Kollmorgen Automation Suite

KAS Reference Manual - Motion Library

Document Edition: F, May 2015

Valid for KAS Software Revision 2.9

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

Trademarks

KAS and AKD are registered trademarks of Kollmorgen™.
SERVO**STAR** is a registered trademark of Kollmorgen™.
Kollmorgen™ is part of the Danaher Motion company.
Windows® is a registered trademark of Microsoft Corporation
EnDat is a registered trademark of Dr. Johannes Heidenhain GmbH.
EtherCAT® is registered trademark of Ethercat Technology Group.
PLCopen® is an independent association providing efficiency in industrial automation.
INtime® is a registered trademark of TenAsys® Corporation.
Codemeter is a registered trademark of WIBU-Systems AG.

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

Kollmorgen Automation Suite is based on the work of:

- AjaxFileUpload, software (distributed under the MPL License).
- Apache log4net library for output logging (distributed under the Apache License).
- bsdtar and libarchive2, a utility and library to create and read several different archive formats (distributed under the terms of the BSD License).
- bzip2.dll, a data compression library (distributed under the terms of the BSD License).
- Curl software library
- DockPanel Suite, a docking library for .Net Windows Forms (distributed under the MIT License).
- FileHelpers library to import/export data from fixed length or delimited files.
- GNU gzip[1] (www.gnu.org) is used by the PDMM (distributed under the terms of the GNU General Public License http://www.gnu.org/licenses/gpl-2.0.html).
- GNU Tar[2] (www.gnu.org) is used by the PDMM (distributed under the terms of the GNU General Public License http://www.gnu.org/licenses/gpl-2.0.html).
- Icons provided by Oxygen Team, (distributed under the terms of the GNU Lesser General Public License https://www.gnu.org/licenses/lgpl.html).
- jQuery.Cookies, a Javascript library for accessing and manipulating HTTP cookies in the web browser (distributed under the MIT License).
- jquery-csv, a library for parsing CSV files in javascript (distributed under the MIT license http://www.opensource.org/licenses/mit-license.php).
- jQuery File Tree, a file browser plugin (distributed under the MIT License).
- jQueryRotate, a plugin which rotates images (img html objects) by a given angle on web pages (distributed under the MIT License, http://opensource.org/licenses/mit-license.php).

[1]Copyright (C) 2007 Free Software Foundation, Inc. Copyright (C) 1993 Jean-loup Gailly. This is free software. You may redistribute copies of it under the terms of the GNU General Public License <http://www.gnu.org/licenses/gpl.html>. There is NO WARRANTY, to the extent permitted by law. Written by Jean-loup Gailly.
[2]Copyright (C) 2007 Free Software Foundation, Inc. License GPLv2+: GNU GPL version 2 or later <http://gnu.org/licenses/gpl.html> This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law. Written by John Gilmore and Jay Fenlason.

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

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

Disclaimer

The information in this document (Version 2.9 published on 5/18/2015) is believed to be accurate and reliable at the time of its release. Notwithstanding the foregoing, Kollmorgen assumes no responsibility for any damage or loss resulting from the use of this help, and expressly disclaims any liability or damages for loss of data, loss of use, and property damage of any kind, direct, incidental or consequential, in regard to or arising out of the performance or form of the materials presented herein or in any software programs that accompany this document.

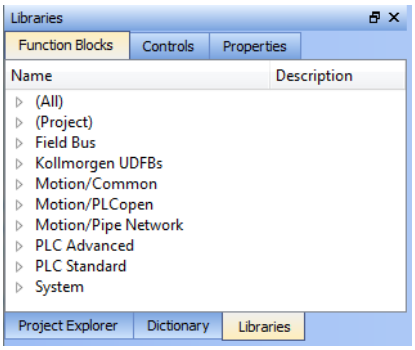
All timing diagrams, whether produced by Kollmorgen or included by courtesy of the PLCopen organization, are provided with accuracy on a best-effort basis with no warranty, explicit or implied, by Kollmorgen. The user releases Kollmorgen from any liability arising out of the use of these timing diagrams.

This page intentionally left blank.

