X7 Pin 10), 1 for second fast input (X7 pin 9)

	<b>Data type</b>	BOOL														
	<b>Range</b>	[0 , 1]														
	<b>Unit</b>	n/a														
	<b>Default</b>	—														
<b>BufferMode</b>	<b>Description</b>	Define the homing move start action														
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>abort</td> </tr> <tr> <td>1</td> <td>buffer</td> </tr> <tr> <td>2</td> <td>Blend to active</td> </tr> <tr> <td>3</td> <td>blend to next</td> </tr> <tr> <td>4</td> <td>blend to low velocity</td> </tr> <tr> <td>5</td> <td>blend to high velocity</td> </tr> </tbody> </table>	Value	Description	0	abort	1	buffer	2	Blend to active	3	blend to next	4	blend to low velocity	5	blend to high velocity
Value	Description															
0	abort															
1	buffer															
2	Blend to active															
3	blend to next															
4	blend to low velocity															
5	blend to high velocity															
	<b>Data type</b>	SINT														
	<b>Range</b>	[0 , 5]														
	<b>Unit</b>	n/a														
	<b>Default</b>	—														

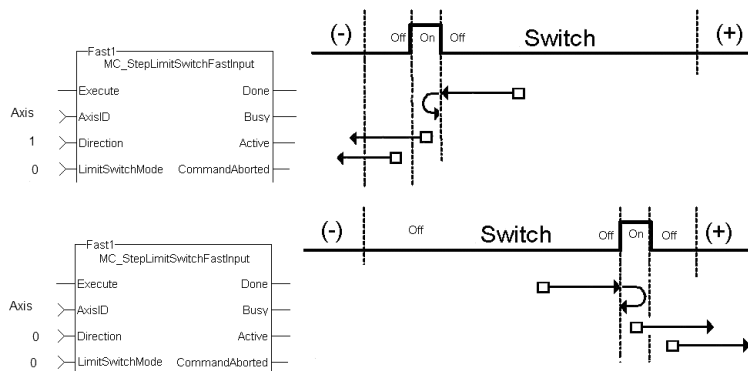
#### 5.2.7.109.3.2 Output

<b>Done</b>	<b>Description</b>	Indicates the move completed successfully. The Command Position has reached the endpoint
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Busy</b>	<b>Description</b>	High from the moment the Execute input is one-shot to the time the move is ended
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Active</b>	<b>Description</b>	Set when the function block is active
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>CommandAborted</b>	<b>Description</b>	Indicates the move was aborted
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>Error</b>	<b>Description</b>	Signals that an error has occurred within the function block

	<b>Data type</b>	BOOL														
	<b>Unit</b>	n/a														
<b>ErrorID</b>	<b>Description</b>	Indicates the error if Error output is set to TRUE														
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>TimeLimit exceeded</td> </tr> <tr> <td>2</td> <td>DistanceLimit exceeded</td> </tr> <tr> <td>3</td> <td>TorqueLimit exceeded</td> </tr> <tr> <td>4</td> <td>axis error stop state</td> </tr> <tr> <td>5</td> <td>axis not enabled</td> </tr> <tr> <td>6</td> <td>invalid inputs for Velocity-Accel-Decel</td> </tr> </tbody> </table>	Value	Description	1	TimeLimit exceeded	2	DistanceLimit exceeded	3	TorqueLimit exceeded	4	axis error stop state	5	axis not enabled	6	invalid inputs for Velocity-Accel-Decel
Value	Description															
1	TimeLimit exceeded															
2	DistanceLimit exceeded															
3	TorqueLimit exceeded															
4	axis error stop state															
5	axis not enabled															
6	invalid inputs for Velocity-Accel-Decel															
	<b>Data type</b>	INT														
	<b>Unit</b>	n/a														

#### 5.2.7.110.4 Usage

The homing is commanded in the most likely direction were the sensor can be found. In this example (-).



#### 5.2.7.111.5 Related Functions

[MCFB\\_StepAbsSwitchFastInput](#)

#### 5.2.7.112.6 Example

##### 5.2.7.113.7.1 Structured Text

```

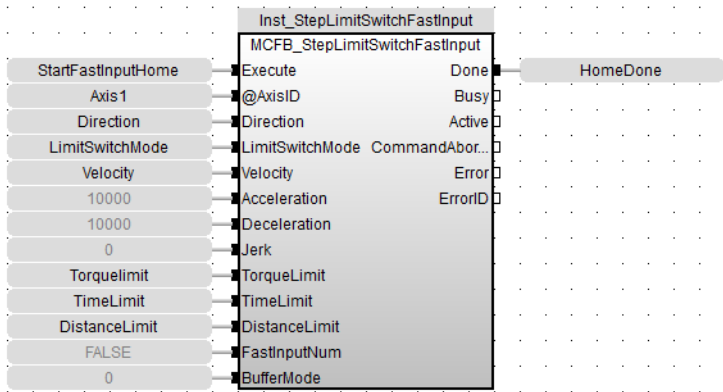
Execute_1 :=1;

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

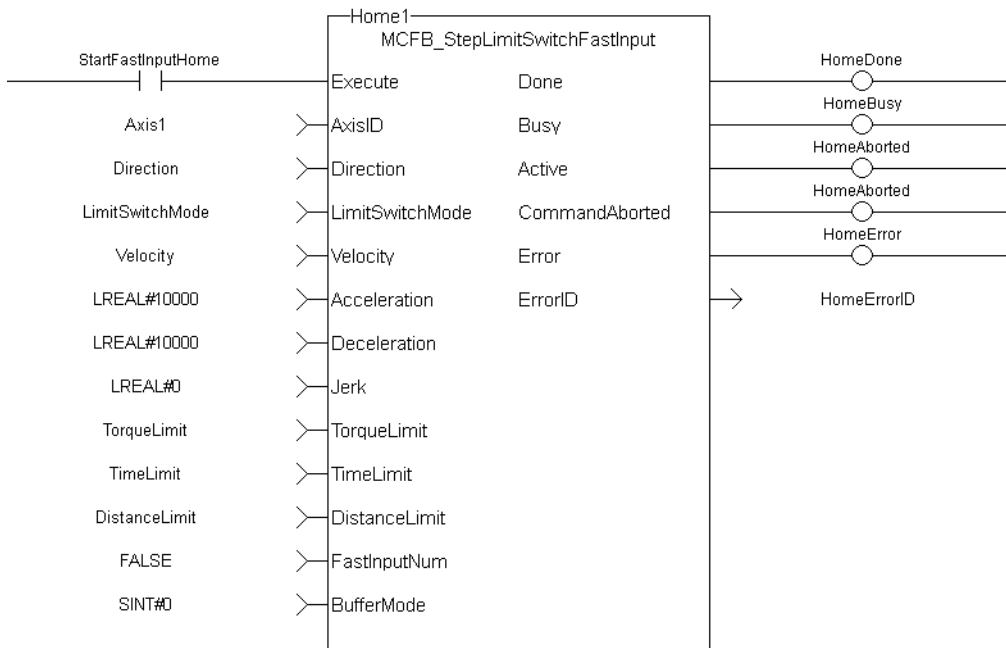
HomeComplete := Inst_MCFB_StepLimitSwitchFastInput.Done;
HomeBusy := Inst_MCFB_StepLimitSwitchFastInput.Busy;
HomeActive := Inst_MCFB_StepLimitSwitchFastInput.Active;
HomeAborted := Inst_MCFB_StepLimitSwitchFastInput.CommandAborted;
HomeError := Inst_MCFB_StepLimitSwitchFastInput.Error;
HomeErrorID :=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

```

### 5.2.7.114.8.2 FBD



### 5.2.7.115.9.3 Ladder Diagram



## 5.2.8 Jog for PLCopen

### 5.2.8.1 Description

This function block is defined to jog an axis in the selected direction at a defined speed. The En input (FFLD editor only) must be high. Typically wired to the rail.

The AxisID selects the axis to jog. The JogPlus and JogMinus inputs select the direction the motion will occur in. Only one of these inputs should be enabled at a given time. If both are selected the motion will stop. If other motion is active when the jog is requested that motion will be aborted and the jog will start.

The following figure shows the function block I/O

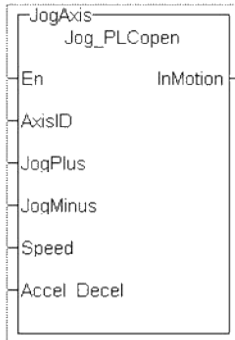


Figure 1-154: Jog for PLCopen

### 5.2.8.2 Arguments

#### 5.2.8.3.1 Input

<b>En</b>	<b>Description</b>	Enables execution (FFLD only )
	<b>Data type</b>	BOOL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>AxisID</b>	<b>Description</b>	ID Name of the Axis
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>JogPlus</b>	<b>Description</b>	Enables a Jog in the plus direction
	<b>Data type</b>	BOOL

	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>JogMinus</b>	<b>Description</b>	Enables a Jog in the Minus direction
	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Speed</b>	<b>Description</b>	Rate at which the axis will move
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec
	<b>Default</b>	—
<b>Accel Decel</b>	<b>Description</b>	Linear Acc/Dec rate
	<b>Data type</b>	LREAL
	<b>Range</b>	—
	<b>Unit</b>	User unit/sec <sup>2</sup>
	<b>Default</b>	—

#### 5.2.8.4.2 Output

<b>InMotion</b>	<b>Description</b>	Jogging is active when TRUE
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

#### 5.2.8.5 Usage

This function Block is used to command motion in a designated direction at a defined rate. This may be used where continuous motion required as in a conveyor system, or in a setup mode for manually jogging the axis. Motion will start when the JogPlus or JogMinus input is true. It will stop when the input goes false.

#### 5.2.8.6 Related Functions

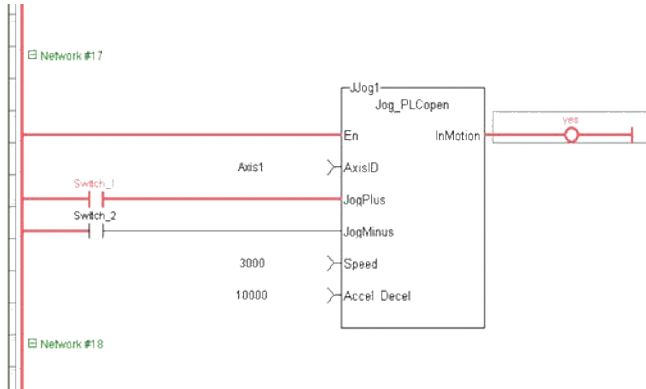
[MC\\_MoveVelocity](#)

#### 5.2.8.7 Example

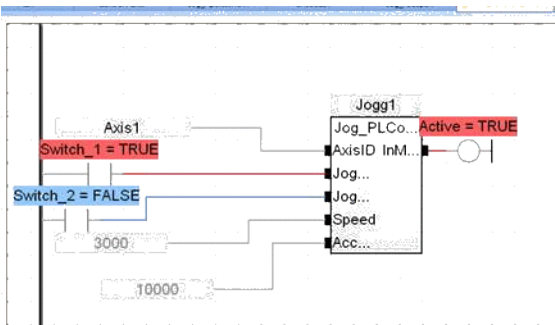
##### 5.2.8.8.1 Structured Text

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

### 5.2.8.9.2 Ladder Diagram



### 5.2.8.10.3 Function Block Diagram



### 5.2.9 MCFB\_GearedWebTension PLCopen ✔

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

The following figure shows the function block I/O.