1 Table of Contents

2 Motion Library

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

Table 1-1: List of Pipe Network FB

IMPORTANT

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

2.1.1 Motion Library - Pipe Network

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

2.1.1.1 MLPipeAct

Description

Activates a pipe. A Pipe contains an Input Pipe Block (Master, PMP, or Sampler), a Converter Output Pipe Block, and any Transformation Pipe Block that can be in between. The figure below shows two Pipes, both with the same Master Input Pipe Block. The first ends with the first converter, and has a Gear Pipe Block to transform the input values from the Master. The second pipe ends with the second converter, and has a CAM Pipe Block to modify the input values from the Master.

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

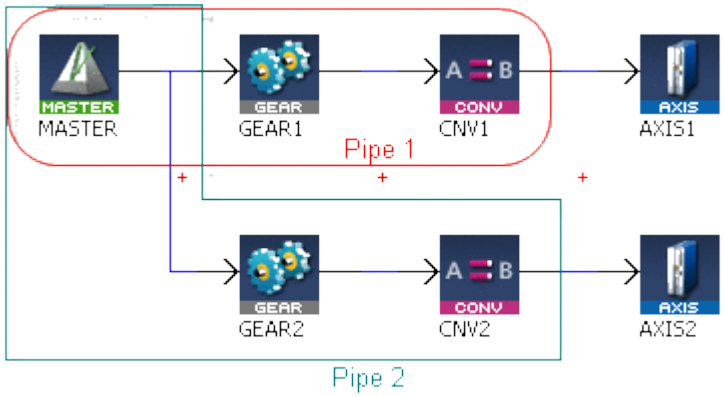


Figure 1-1: MLPipeAct

NOTE

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

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLPipeDeact

"MLCNVConnect" (➜ p. 193)

PipeNetwork(MLPN_ACTIVATE)

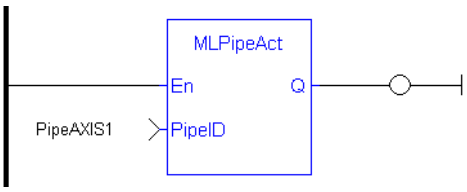
MLPipeAddBlock

Example

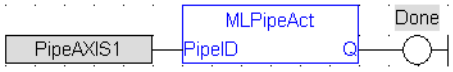
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.1.2 MLPipeAddBlock

Description

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

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

MLPipeAddBlock(PipeAXIS1, MASTER);

MLPipeAddBlock(PipeAXIS1, MyGear);

MLPipeAddBlock(PipeAXIS1, CNV1);

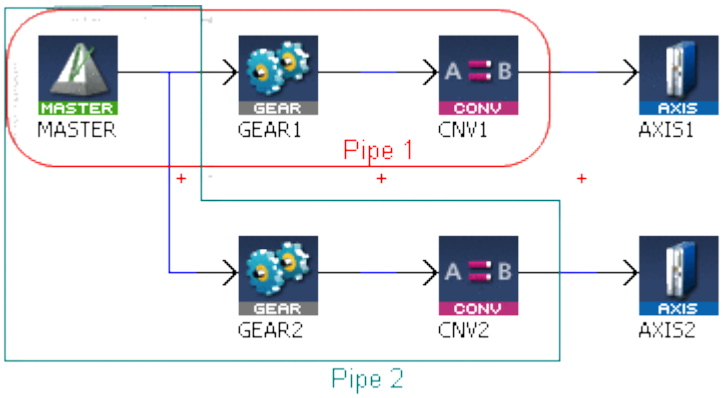


Figure 1-2: MLPipeAddBlock

NOTE

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

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

PipeNetwork(MLPN_CREATE_OBJECTS)

MLPipeAct

MLPipeCreate

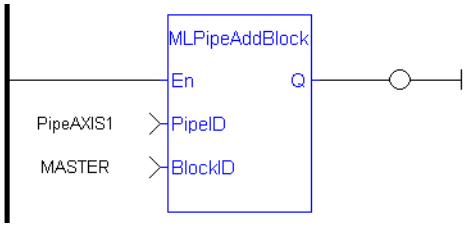
MLPipeDeact

Example

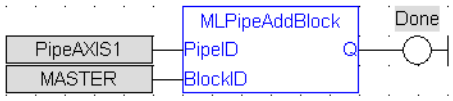
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.1.3 MLPipeCreate

Description

Create a new pipe object. A Pipe contains an Input Pipe Block (Master, PMP, or Sampler), a Converter Output Pipe Block, and any Transformation Pipe Block that can be in between. The figure below shows two Pipes, both with the same Master Input Pipe Block.

NOTE

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

Arguments

Input

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

Output

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

Related Functions

PipeNetwork(MLPN_CREATE_OBJECTS)

MLPipeAddBlock

MLPipeAct

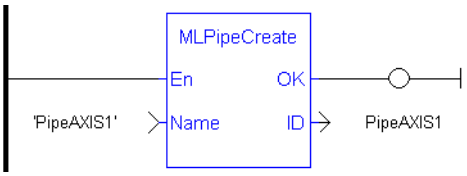
MLPipeDeact

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.1.4 MLPipeDeact

Description

Deactivates a pipe. A Pipe contains an Input Pipe Block (Master, PMP, or Sampler), a Converter Output Pipe Block, and any Transformation Pipe Block that can be in between. The figure below shows two Pipes, both with the same Master Input Pipe Block. The first ends with the first converter, and has a Gear Pipe Block to transform the input values from the Master. The second pipe ends with the second converter, and has a CAM Pipe Block to modify the input values from the Master.

Once a Pipe is activated then history on the values in the Pipe's Blocks are lost and no longer updated. A Converter object connected to a destination Axis object cannot send updated position values once its Pipe is deactivated.

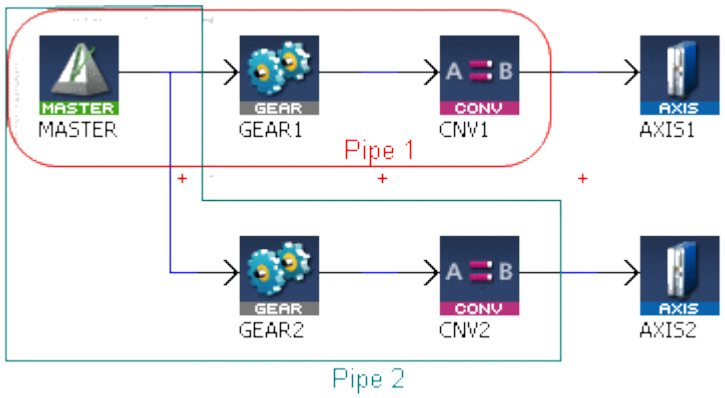


Figure 1-3: MLPipeDeact

NOTE

All Pipes in the Pipe Network can be deactivated at once with the command PipeNetwork(MLPN_DEACTIVATE). This calls automatically generated code with MLPipeDeact commands for each Pipe object. Therefore, in a multi-pipe program only one command can be used to deactivate Pipes instead of writing code for each Pipe separately.

Arguments

Input

PipeID	**Description**	ID number of a created Pipe object
	Data type	DINT
	Range	[-2147483648, 2147483648]

	Unit	n/a
	Default	—

Output

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

Return Type

BOOL

Related Functions

MLPipeAct

"MLCNVDisconnect" (➜ p. 199)

PipeNetwork(MLPN_DEACTIVATE)

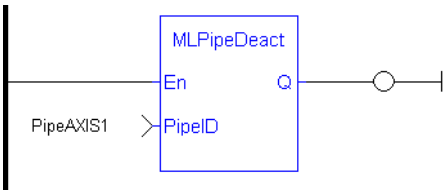
MLPipeAddBlock

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.2 Motion Library - Block

Name	Description	Return type
"MLBlkCreate" (➜ p. 86)	Creates a new Pipe Block object	None
"MLBlkIsReady" (➜ p. 89)	Checks if a Pipe Block currently has a function running	BOOL
"MLBlkReadModPos" (➜ p. 88)	Gets the value of the period of a block in user units	None

Name	Description	Return type
"MLBlkReadOutVal" (➜ p. 87)	Gets the output value of a selected Pipe Block	None
"MLBlkWriteModPos" (➜ p. 90)	Sets the value of the period of a block in user units	BOOL

2.1.2.1 MLBlkCreate

Description

Creates a new Pipe Block object. Before a Pipe Block is Initialized the block needs to be created and assigned an ID number. MLBlkCreate 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 MLBlkCreate function blocks to their programs. Blocks are created graphically, and the code with MLBlkCreate 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. 426) is called.