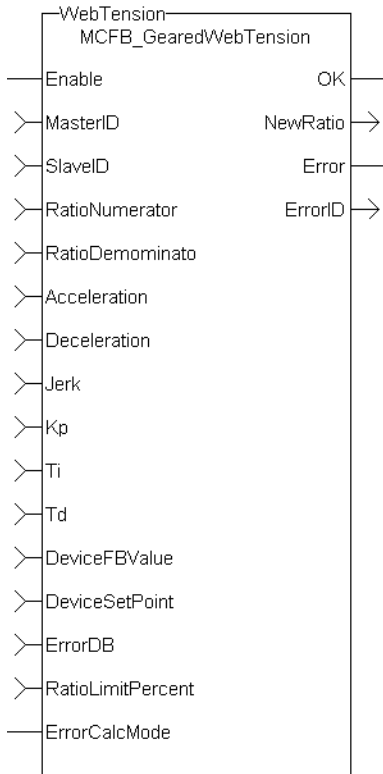


Figure 1-155: MCFB\_GearedWebTension Function Block I/O

#### 5.2.9.1 Arguments

##### 5.2.9.2.1 Inputs

<b>Enable</b>	<b>Description</b>	Enables execution
	<b>Data Type</b>	BOOL
	<b>Range</b>	[0,1]
	<b>Unit</b>	n/a
	<b>Default</b>	-
<b>MasterID</b>	<b>Description</b>	Identifies the master axis
	<b>Data Type</b>	AXIS_REF
	<b>Range</b>	



	<b>Unit</b>	n/a
	<b>Default</b>	-
<b>SlaveID</b>	<b>Description</b>	Identifies the slave axis
	<b>Data Type</b>	AXIS_REF
	<b>Range</b>	
	<b>Unit</b>	n/a
	<b>Default</b>	-
<b>RatioNumerator</b>	<b>Description</b>	Numerator of the master/slave ratio
	<b>Data Type</b>	DINT
	<b>Range</b>	[-2147483648 to +2147483647]
	<b>Unit</b>	n/a
	<b>Default</b>	-
<b>RatioDenominator</b>	<b>Description</b>	Denominator of the master/slave ratio
	<b>Data Type</b>	DINT
	<b>Range</b>	[-2147483648 to +2147483647]
	<b>Unit</b>	n/a
	<b>Default</b>	-
<b>Acceleration</b>	<b>Description</b>	Trapezoidal: acceleration rate, S-Curve: maximum acceleration
	<b>Data Type</b>	LREAL
	<b>Range</b>	[-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)]
	<b>Unit</b>	n/a
	<b>Default</b>	-
<b>Deceleration</b>	<b>Description</b>	Trapezoidal: deceleration rate, S-Curve: not used
	<b>Data Type</b>	LREAL
	<b>Range</b>	[-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)]
	<b>Unit</b>	n/a
	<b>Default</b>	-
<b>Jerk</b>	<b>Description</b>	Trapezoidal: 0, S-Curve: constant jerk

	<b>Data Type</b>	LREAL
	<b>Range</b>	[-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)]
	<b>Unit</b>	n/a
	<b>Default</b>	-
<b>Kp</b>	<b>Description</b>	Proportional gain
	<b>Data Type</b>	LREAL
	<b>Range</b>	[-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)]
	<b>Unit</b>	n/a
	<b>Default</b>	-
<b>Ti</b>	<b>Description</b>	Integral gain
	<b>Data Type</b>	LREAL
	<b>Range</b>	[-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)]
	<b>Unit</b>	n/a
	<b>Default</b>	-
<b>Td</b>	<b>Description</b>	Derivative gain
	<b>Data Type</b>	LREAL
	<b>Range</b>	[-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)]
	<b>Unit</b>	n/a
	<b>Default</b>	-
<b>DeviceFBValue</b>	<b>Description</b>	Analog input
	<b>Data Type</b>	DINT
	<b>Range</b>	[-2147483648 to +2147483647]
	<b>Unit</b>	n/a
	<b>Default</b>	-
<b>DeviceSetPoint</b>	<b>Description</b>	Analog set point
	<b>Data Type</b>	DINT
	<b>Range</b>	[-2147483648 to +2147483647]

	<b>Unit</b>	n/a
	<b>Default</b>	-
<b>ErrorDB</b>	<b>Description</b>	Maximum or minimum error between DeviceFBValue and DeviceSetPoint before a change will take place.
	<b>Data Type</b>	LREAL
	<b>Range</b>	[-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)]
	<b>Unit</b>	n/a
	<b>Default</b>	-
<b>RatioLimitPercent</b>	<b>Description</b>	Maximum and minimum master/slave ratio window
	<b>Data Type</b>	LREAL
	<b>Range</b>	[-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)]
	<b>Unit</b>	n/a
	<b>Default</b>	-
<b>ErrorCalcMode</b>	<b>Description</b>	Not set: DeviceFBValue-DeviceSetPoint, Set: DeviceSetPoint-DeviceFBValue
	<b>Data Type</b>	BOOL
	<b>Range</b>	[0,1]
	<b>Unit</b>	n/a
	<b>Default</b>	-

#### 5.2.9.3.2 Output

<b>OK</b>	<b>Description</b>	The output will have power flow after the enable input has been energized.
	<b>Data Type</b>	BOOL
	<b>Range</b>	[0,1]
	<b>Unit</b>	n/a
<b>NewRatio</b>	<b>Description</b>	New master/slave ratio
	<b>Data Type</b>	REAL
	<b>Range</b>	[-3.4E38 to 3.4E38 and -3.4E-38 to 3.4E-38 (6 to 7 significant digits of accuracy)]
	<b>Unit</b>	n/a

<b>Error</b>	<b>Description</b>	Function block error
	<b>Data Type</b>	BOOL
	<b>Range</b>	[0,1]
	<b>Unit</b>	n/a
<b>ErrorID</b>	<b>Description</b>	Function block error value
	<b>Data Type</b>	INT
	<b>Range</b>	[-32768 to +32767]
	<b>Unit</b>	n/a

#### 5.2.9.4 Usage

This Kollmorgen UDFB is used in conjunction with the main ladder MC\_GearIn function and it is assumed that the master/slave move is active. Internal to the Kollmorgen UDFB is another call to the MC\_GearIn function therefore the MasterID, SlaveID, RatioNumerator, RatioDenominator, Acceleration, Deceleration, and Jerk inputs are the same values as the main ladder MC\_GearIn function input values, both with the Buffer input of 0. This assures that the initial starting master/slave ratio will transition to the new Kollmorgen UDFB ratio smoothly.

This Kollmorgen UDFB will change the master/slave ratio that was defined by the MC\_GearIn function based on the error between the analog input and the analog set-point. The magnitude of the ratio and the rate of the ratio change is defined by the Kp, Ti, Td PID gain values. The new ratio calculated is output at the NewRatio output.

The RatioLimitPercent input is the maximum and minimum theoretical new ratio that can be changed. This provides a +/- window limit around the running ratio to prevent unwanted motion in the event of a web break or analog feedback failure.

#### 5.2.9.5.1 Example 1

##### NOTE

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

RatioNumerator = 1

RatioDenominator = 2 Therefore the master/slave starting ratio is 0.5000000

ErrorCaclMode = 0

DeviceFBValue = 6

DeviceSetPoint = 4 Therefore error  $6 - 4 = 2$

Kp = 0.005

Ti = 0

Td = 0

From the equation:

**New RatioDenominator = (RatioDenominator - Kp \* error)**

Therefore the new RatioDenominator =  $(2 - 0.005*2) = 1.99$

Thus the new master/slave running ratio is  $1 / 1.99 = 0.502512562$

Since the master/slave ratio is greater than the previous ratio the slave axis is going faster and the tension is reduced.

#### 5.2.9.6.2 Example 2

**NOTE**

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

This is the same example as example 1 with the exception of the ErrorCaclMode input Boolean set.

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

From the equation:

**New RatioDemominator = (RatioDemoninator – (Kp \* error))**

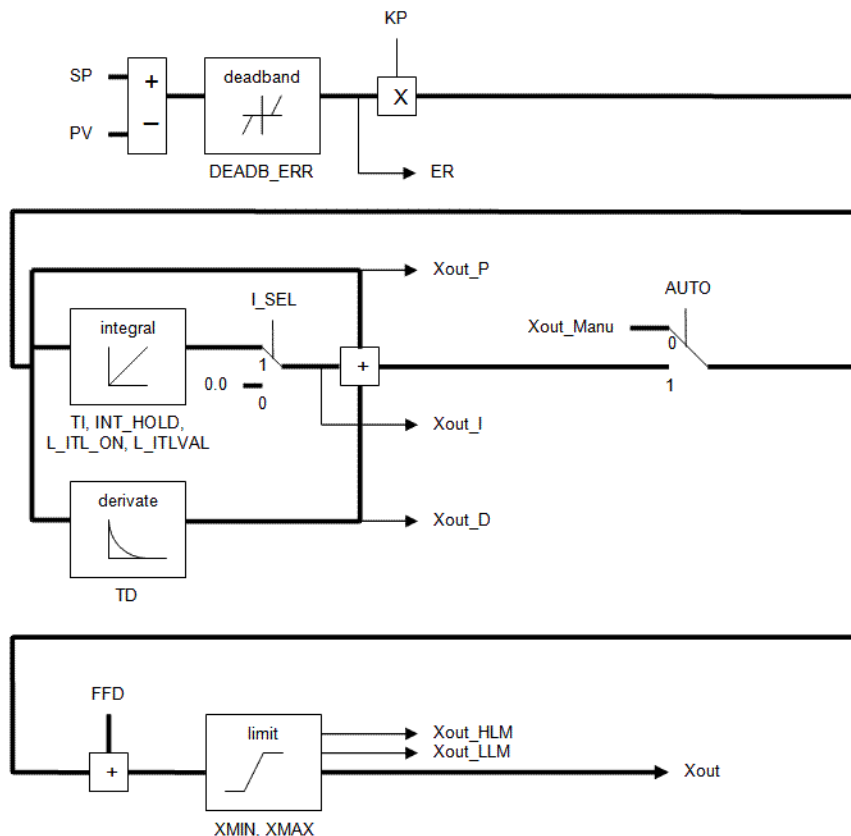
Therefore the new RatioDenominator = (2 + 0.005\*2) = 2.01

Thus the new master/slave running ratio is 1 / 2.01 = 0.497512437

Since the master/slave ratio is less than the previous ratio the slave axis is going slower and the tension is reduced.

**5.2.9.7.3 PID Function in KAS:**

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



**5.2.9.8.4 Programming tips:**

The First Order Digital Filter Kollmorgen UDFB can be used to decrease excess dither on the analog input. The filtered analog value is then used at the DeviceFBValue input of the MCFB\_GearedWebTension Kollmorgen UDFB .

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

#### 5.2.9.9.5.1 Example 1

No scaling

Initial MC\_GearIn input RatioNumerator = 2

Initial MC\_GearIn input RatioDenominator = 1 then initial Master/Slave ratio = 2

Kollmorgen UDFB input RatioNumerator = 2

Kollmorgen UDFB input RatioDenominator = 1 then Kollmorgen UDFB Master/Slave ratio = 2

Kollmorgen UDFB input DeviceFBValue = 4

Kollmorgen UDFB input DeviceFBSetpoint = 3 then Device PID error = 1 assume  $K_P = 1$ ,  $T_i$  and  $T_d = 0$

New Kollmorgen UDFB RatioNumerator = Current RatioNumerator – PID error =  $2 - 1 = 1$  then new

Kollmorgen UDFB Master/Slave ratio = 1

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

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

#### 5.2.9.10.6.2 Example 2

Scaling value = 1000

Initial MC\_GearIn input RatioNumerator = 2

Initial MC\_GearIn input RatioDenominator = 1 then initial Master/Slave ratio = 2

Kollmorgen UDFB input RatioNumerator = 2000

Kollmorgen UDFB input RatioDenominator = 1000 then Kollmorgen UDFB Master/Slave ratio = 2

Kollmorgen UDFB input DeviceFBValue = 4

Kollmorgen UDFB input DeviceFBSetpoint = 3 then Device PID error = 1 assume  $K_P = 1$ ,  $T_i$  and  $T_d = 0$

New Kollmorgen UDFB RatioNumerator = Current RatioNumerator – PID error =  $2000 - 1 = 1999$  then new

Kollmorgen UDFB Master/Slave ratio = 1999

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

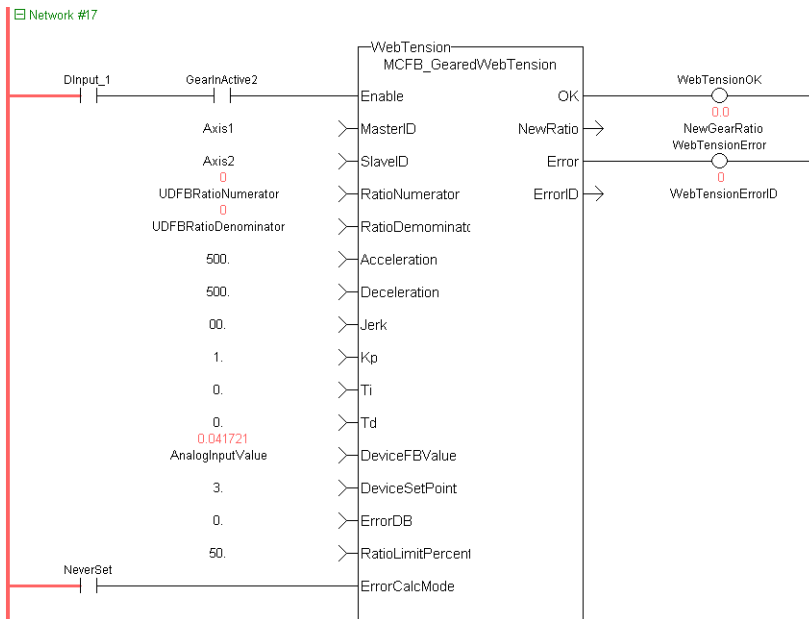
This resolution is much finer than example 1 so for a change of 1 for the error output of the PID this creates a Master/Slave ratio change of 1999. This results in a slower rate of change to the slave velocity that is more suited to good tension in a machine process.

#### 5.2.9.11 Related Functions

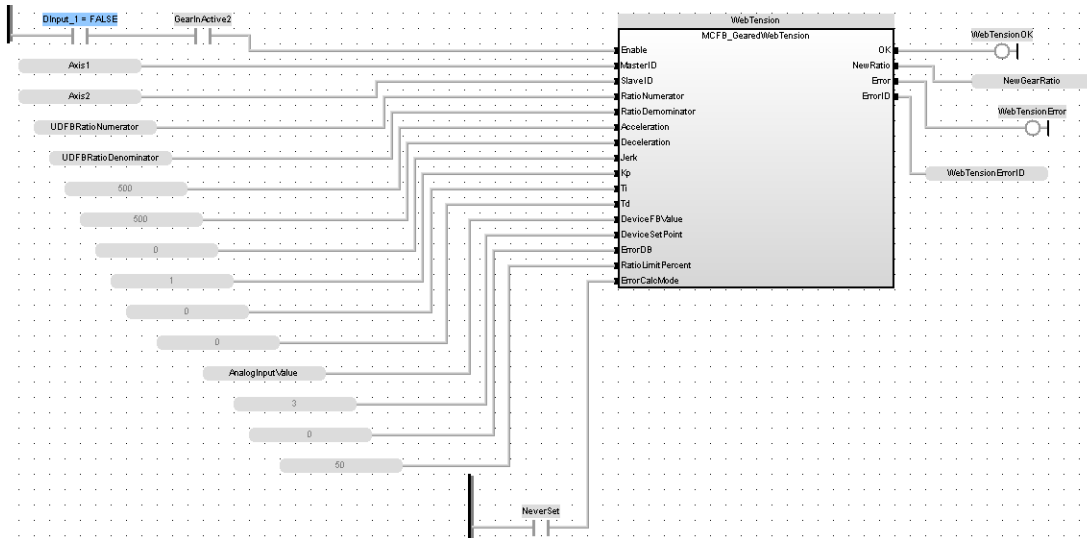
["FB\\_FirstOrderDigitalFilter" \(p. 880\)](#)

#### 5.2.9.12 Example

##### 5.2.9.13.1 Ladder Example



### 5.2.9.14.2 Function Block Diagram Example



### 5.2.9.15.3 Structured Text Example

```

Inst_MCFB_GearedWebTension
( DInput_1 FALSE, Axis1, Axis2, UDFBRatioNumerator 0, UDFBRatioDenominator 0,
500.0, 500.0, 0.0, 1.0, 0.0, 0.0, AnalogInputValue 0.0, 3.0, 0.0, 50.0, NeverSet FALSE );
WebTensionOk FALSE := Inst_MCFB_GearedWebTension.OK FALSE;
NewGearRatio 0.0 := Inst_MCFB_GearedWebTension.NewRatio 0.0;
WebTensionError FALSE := Inst_MCFB_GearedWebTension.Error FALSE;
WebTensionErrorID 0 := Inst_MCFB_GearedWebTension.ErrorID 0;
    
```

## 5.2.10 FB\_FirstOrderDigitalFilter

### 5.2.10.1 Description

This FB is defined to filter an Analog signal.

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

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

The following figure shows the function block I/O

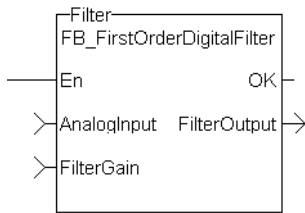


Figure 1-156: CBS First Order Digital Filter

### 5.2.10.2 Arguments

#### 5.2.10.3.1 Inputs

<b>EN</b>	<b>Description</b>	Enables execution (FFLD only )
	<b>Data type</b>	BOOL
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	
<b>AnalogInput</b>	<b>Description</b>	Analog Input from transducer
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	
<b>FilterGain</b>	<b>Description</b>	Filter Gain
	<b>Data type</b>	REAL
	<b>Range</b>	[1 - 0.05]
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 5.2.10.4.2 Outputs



<b>OK</b>	<b>Description</b>	Execution Complete
	<b>Data type</b>	BOOL
	<b>Range</b>	[0,1]
	<b>Unit</b>	
<b>FilterOutput</b>	<b>Description</b>	Filtered analog input value
	<b>Data type</b>	REAL
	<b>Range</b>	[0,1]
	<b>Unit</b>	

### 5.2.10.5 Usage

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

- The AnalogInput input is the unfiltered “raw” analog feedback signal from an LVDT, tension transducer, potentiometer, or some other like device.
- The FilterGain defines the amount of filtering to be used. The range of the gain is from 1.0 or no filtering to 0.05 or the maximum filtering.
- The FilterOutput is the filtered analog input and is typically used as an input to some other function block or UDFB that has an analog input, for example the MCFB\_GearedWebTension UDFB.
- The implementation of the digital first order filter is for PLCopen.
- The equation is defined as:  $Input * Gain + Output * (1 - Gain) = Output$
- The steady state filter delay with a gain of 0.8 can be seen in the following table.

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

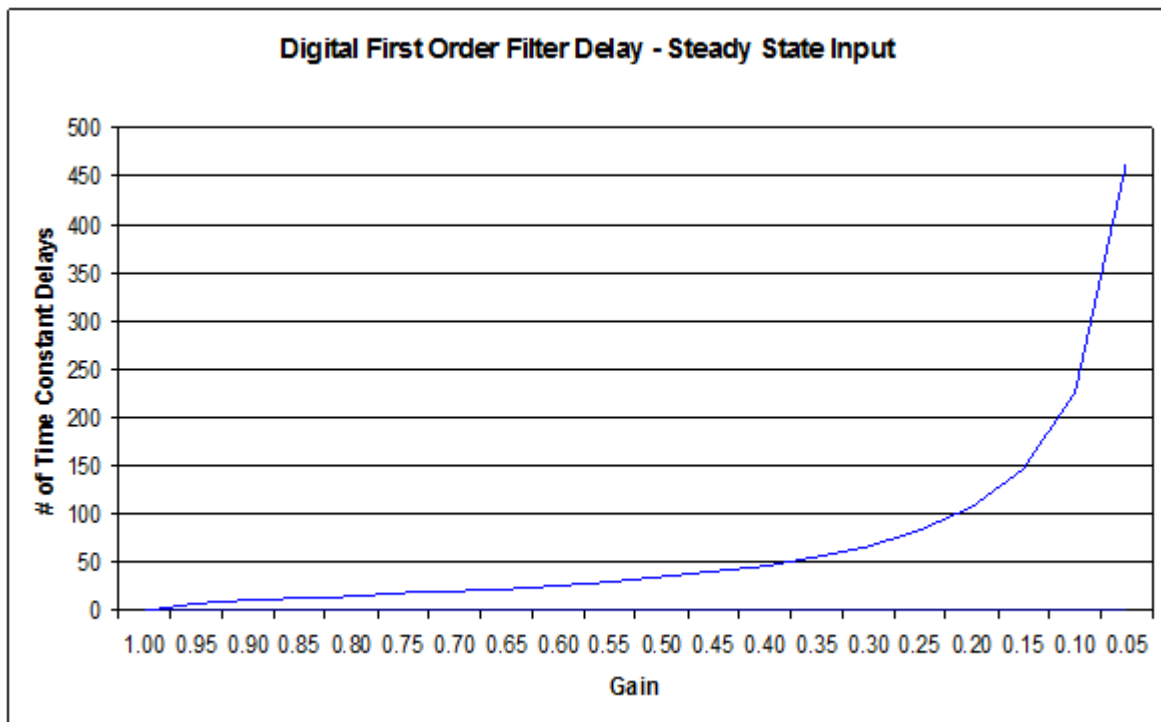
FilterGain	FilterInput	FilterOutput
100	100	100
100	100	100
100	100	100
100	100	100

Table 1-13: Filter Input Delay Example

The range of the filter gain is between 1.00 and 0.05. From the table, for a filter gain of 0.8 there is a delay of 15 time constants with a time constant defined as the rate the UDFB is scanned or executed in the application. For example if the UDFB was executed every millisecond a gain of 0.8 would provide a filter delay of 15ms. Conversely a gain of 1.00 provides zero filtering and the output signal follows the input signal, and a gain of 0.05 provides the most filtering for 463 ms.