Arguments

Input

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

Output

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

Related Functions

PipeNetwork(MLPN_CREATE_OBJECTS)

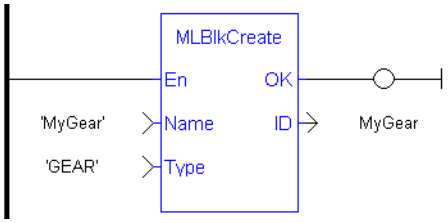
MLAxisInit

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.2.2 MLBlkReadOutVal

Description

Get the output value a selected Pipe Block.

Arguments

Input

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

Output

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

Related Functions

MLBlkReadModPos

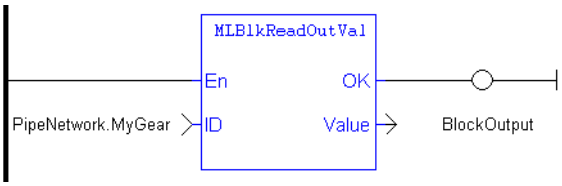
MLBlkCreate

Example

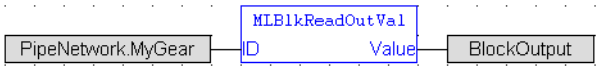
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.2.3 MLBlkReadModPos

Description

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

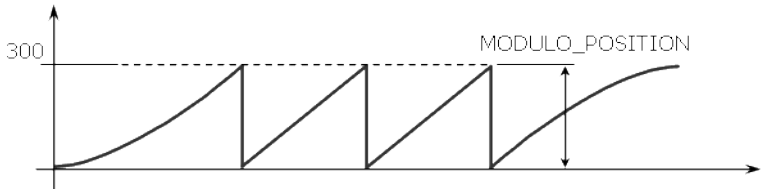


Figure 1-4: MLBlkReadModPos

Arguments

Input

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

Output

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

Default —

Related Functions

MLBlkWriteModPos

MLBlkCreate

MLBlkReadOutVal

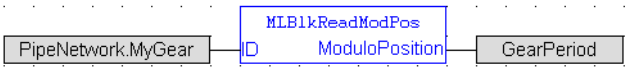
Example

Structured Text

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

Ladder Diagram

Function Block Diagram



2.1.2.4 MLBlkIsReady

Description

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

NOTE

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

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLBlkReadOutVal

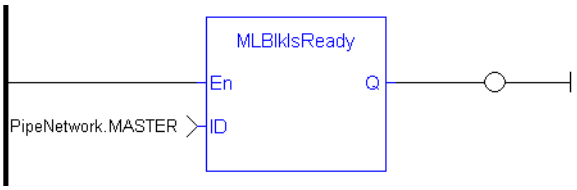
MLBlkReadModPos

Example

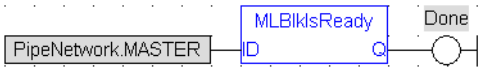
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.2.5 MLBlkWriteModPos

Description

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

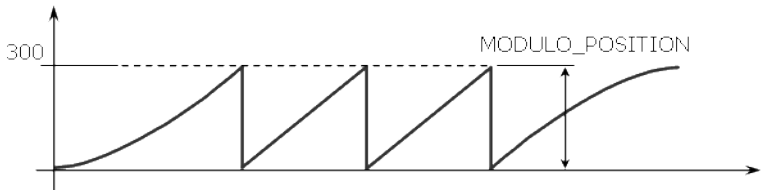


Figure 1-5: MLBlkReadModPos

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLBlkReadModPos

MLBlkCreate

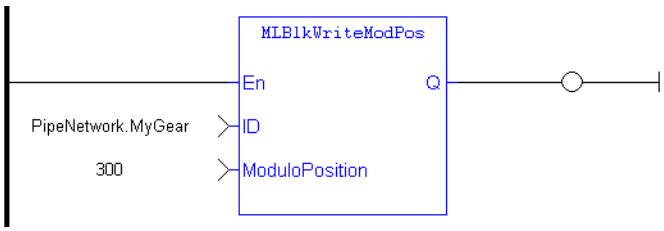
MLBlkReadOutVal

Example

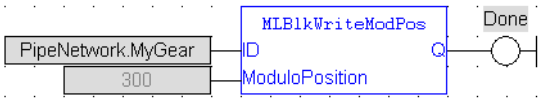
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.3 Motion Library - Adder

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

2.1.3.1 MLAddInit

Description

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

The Pipe Block is assigned ratios and offsets for both inputs. After an Adder block is initialized, the inputs still need to be selected using the MLAddWriteInput function block or graphically using the Pipe Network.