The numbers of filter delays for a steady state analog input at a given gain are shown in the table and graph below.

Gain	Filter Delay Tn
1.00	0
0.95	8
.90	11
.85	13
.80	15
.75	18
.70	20
.65	23
.60	26
.55	30
.50	35
.45	40
.40	47
.35	56
.30	66
.25	83
.20	107
.15	146
.10	226
.05	463



Of course a real world analog input is most always a varying feedback signal. In Table 2.3 this is shown with an initial input of 100, a gain of 0.8, and a random variability of 10%. Filter Input

Filter Input	Filter Current Output	Amount of Input Filtering	Random Filter % Variation
0	0	0	10%
100	80	-20	
97.38903813	93.9112305	-3.477807626	
92.67638093	92.92335084	0.246969915	
94.12988912	93.88858146	-0.241307655	
103.0835564	101.2445614	-1.838994993	
91.16845433	93.18367575	2.015221422	
93.23936976	93.22823096	-0.011138803	
94.90272089	94.56782291	-0.334897986	
103.3070737	101.5592235	-1.747850153	
96.83149418	97.77704005	0.945545867	
96.35024002	96.63560002	0.285360007	
99.82417525	99.1864602	-0.637715045	
105.0792636	103.9007029	-1.178560685	
97.36988208	98.67604626	1.306164172	
107.82502	105.9952253	-1.829794752	
97.7886524	99.42996698	1.641314572	
108.2038024	106.4490353	-1.754767081	
91.58527607	94.55802792	2.972751845	
93.6783421	93.85427926	0.175937164	
102.8695349	101.0664838	-1.803051129	
93.95916817	95.3806313	1.421463121	

Filter Input	Filter Current Output	Amount of Input Filtering	Random Filter % Variation
108.6579707	106.0025028	-2.655467871	
109.3425748	108.6745604	-0.668014397	
103.9066	104.8601921	0.953592077	
92.30112142	94.81293555	2.511814127	
109.4460726	106.5194452	-2.926627416	
94.88799896	97.21428821	2.326289251	
105.4738635	103.8219484	-1.651915057	
102.988167	103.1549233	0.166756284	
92.92925408	94.97438792	2.045133846	
95.58185568	95.46036213	-0.121493552	
109.414248	106.6234708	-2.790777178	
106.5661311	106.577599	0.011467953	
99.85857253	101.2023778	1.343805301	
107.865421	106.5328124	-1.332608643	
92.19683177	95.0640279	2.867196126	
104.8558146	102.8974573	-1.958357346	
104.5140236	104.1907104	-0.323313268	
104.3675014	104.3321432	-0.035358206	
109.2704266	108.2827699	-0.987656683	
101.4962729	102.8535723	1.35729941	
92.19199163	94.32430776	2.132316128	
99.13065312	98.16938405	-0.961269073	
103.5068114	102.4393259	-1.067485466	
109.502983	108.0902516	-1.412731426	
99.05504822	100.8620889	1.80704068	
94.97711299	96.15410817	1.176995182	
107.1063597	104.9159094	-2.190450308	
91.12245188	93.88114339	2.758691504	
108.130314	105.2804799	-2.849834129	
104.2923832	104.4900025	0.197619344	
101.3775072	102.0000062	0.62249907	
<b>100.5303014</b>	<b>100.0399168</b>	<b>-0.490384645</b>	<b>Averages</b>

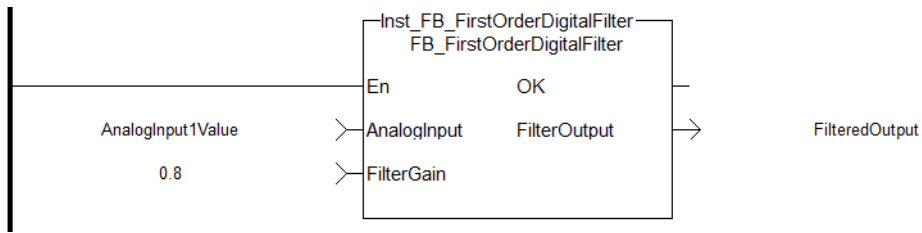
Table 1-14: Filter Input Lag Example - Random Input

### 5.2.10.6 Example

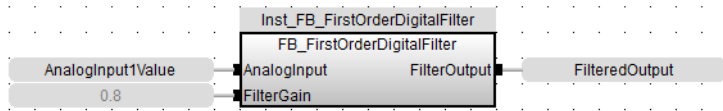
#### 5.2.10.7.1 Structured Text

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

#### 5.2.10.8.2 Ladder Diagram



### 5.2.10.9.3 Function Block Diagram



## 5.2.11 MCFB\_AKDFault PLCopen ✓

### 5.2.11.1 Description

Outputs AKD drive fault Information. The FAULT output turns TRUE when the selected drive goes into a fault state. The fault number is the same number as reported on the display of the AKD drive. This function can be used with the PLCopen Motion engines. The following figure shows the function block I/O:

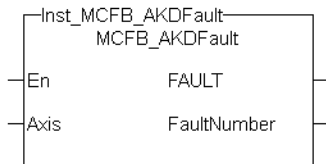


Figure 1-157: MCFB\_AKDFault

### 5.2.11.2 Arguments

#### 5.2.11.3.1 Input

<b>EN</b>	<b>Description</b>	ENABLES the Kollmorgen UDFB (used in FFLD editor only)
	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>Axis</b>	<b>Description</b>	Name of a declared instance of the AXIS_REF library function. For more details, <a href="#">About Axis Name and Number</a> .
	<b>Data type</b>	<a href="#">AXIS_REF</a>
	<b>Range</b>	[1, 256]
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 5.2.11.4.2 Output

<b>FAULT</b>	<b>Description</b>	TRUE if selected drive currently has a Fault
	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
<b>FaultNumber</b>	<b>Description</b>	Three digit AKD Fault identifier
	<b>Data type</b>	DINT
	<b>Range</b>	[100 , 999]
	<b>Unit</b>	n/a

### 5.2.11.5 Usage

Typical usage for this UDFB are:

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

Related Functions

["MCFB\\_AKDFaultLookup" \(p. 888\)](#)

["FB\\_AKDFltRpt" \(p. 893\)](#)

["MC\\_ReadStatus" \(p. 437\)](#) (PLCopen Motion Engine)

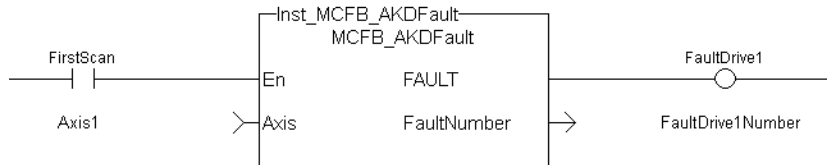
### 5.2.11.6 Example

#### 5.2.11.7.1 Structured Text

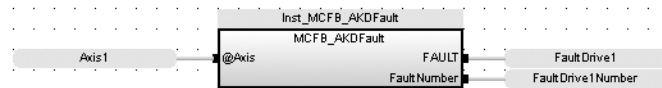
```
//Execute and Read the Function Block
Inst_MCFB_AKDFault( Axis1(*lib:AXIS_REF*) );
FaultDrive1 := Inst_MCFB_AKDFault.FAULT;
FaultDrive1Number := Inst_MCFB_AKDFault.FaultNumber;

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

#### 5.2.11.8.2 Ladder Diagram



#### 5.2.11.9.3 Function Block Diagram



## 5.2.12 MCFB\_AKDFaultLookup PLCopen

### 5.2.12.1 Description

String message of the corresponding AKD drive fault number. The OK output turns TRUE when there is a match for the FaultNumber. The FaultDescription displays the corresponding text string. The FaultNumber is the same number as reported on the display of the AKD drive. This function can be used with the PLCopen Motion engines. The following figure shows the function I/O:

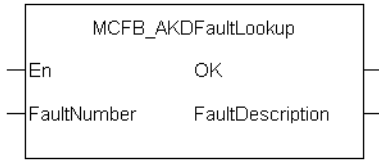


Figure 1-158: MCFB\_AKDFaultLookup

### 5.2.12.2 Arguments

#### 5.2.12.3.1 Input

<b>EN</b>	<b>Description</b>	ENABLES the Kollmorgen UDFB (used in FFLD editor only)
	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>FaultNumber</b>	<b>Description</b>	The AKD drive fault number
	<b>Data type</b>	DINT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 5.2.12.4.2 Output

<b>OK</b>	<b>Description</b>	TRUE if there is a match for the FaultNumber
	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
<b>FaultDescription</b>	<b>Description</b>	Description of the Fault
	<b>Data type</b>	STRING
	<b>Range</b>	n/a
	<b>Unit</b>	n/a



### 5.2.12.5 Usage

Typical usage for this UDFB are:

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

### 5.2.12.6 Related Functions

"MCFB\_AKDFault" (p. 886)

"FB\_AKDFitRpt" (p. 893)

"MC\_ReadStatus" (p. 437) (PLCopen Motion Engine)

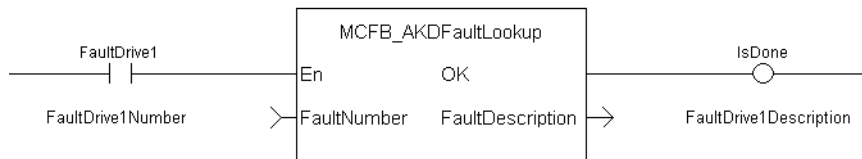
### 5.2.12.7 Example

#### 5.2.12.8.1 Structured Text

```
//Execute and Read the Function Block
Inst_MCFB_AKDFault( Axis1(*lib:AXIS_REF*) );
FaultDrive1 := Inst_MCFB_AKDFault.FAULT;
FaultDrive1Number := Inst_MCFB_AKDFault.FaultNumber;

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

#### 5.2.12.9.2 Ladder Diagram



#### 5.2.12.10.3 Function Block Diagram



## 5.2.13 FB\_Cylinder

### 5.2.13.1 Description

This function block can be used to control a cylinder and the Limit Switches.

There are two inputs InA and InB to set the direction of the movement and the belonging LimSwitches LsA and LsB.

If InA is set to TRUE the output DirA is set to TRUE and after a time value defined by CtrlTime the LsA has to become TRUE otherwise a fault FaultLsA appears. Just as in direction B.

If both LsA and LsB are TRUE then a Fault depending of the output is set. If both InA and InB are given (e.g. to stop the cylinder movement) no limit switch is controlled.

All faults can be reset by input iResetFault.

### 5.2.13.2 Arguments

#### 5.2.13.3.1 Input

<b>iInA</b>	<b>Description</b>	Set direction A
	<b>Data type</b>	BOOL
<b>iInB</b>	<b>Description</b>	Set direction B
	<b>Data type</b>	BOOL
<b>iLsA</b>	<b>Description</b>	Limit Switch at End of direction A
	<b>Data type</b>	BOOL
<b>iLsB</b>	<b>Description</b>	Limit Switch at End of direction B
	<b>Data type</b>	BOOL
<b>iCtrlTime</b>	<b>Description</b>	Max Time till Lim.Sw. has to be reached
	<b>Data type</b>	TIME
<b>iResetFault</b>	<b>Description</b>	Reset Fault (Is set to FALSE by UDFB!)
	<b>Data type</b>	BOOL

#### 5.2.13.4.2 Output

<b>oDirA</b>	<b>Description</b>	Direction A
	<b>Data type</b>	BOOL
<b>oDirB</b>	<b>Description</b>	Direction B
	<b>Data type</b>	BOOL
<b>oFaultLsA</b>	<b>Description</b>	Fault of Lim.Sw. at End direction A
	<b>Data type</b>	BOOL
<b>oFaultLsB</b>	<b>Description</b>	Fault of Lim.Sw. at End direction B

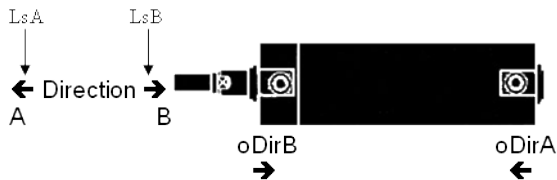
**Data type**      BOOL

### 5.2.13.5 Usage

The signal flow is valid for both directions (A and B)

If oDirA AND oDirB are active there is no Fault Control.

The Fault can be reset by iRestFault = True.

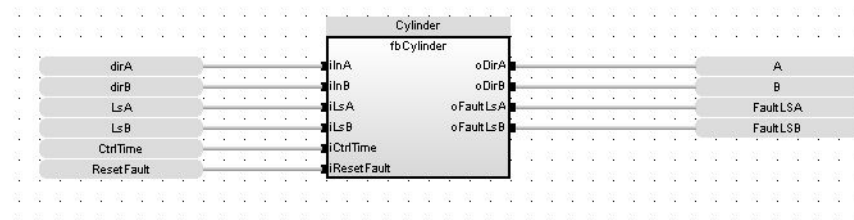


### 5.2.13.6 Example

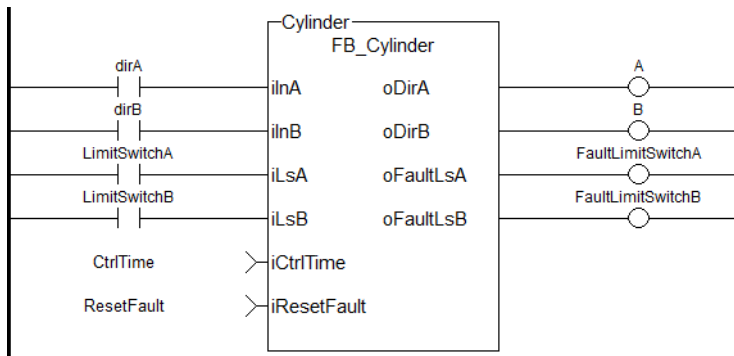
#### 5.2.13.7.1 ST

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

### 5.2.13.8.2 Function Block Diagram



### 5.2.13.9.3 FFLD



**5.2.13.10 FB\_AKDFitRpt** PLCopen ✓ Pipe Network ✓

**5.2.13.11.1 Description**

Outputs AKD drive fault Information.

The oFAULT output turns TRUE when the selected drive goes into a fault state. This function block outputs the total number of faults in the AKD drive fault history variable (Pre-Defined Error Field Object 1003h), and the fault number and message for the last 3 drive faults.

Each fault has two outputs: the fault number and a fault message. The fault number is the same number as reported on the display of the AKD drive. The fault message provides a short description of the fault. For example if the first fault is a feedback error with a F401 displayed on the front of the drive, the output of this FB are:

- **oFirstFaultNumber** = 401
- **oFirstFaultMessage** = Failed To Set Feedback Type

The **iResetfaultHistory** Input resets the faults reported by the FB.

The **oDriveNotUsed** outputs a 1 (True) if the axis is configured to Simulated in the ProjectEthercat setup screen.

This function Block can be used with either the PipeNetwork or PLCopen Motion engines. The following figure shows the function block I/O:

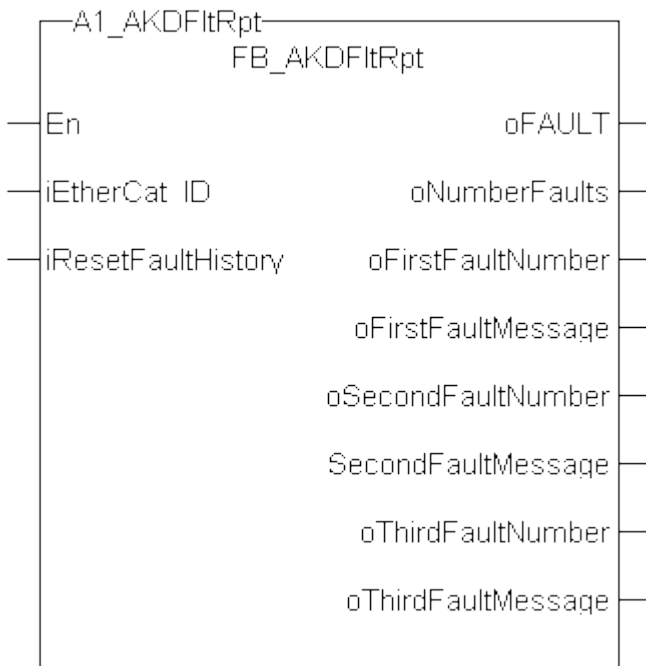


Figure 1-159: AKDFitRpt

**5.2.13.12.2 Arguments**

**5.2.13.13.3.1 Input**

<b>EN</b>	<b>Description</b>	ENABLES the Kollmorgen UDFB (used in FFLD editor only)
	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a

	<b>Default</b>	—
<b>iEtherCat_ID</b>	<b>Description</b>	EtherCAT address desired AKD Drive ex. 1001 or AKD_1
	<b>Data type</b>	INT
	<b>Range</b>	—
	<b>Unit</b>	n/a
	<b>Default</b>	—
<b>iRstFitHist</b>	<b>Description</b>	When input is TRUE, clears all Faults saved to drives history
	<b>Data type</b>	BOOL
	<b>Range</b>	[0 , 1]
	<b>Unit</b>	n/a
	<b>Default</b>	—

#### 5.2.13.14.4.2 Output

<b>oFAULT</b>	<b>Description</b>	TRUE if selected drive currently has a Fault
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a
<b>oNumberFaults</b>	<b>Description</b>	Number of faults saved in the Drive's history
	<b>Data type</b>	DINT
	<b>Range</b>	[0 , 10]
	<b>Unit</b>	n/a
<b>oFirstFaultNumber</b>	<b>Description</b>	Three digit AKD Fault identifier
	<b>Data type</b>	DINT
	<b>Range</b>	[100 , 999]
	<b>Unit</b>	n/a
<b>oFirstFaultMessage</b>	<b>Description</b>	Description of the Fault
	<b>Data type</b>	STRING
	<b>Unit</b>	n/a
<b>oSecondFaultNumber</b>	<b>Description</b>	Three digit AKD Fault identifier.
	<b>Data type</b>	DINT
	<b>Range</b>	[100 , 999]

	<b>Unit</b>	n/a
<b>oSecondFaultMessage</b>	<b>Description</b>	Description of the Fault
	<b>Data type</b>	STRING
	<b>Unit</b>	n/a
<b>oThirdFaultNumber</b>	<b>Description</b>	Three digit AKD Fault identifier
	<b>Data type</b>	DINT
	<b>Range</b>	[100 , 999]
	<b>Unit</b>	n/a
<b>oThirdFaultMessage</b>	<b>Description</b>	Description of the Fault
	<b>Data type</b>	STRING
	<b>Unit</b>	n/a
<b>oDriveNotUsed</b>	<b>Description</b>	Is this Drive Real (0) on Simulated (1)
	<b>Data type</b>	BOOL
	<b>Unit</b>	n/a

#### 5.2.13.15.5 Usage

Typical usage for this UDFB are:

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

#### 5.2.13.16.6 Related Functions

[MC\\_ReadStatus](#) (PLCopen Motion Engine)

[MLAxisStatus](#) (Pipe Network Motion Engine)

#### 5.2.13.17.7 Example

#### 5.2.13.18.8.1 Structured Text

```

//Execute the Function Block
A1_AKDFltRpt (1001, resetFaultHistST);

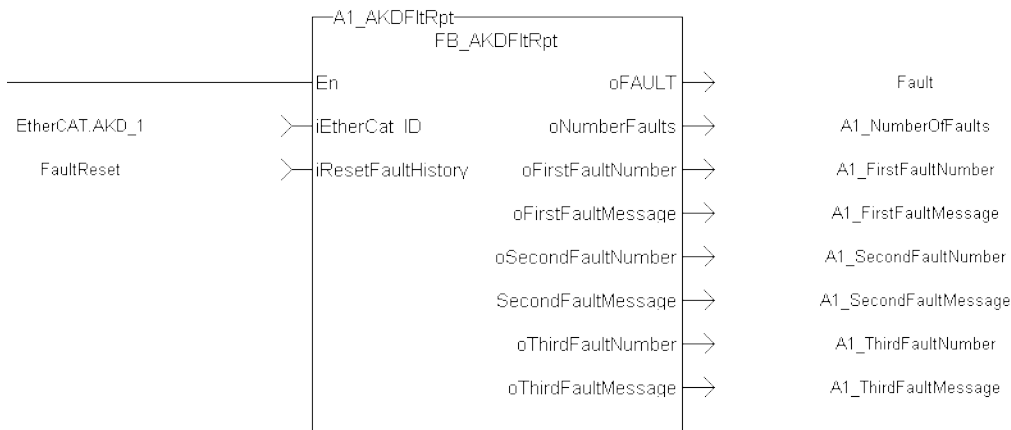
//Read Function Block Outputs
AKD1_Fault:= A1_AKDFltRpt.oFault;
AKD1_NumFault:= A1_AKDFltRpt.oNumberFaults;
AKD1_FirstFaultNumber:= A1_AKDFltRpt.oFirstFaultNumber;
AKD1_FirstFaultMessage:= A1_AKDFltRpt.oFirstFaultMessage;
AKD1_SecondFaultNumber:= A1_AKDFltRpt.oSecondFaultNumber;
AKD1_SecondFaultMessage:=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

;
    
```

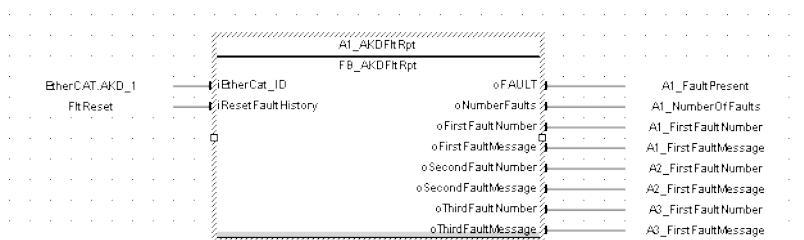
**NOTE**

A1\_FaultReporting is an instance of the FB\_S700FItRpt function block.