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

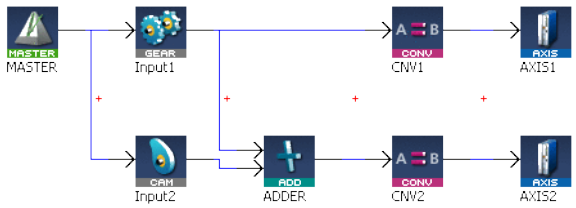


Figure 1-6: MLAddInit

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.

Arguments

Input

BlockID	**Description**	ID number of a created Pipe Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
Ratio1	**Description**	Sets the Ratio value of the first entry of an Adder object
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—
Offset1	**Description**	Sets the Offset value of the first entry of an Adder object
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—
Ratio2	**Description**	Sets the Ratio value of the second entry of an Adder object
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—
Offset2	**Description**	Sets the Offset value of the second entry of an Adder object
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—

Output

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

Return Type

BOOL

Related Functions

MLBlkCreate

MLAddWriteInput

MLAddReadOff1

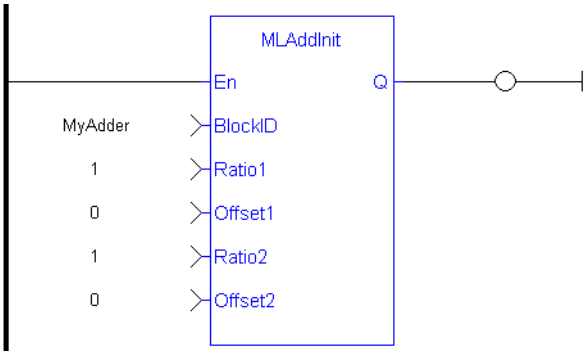
MLAddReadRatio1

Example

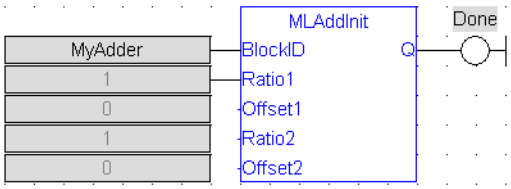
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.3.2 MLAddReadOff1

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

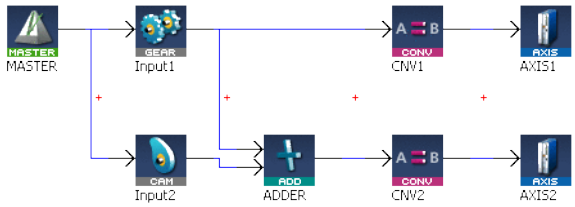


Figure 1-7: MLAddReadOff1

Arguments

Input

BlockID

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

Output

Offset

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

Related Functions

MLAddWriteOff1

MLAddReadOff2

MLAddReadRatio1

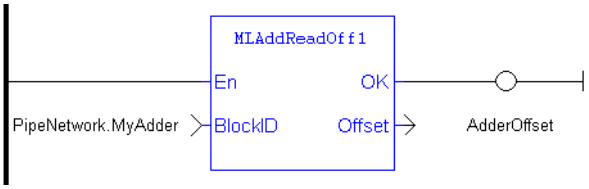
MLAddWriteRat1

Example

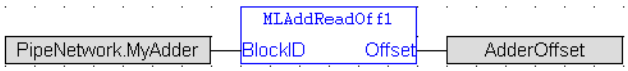
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.3.3 MLAddReadOff2

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

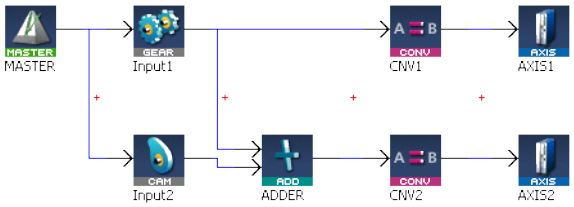


Figure 1-8: MLAddReadOff2

Arguments

Input

BlockID

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

Output

Offset

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

Related Functions

MLAddWriteOff2

MLAddReadOff1

MLAddReadRatio2

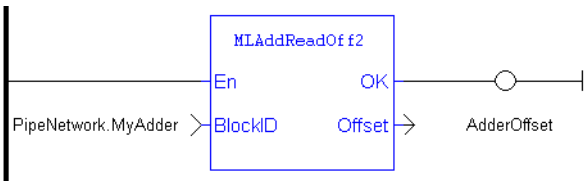
MLAddWriteRat2

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.3.4 MLAddReadRatio1

Description

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

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

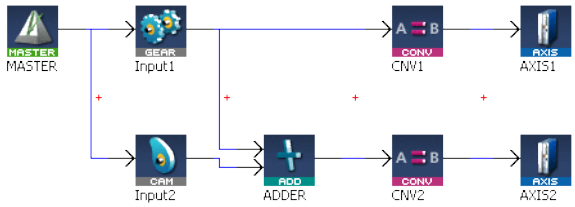


Figure 1-9: MLAddReadRatio1

Arguments

Input

BlockID

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

Output

Ratio

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

Related Functions

MLAddWriteRat1

MLAddReadRatio2

MLAddReadOff1

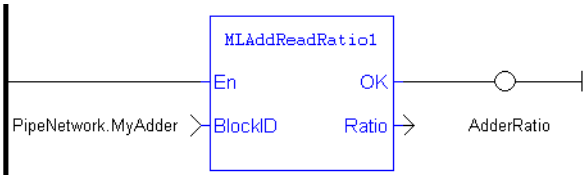
MLAddReadOff2

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.3.5 MLAddReadRatio2

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

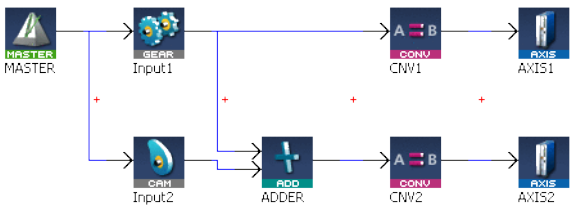


Figure 1-10: MLAddReadRatio2

Arguments

Input

BlockID	**Description**	ID number of an initiated Adder object
	Data type	DINT

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

Output

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

Related Functions

MLAddWriteRat2

MLAddReadRatio1

MLAddReadOff1

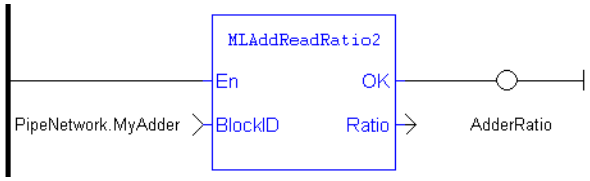
MLAddReadOff2

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.3.6 MLAddWriteInput

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.

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

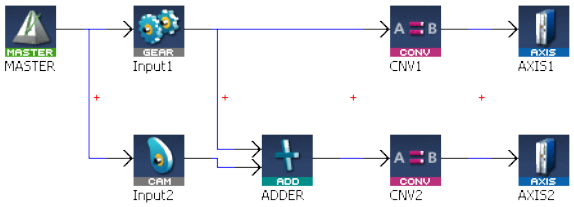


Figure 1-11: MLAddWriteInput

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.

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLBlkCreate

MLAddInit

MLAddReadOff1

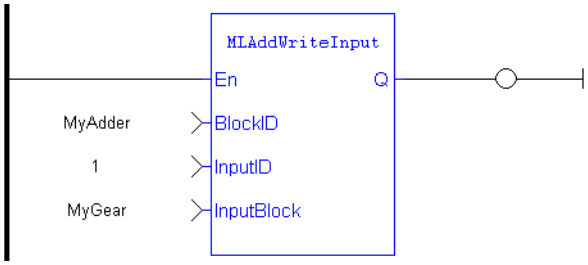
MLAddReadRatio1

Example

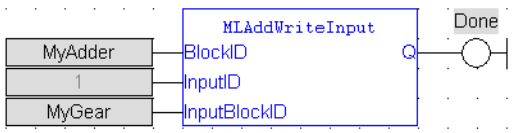
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.3.7 MLAddWriteOff1