**5.2.13.19.9.2 Ladder Diagram**



**5.2.13.20.10.3 Function Block Diagram**





### 5.2.13.21 FB\_S700FltRpt PLCopen ✓ Pipe Network ✓

#### 5.2.13.22.1 Description

Outputs S700 drive fault Information.

The oFAULT output turns TRUE when the selected drive goes into a fault state. This function block outputs the total number of faults in the S700 drive fault history variable (FLTHIST), and the fault number and message for the last 3 drive faults.

Each fault has two outputs: the fault number and a fault message. The fault number is the same number as reported on the display of the S700 drive. The fault message provides a short description of the fault. For example if the first fault is a feedback error with a F04 is displayed on the front of the drive, the output of this FB are:

- **oFirstFaultNumber** = 04
- **oFirstFaultMessage** = Feedback Error

The **iResetfaultHistory** Input resets the faults reported by the FB.

The **oDriveNotUsed** outputs a 1 (True) if the axis is configured to Simulated in the ProjectEthercat setup screen.

This function Block can be used with either the PipeNetwork or PLCopen Motion engines.

The following figure shows the function block I/O:

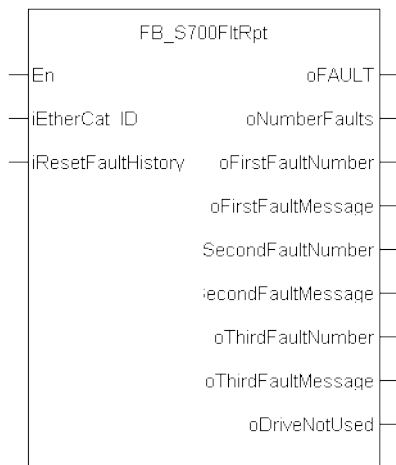


Figure 1-160: S700FltRpt

#### 5.2.13.23.2 Arguments

##### 5.2.13.24.3.1 Input

EN	Description	ENABLES the Kollmorgen UDFB (used in FFLD editor only)
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
iEtherCat_ID	Description	EtherCAT address desired AKD Drive ex. 1001 or AKD_1
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—

iRstFltHist	Description	When input is TRUE, clears all Faults saved to drives history
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—

#### 5.2.13.25.4.2 Output

oFAULT	Description	TRUE if selected drive currently has a Fault
	Data type	BOOL
	Unit	n/a
oNumberFaults	Description	Number of faults saved in the Drive's history
	Data type	DINT
	Range	[0 , 10]
	Unit	n/a
oFirstFaultNumber	Description	Two digit S700 drive Fault identifier
	Data type	DINT
	Range	[00 , 32]
	Unit	n/a
oFirstFaultMessage	Description	Description of the Fault
	Data type	STRING
	Unit	n/a
oSecondFaultNumber	Description	Two digit S700 drive Fault identifier
	Data type	DINT
	Range	[00 , 32]
	Unit	n/a
oSecondFaultMessage	Description	Description of the Fault
	Data type	STRING
	Unit	n/a
oThirdFaultNumber	Description	Two digit S700 drive Fault identifier
	Data type	DINT
	Range	[00 , 32]
	Unit	n/a
oThirdFaultMessage	Description	Description of the Fault
	Data type	STRING
	Unit	n/a
oDriveNotUsed	Description	Is this Drive Real (0) on Simulated (1)
	Data type	BOOL
	Unit	n/a

#### 5.2.13.26.5 Usage

Typical usage for this UDFB are:

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

#### 5.2.13.27.6 Related Functions

[MC\\_ReadStatus](#) (PLCopen Motion Engine)

[MLAxisStatus](#) (Pipe Network Motion Engine)

### 5.2.13.28.7 Example

#### 5.2.13.29.8.1 Structured Text

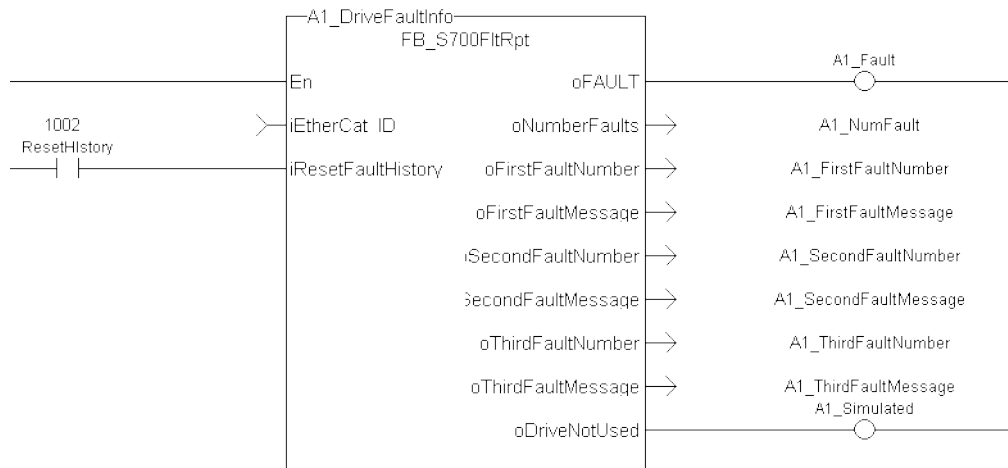
```
//Execute the Function Block
A1_FaultReporting (1001, 0);

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

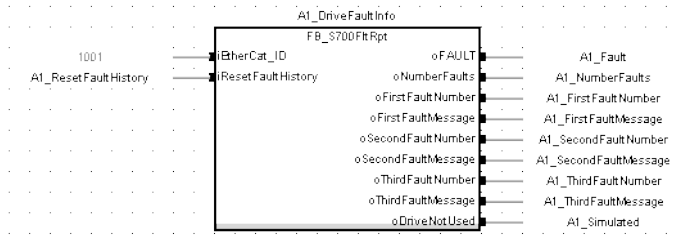
#### NOTE

A1\_FaultReporting is an instance of the FB\_S700FitRpt function block.

#### 5.2.13.30.9.2 Ladder Diagram



#### 5.2.13.31.10.3 Function Block Diagram



### 5.2.13.32 FB\_AxisPLsPosModulo

#### 5.2.13.33.1 Description

This function block can be used for any position of a modulo axis in both directions. The Boolean output oPLS is set to TRUE if the position has crossed the start position and is set to FALSE if the position has crossed the end position. The function block is executed cyclically. The function block has the possibility to compensate a delay time of the connected device, e.g. glue nozzles. It is also possible to define a hysteresis for switching on and off of the PLS.

#### 5.2.13.34.2 Arguments

##### 5.2.13.35.3.1 Input

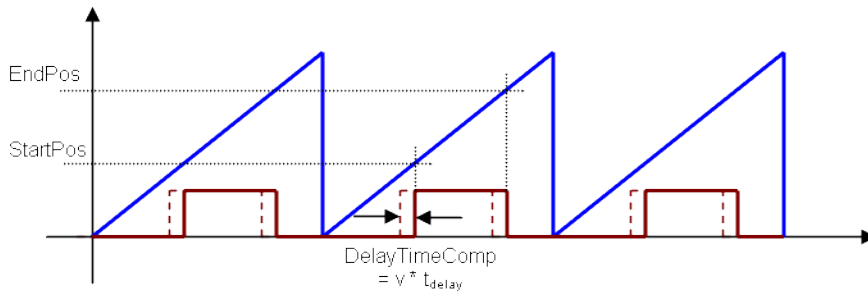
<b>ibExecute</b>	<b>Description</b>	Enable PLS
	<b>Data type</b>	BOOL
<b>iPosition</b>	<b>Description</b>	Any position of a modulo axis
	<b>Data type</b>	LREAL
<b>iModuloPosition</b>	<b>Description</b>	Modulo position of axis
	<b>Data type</b>	LREAL
<b>iStartPos</b>	<b>Description</b>	Start position of PLS
	<b>Data type</b>	LREAL
<b>iEndPos</b>	<b>Description</b>	End position of PLS
	<b>Data type</b>	LREAL
<b>iDelayTime</b>	<b>Description</b>	Delay time for compensation
	<b>Data type</b>	TIME
<b>iHysteresis</b>	<b>Description</b>	Hysteresis
	<b>Data type</b>	LREAL
<b>ibForce</b>	<b>Description</b>	Force PLS
	<b>Data type</b>	BOOL

##### 5.2.13.36.4.2 Output

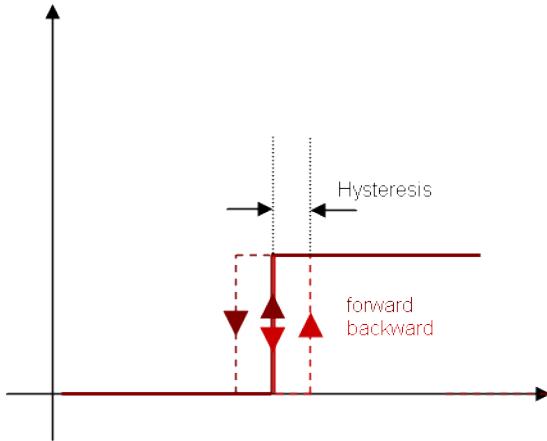
<b>oPLS</b>	<b>Description</b>	Position limit switch
	<b>Data type</b>	BOOL

##### 5.2.13.37.5 Example

##### 5.2.13.38.6 Timing



### 5.2.13.39.7 Hysteresis



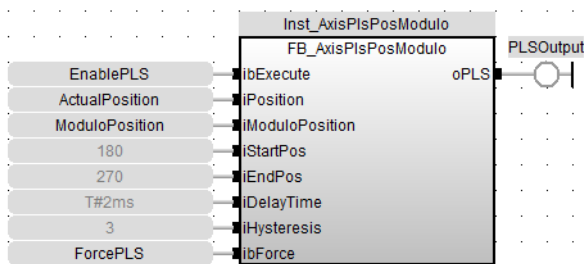
### 5.2.13.40.8.1 ST

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

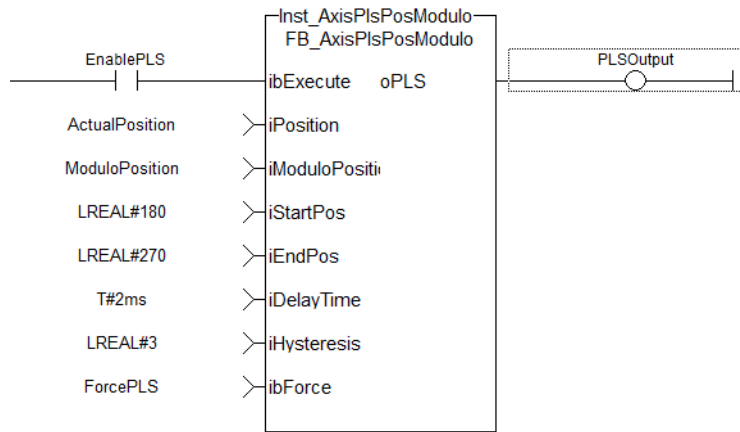
```
Inst_FB_AxisPlsPosModulo( EnablePLS, ActualPosition,
ModuloPosition, 180, 270, T#2ms, 3, ForcePLS );
```