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

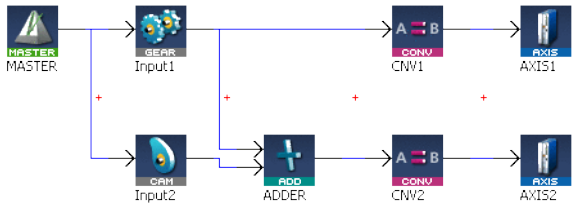


Figure 1-12: MLAddWriteOff1

IMPORTANT

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

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLAddReadOff1

MLAddWriteOff2

MLAddReadRatio1

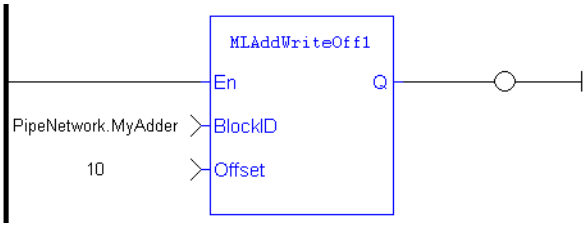
MLAddWriteRat1

Example

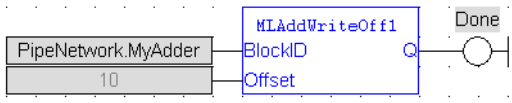
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.3.8 MLAddWriteOff2

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

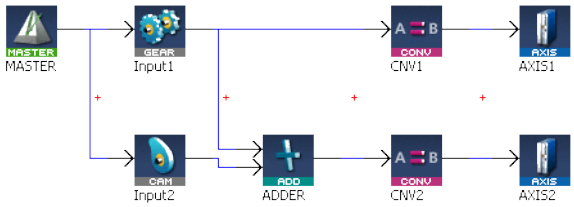


Figure 1-13: MLAddWriteOff2

IMPORTANT

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

Arguments

Input

BlockID

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

Offset	**Description**	Desired new value for the Adder Object's Offset2
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—

Output

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

Return Type

BOOL

Related Functions

MLAddReadOff2

MLAddWriteOff1

MLAddReadRatio2

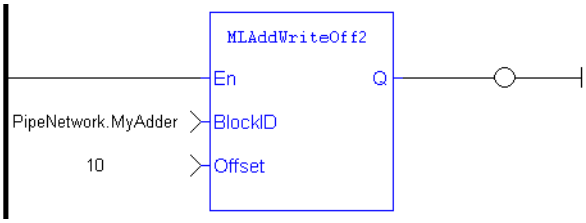
MLAddWriteRat2

Example

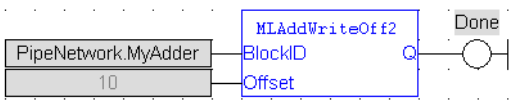
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.3.9 MLAddWriteRat1

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.

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

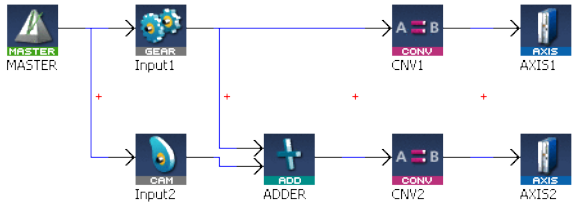


Figure 1-14: MLAddWriteRat1

IMPORTANT

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

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLAddReadRatio1

MLAddWriteRat2

MLAddReadOff1

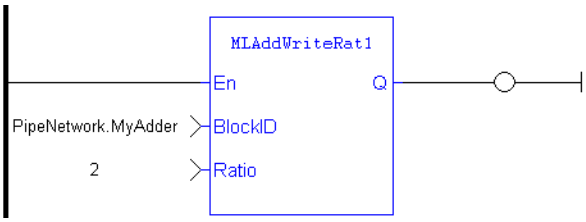
MLAddWriteOff1

Example

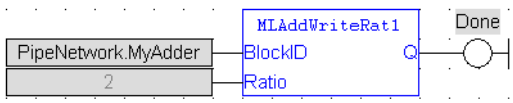
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.3.10 MLAddWriteRat2