```
PLSOutput := Inst_FB_AxisPlsPosModulo.oPLS;
```

### 5.2.13.41.9.2 Function Block Diagram



### 5.2.13.42.10.3 FFLD



**5.2.13.43 FB\_AxisPLsPosNoModulo**

PLCopen ✓

Pipe Network ✓

**5.2.13.44.1 Description**

This function block can be used for any position of a none-modulo axis in both directions. The Boolean output oPLS is set to TRUE if the position has crossed the start position and is set to FALSE if the position has crossed the end position. The function block has the possibility to compensate a delay time of the connected device, e.g. glue nozzles. It is also possible to define a hysteresis for switching on and off of the PLS.

**5.2.13.45.2 Arguments****5.2.13.46.3.1 Input**

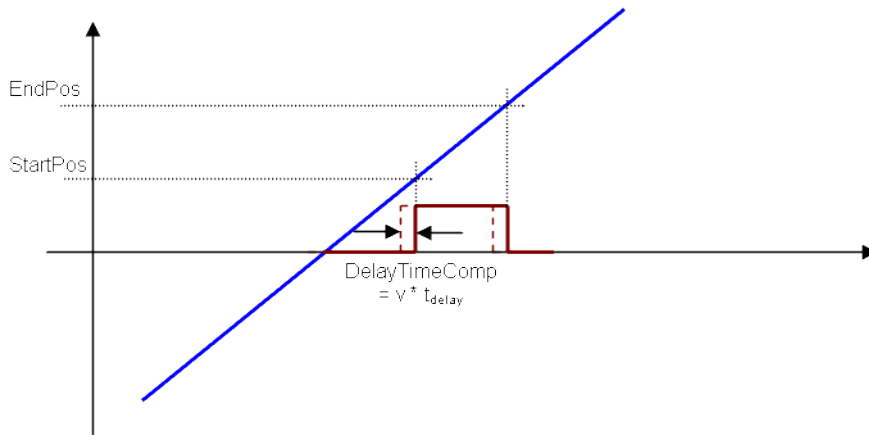
<b>ibExecute</b>	<b>Description</b>	Enable PLS
	<b>Data type</b>	BOOL
<b>iPosition</b>	<b>Description</b>	Any position of a none-modulo axis
	<b>Data type</b>	LREAL
<b>iStartPos</b>	<b>Description</b>	Start position of PLS
	<b>Data type</b>	LREAL
<b>iEndPos</b>	<b>Description</b>	End position of PLS
	<b>Data type</b>	LREAL
<b>iDelayTime</b>	<b>Description</b>	Delay time for compensation
	<b>Data type</b>	TIME
<b>iHysteresis</b>	<b>Description</b>	Hysteresis
	<b>Data type</b>	LREAL
<b>ibForce</b>	<b>Description</b>	Force PLS
	<b>Data type</b>	BOOL

**5.2.13.47.4.2 Output**

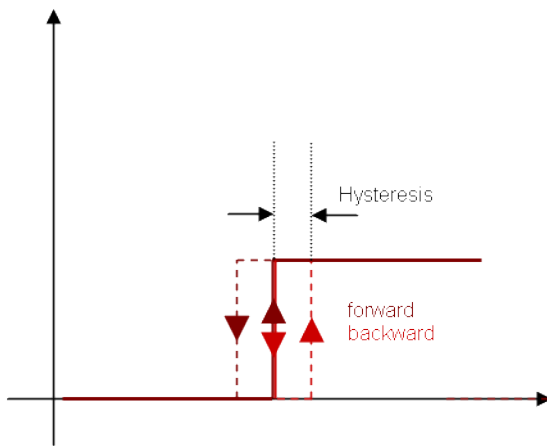
<b>oPLS</b>	<b>Description</b>	Position limit switch
	<b>Data type</b>	BOOL

**5.2.13.48.5 Example****5.2.13.49.6 Timing**





### 5.2.13.50.7 Hysteresis



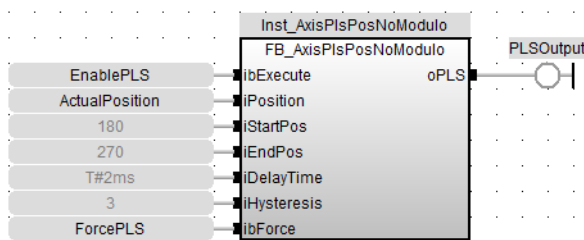
### 5.2.13.51.8.1 ST

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

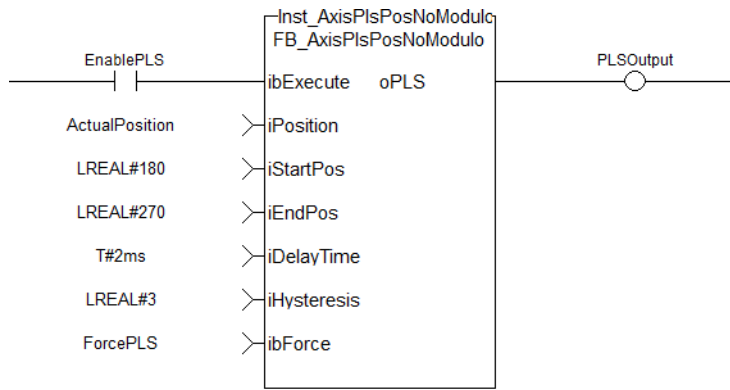
```
Inst_FB_AxisPlsPosNoModulo( EnablePLS, ActualPosition, 180, 270,
T#2ms, 3, ForcePLS );
```

```
PLSOutput := Inst_FB_AxisPlsPosNoModulo.oPLS;
```

### 5.2.13.52.9.2 FBD



### 5.2.13.53.10.3 FFLD



## 6 Index

### A

<b>Abort Camming</b> .....	<b>479</b>
<b>Abort Gearing</b> .....	<b>500, 505</b>
<b>actual position</b>	
pipe network .....	204
<b>actual velocity</b> .....	<b>429</b>
<b>AKDFaultLookup</b>	
PLCopen .....	886, 888
<b>AKDFItRpt</b> .....	<b>893</b>
<b>ASCII</b> .....	<b>685</b>
<b>AxisPlsPosModulo</b> .....	<b>901</b>
<b>AxisPlsPosNoModulo</b> .....	<b>904</b>

### C

<b>Cam</b>	
Aborting .....	479
<b>Coordinated Motion</b>	
Group Control items .....	581
Info items .....	616
List of Function Blocks .....	580
Motion items .....	638
Reference items .....	680
<b>copyrights</b> .....	<b>2</b>
<b>CurrentPosition</b>	
Pipe Network .....	204
<b>curve</b>	
synchronizer .....	380
<b>cycle</b>	
missed .....	685

### D

<b>debug</b>	
EtherCAT .....	704
<b>delay compensation</b> .....	<b>388</b>
<b>disclaimer</b> .....	<b>2</b>
<b>DriveParamRead</b> .....	<b>686</b>
<b>DriveParamWrite</b> .....	<b>691</b>

### E

<b>ECATGetObjVal</b> .....	<b>704</b>
<b>ECATGetStatus</b> .....	<b>715</b>
<b>ECATReadData</b> .....	<b>704</b>
<b>ECATReadSdo</b> .....	<b>694</b>
<b>ECATSetControl</b> .....	<b>720</b>
<b>ECATWCStatus</b> .....	<b>721</b>
<b>ECATWriteData</b> .....	<b>706</b>
<b>ECATWriteSdo</b> .....	<b>699</b>
<b>ECERR_DEVICE_ERROR</b> .....	<b>709</b>
<b>ECERR_FEATURE_NOT_AVAILABLE</b> .....	<b>709</b>
<b>ECERR_INVALID_ARRAY_SIZE</b> .....	<b>709</b>

<b>EtherCAT</b>	
image .....	704, 706
timeout .....	688
<b>EtherCAT library</b> .....	<b>684</b>
<b>EtherCAT library function</b>	
DriveParamRead .....	686
DriveParamWrite .....	691
ECATGetObjVal .....	704
ECATGetStatus .....	715
ECATReadData .....	704
ECATReadSdo .....	694
ECATSetControl .....	720
ECATWCStatus .....	721
ECATWriteData .....	706
ECATWriteSdo .....	699

## F

falling edge .....	383
fast input .....	132, 389
fbCylinder .....	890
feed-forward	
torque- .....	244
feedback position .....	121, 204

## G

<b>Gear</b>	
Aborting .....	500, 505
generator position .....	204
GetCtrlPerf .....	733

## H

HomeFindHomeFastInput .....	790
HomeFindHomeFastInputModulo .....	797
HomeFindHomeInput .....	769
HomeFindHomeInputThenZeroAngle .....	772
HomeFindLimitFastInput .....	804
HomeFindLimitFastInputModulo .....	810
HomeFindLimitInput .....	775
HomeFindLimitInputThenZeroAngle .....	777
HomeFindZeroAngle .....	779
HomeMoveUntilPosErrExceeded .....	782
HomeMoveUntilPosErrExceededThenZeroAngle .....	785
HomeUsingCurrentPosition .....	788
homing .....	525

## I

image EtherCAT .....	704, 706
----------------------	----------

## J

<b>Jog</b>	
Pipe Network .....	815
PLCopen .....	869

**K**

<b>Kollmorgen UDFB</b> .....	<b>742, 749, 758, 880</b>
AKDFaultLookup .....	886, 888
AKDFitRpt .....	893
AxisPlsPosModulo .....	901
AxisPlsPosNoModulo .....	904
fbCylinder .....	890
HomeFindHomeFastInput .....	790
HomeFindHomeFastInputModulo .....	797
HomeFindHomeInput .....	769
HomeFindHomeInputThenZeroAngle .....	772
HomeFindLimitFastInput .....	804
HomeFindLimitFastInputModulo .....	810
HomeFindLimitInput .....	775
HomeFindLimitInputThenZeroAngle .....	777
HomeFindZeroAngle .....	779
HomeMoveUntilPosErrExceeded .....	782
HomeMoveUntilPosErrExceededThenZeroAngle .....	785
HomeUsingCurrentPosition .....	788
Jog .....	815, 869
PlsPosFw .....	818
PlsPosFwBw .....	820
PlsTimeFw .....	822
S700FitRpt .....	897
ScaleInput .....	752
ScaleOutput .....	755
StepAbsolute .....	824
StepAbsSwitch .....	827
StepAbsSwitchFastInput .....	855
StepBlock .....	835
StepLimitSwitch .....	841
StepLimitSwitchFastInput .....	863
StepRefPulse .....	848
TemperaturePID .....	765

**M**

<b>Mark-to-Machine</b> .....	<b>530</b>
<b>Mark-to-Mark</b> .....	<b>530</b>
<b>MC_AbortTrigger</b> .....	<b>420</b>
<b>MC_AddSuperAxis</b> .....	<b>545</b>
<b>MC_AxisSetDefaults</b> .....	<b>638</b>
<b>MC_CamIn</b> .....	<b>479</b>
<b>MC_CamOut</b> .....	<b>486</b>
<b>MC_CamResumePos</b> .....	<b>490</b>
<b>MC_CamStartPos</b> .....	<b>493</b>
<b>MC_CamTblSelect</b> .....	<b>497</b>
<b>MC_ClearFaults</b> .....	<b>399</b>
<b>MC_CreateAxesGrp</b> .....	<b>585</b>
<b>MC_CreateAxis</b> .....	<b>401</b>
<b>MC_ErrorDescription</b> .....	<b>413, 549</b>
<b>MC_EStop</b> .....	<b>405</b>
<b>MC_GearIn</b> .....	<b>500</b>
<b>MC_GearInPos</b> .....	<b>505</b>
<b>MC_GearOut</b> .....	<b>512</b>
<b>MC_GrpDisable</b> .....	<b>588</b>
<b>MC_GrpEnable</b> .....	<b>590</b>
<b>MC_GrpHalt</b> .....	<b>642</b>
<b>MC_GrpReadActAcc</b> .....	<b>616</b>

MC_GrpReadActPos .....	620
MC_GrpReadActVel .....	623
MC_GrpReadBoolPar .....	592
MC_GrpReadCmdPos .....	626
MC_GrpReadCmdVel .....	629
MC_GrpReadError .....	632
MC_GrpReadStatus .....	634
MC_GrpReset .....	597
MC_GrpSetOverride .....	645
MC_GrpSetPos .....	680
MC_GrpStop .....	599
MC_GrpWriteBoolPar .....	602
MC_Halt .....	444
MC_InitAxesGrp .....	608
MC_InitAxis .....	407
MC_MachRegist .....	530
MC_MarkRegist .....	537
MC_MoveAbsolute .....	448
MC_MoveAdditive .....	453
MC_MoveCircAbs .....	647
MC_MoveCircRel .....	655
MC_MoveDirAbs .....	662
MC_MoveDirRel .....	665
MC_MoveLinAbs .....	668
MC_MoveLinRel .....	674
MC_MoveRelative .....	458
MC_MoveSuperimp .....	463
MC_MoveVelocity .....	468
MC_Phasing .....	515
MC_Power .....	410
MC_ReadActPos .....	427
MC_ReadActVel .....	429
MC_ReadAxisErr .....	431
MC_ReadBoolPar .....	433
MC_ReadParam .....	435
MC_ReadStatus .....	437
MC_Reference .....	522
MC_RemSuperAxis .....	547
MC_ResetError .....	415
MC_SetOverride .....	477
MC_SetPos .....	527
MC_SetPosition .....	530
MC_Stop .....	417
MC_StopRegist .....	543
MC_SyncSlaves .....	519
MC_TouchProbe .....	422
MC_UngroupAllAxes .....	614
MC_WriteBoolPar .....	440
MC_WriteParam .....	442
missing PLC cycles .....	685
MLAxisAbs .....	122
MLAxisActualPos .....	175
MLAxisAdd .....	127
MLAxisClrErrors .....	191
MLAxisGenStatus .....	179
MLAxisPowerOff .....	168
MLAxisPowerOn .....	168
MLAxisRefPos .....	134
MLAxisRun .....	164
MLAxisSetZero .....	205

<b>MLCNVConECAT</b> .....	<b>247</b>
<b>MLPrfGetIRatio</b> .....	<b>216</b>
<b>MLPrfGetORatio</b> .....	<b>220</b>
<b>MLPrfSetIRatio</b> .....	<b>224</b>
<b>MLPrfSetORatio</b> .....	<b>228</b>
<b>MLProfileRelease</b> .....	<b>566</b>
<b>MLSmpConPLCAxis</b> .....	<b>366</b>
<b>MLSmpConPNAxis</b> .....	<b>368</b>
<b>MLTrigGetPos</b> .....	<b>390</b>
<b>MLTrigGetTime</b> .....	<b>392</b>
<b>motion library</b> .....	<b>80-81</b>
adder .....	100
Axis .....	121
Block .....	90
CAM Profile .....	209
Comparator .....	230
Convertor .....	242
Delay .....	253
Derivator .....	255
Gear .....	261
Integrator .....	281
Master .....	286
Phaser .....	319, 685, 694, 704, 708
Pipe .....	81
PMP .....	330
Sampler .....	359
State Machine .....	569
Synchronizer .....	370
Trigger .....	381
<b>motion library function</b> .....	
MC_AbortTrigger .....	420
MC_CamIn .....	479
MC_CamOut .....	486
MC_CamTbiSelect .....	497
MC_ClearFaults .....	399
MC_CreateAxis .....	401
MC_EStop .....	405
MC_GearIn .....	500
MC_GearInPos .....	505
MC_GearOut .....	512
MC_Halt .....	444
MC_InitAxis .....	407
MC_MachRegist .....	530
MC_MarkRegist .....	537
MC_MoveAbsolute .....	448
MC_MoveAdditive .....	453
MC_MoveRelative .....	458
MC_MoveSuperimp .....	463
MC_MoveVelocity .....	468
MC_Phasing .....	515
MC_Power .....	410
MC_ReadActPos .....	427
MC_ReadActVel .....	429
MC_ReadAxisErr .....	431
MC_ReadBoolPar .....	433
MC_ReadParam .....	435
MC_ReadStatus .....	437
MC_Reference .....	522
MC_ResetError .....	415
MC_SetOverride .....	477

MC_SetPos .....	527
MC_SetPosition .....	530
MC_Stop .....	417
MC_StopRegist .....	543
MC_SyncSlaves .....	519
MC_TouchProbe .....	422
MC_WriteBoolPar .....	440
MC_WriteParam .....	442
MLAxisAbs .....	122
MLAxisAdd .....	127
<b>P</b>	
<b>phasing</b>	
PLCopen .....	515
synchronizer pipe block .....	370
<b>pipe position</b> .....	<b>204</b>
<b>PlsPosFw</b> .....	<b>818</b>
<b>PlsPosFwBw</b> .....	<b>820</b>
<b>PlsTimeFw</b> .....	<b>822</b>
<b>position</b>	
actual position .....	204
feedback position .....	121, 204
generator position .....	204
pipe position .....	204
reference position .....	122, 205, 230
<b>Position</b>	
CurrentPosition .....	204
Power ON Delta Offset .....	205
Zero Offset .....	205
<b>Power ON Delta Offset</b>	
Pipe Network .....	205
<b>PrintMessage</b> .....	<b>725</b>
<b>R</b>	
<b>reference position</b> .....	<b>122, 205, 230</b>
<b>registration</b> .....	<b>417, 530, 537</b>
<b>rising edge</b> .....	<b>383</b>
<b>S</b>	
<b>S-curve</b> .....	<b>345</b>
<b>S700FitRpt</b> .....	<b>897</b>
<b>ScaleInput</b> .....	<b>752</b>
<b>ScaleOutput</b> .....	<b>755</b>
<b>servo axis</b> .....	<b>407</b>
<b>state machine</b>	
motion .....	569
<b>stats</b> .....	<b>684-685, 694</b>
<b>StepAbsolute</b> .....	<b>824</b>
<b>StepAbsSwitch</b> .....	<b>827</b>
<b>StepAbsSwitchFastInput</b> .....	<b>855</b>
<b>StepBlock</b> .....	<b>835</b>
<b>StepLimitSwitch</b> .....	<b>841</b>
<b>StepLimitSwitchFastInput</b> .....	<b>863</b>
<b>StepRefPulse</b> .....	<b>848</b>
<b>SuperimposedCmdPos</b> .....	<b>545, 547</b>



**T**

<b>TemperaturePID</b> .....	<b>765</b>
<b>timeout</b>	
EtherCAT .....	688
<b>torque feed-forward</b> .....	<b>244</b>
<b>trademarks</b> .....	<b>2</b>

**U**

<b>UDFB</b>	
INOUT .....	742
out .....	742

**V**

<b>Velocity, current</b> .....	<b>486, 512</b>
--------------------------------	-----------------

**Z**

<b>Zero Offset</b>	
Pipe Network .....	205

## About KOLLMORGEN

Kollmorgen is a leading provider of motion systems and components for machine builders. Through world-class knowledge in motion, industry-leading quality and deep expertise in linking and integrating standard and custom products, Kollmorgen delivers breakthrough solutions that are unmatched in performance, reliability and ease-of-use, giving machine builders an irrefutable marketplace advantage.



Join the [Kollmorgen Developer Network](#) for product support. Ask the community questions, search the knowledge base for answers, get downloads, and suggest improvements.

### North America KOLLMORGEN

203A West Rock Road  
Radford, VA 24141  
USA

**Web:** [www.kollmorgen.com](http://www.kollmorgen.com)  
**Mail:** [support@kollmorgen.com](mailto:support@kollmorgen.com)  
**Tel.:** +1 - 540 - 633 - 3545  
**Fax:** +1 - 540 - 639 - 4162

### South America KOLLMORGEN

Avenida Tamboré, 1077 Tamboré - Barueri - SP Brasil  
CEP: 06460-000

**Web:** [www.kollmorgen.com](http://www.kollmorgen.com)  
**Mail:** [support@kollmorgen.com](mailto:support@kollmorgen.com)  
**Tel.:** +55 11 4191-4771

### Europe KOLLMORGEN Europe GmbH

Pempelfurtstraße 1  
40880 Ratingen  
Germany

**Web:** [www.kollmorgen.com](http://www.kollmorgen.com)  
**Mail:** [technik@kollmorgen.com](mailto:technik@kollmorgen.com)  
**Tel.:** +49 - 2102 - 9394 - 0  
**Fax:** +49 - 2102 - 9394 - 3155

### China and SEA KOLLMORGEN

Room 202, Building 3, Lane 168,  
Lin Hong Road, Changning District  
Shanghai

**Web:** [www.kollmorgen.cn](http://www.kollmorgen.cn)  
**Mail:** [sales.china@kollmorgen.com](mailto:sales.china@kollmorgen.com)  
**Tel.:** +86 - 400 661 2802