Description

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

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

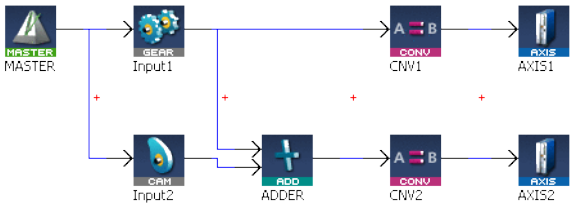


Figure 1-15: MLAddWriteRat2\

IMPORTANT

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

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLAddReadRatio2

MLAddWriteRat1

MLAddReadOff2

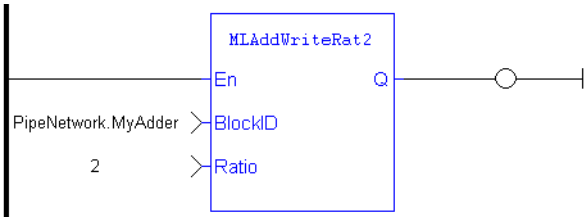
MLAddWriteOff2

Example

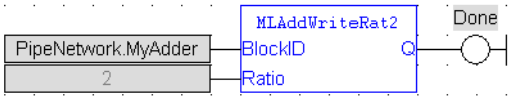
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.4 Motion Library - Axis

For usage example about Axis Functions, see "Usage Example of Axis Functions" on page 165

Function sorted by types:

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

Functions sorted in alphabetical order:

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

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

2.1.4.1 MLAxisAbs

Description

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

Arguments

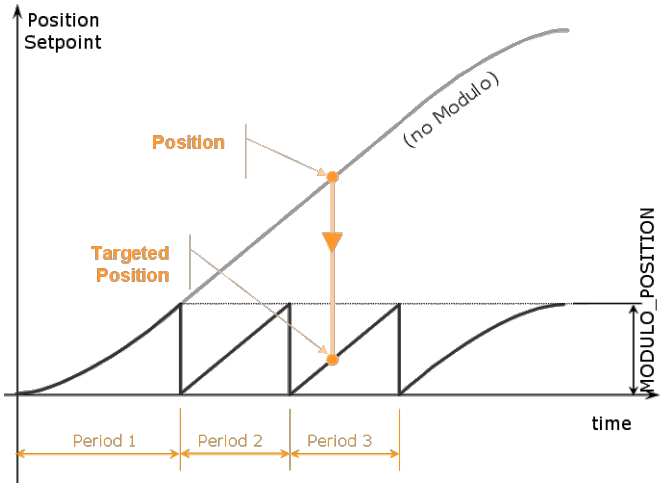
Input

ID	**Description**	ID name of the Axis Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
Position	**Description**	Sets the value of the absolute destination position. When the Modulo is turned on, see more explanations below.
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

2.1.4.2 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).

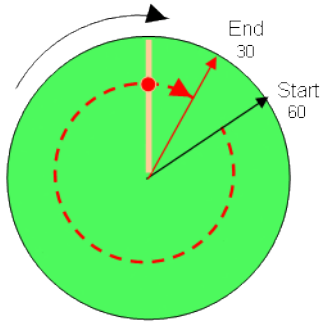


The Axis Block works similarly for negative positions: if the Position input is less than zero, then the Axis Block moves during one of the **previous** period (negative position rollover).

2.1.4.3 Forcing the direction of rotation

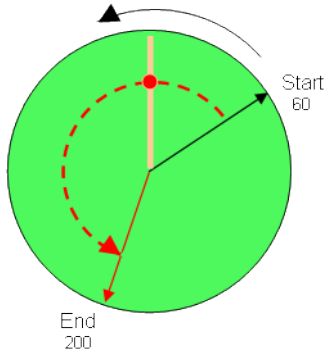
In some applications, the direction of rotation for the axis is forced in one direction only. As a consequence, the motor movement goes to the next or previous modulo in the following situations:

- 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	**Yes**	390	330	(i.e. 30 - 60 + 360)
60	30	counter clockwise	No	30	-30	(i.e. 30 - 60 - 0)
60	200	counter clockwise	**Yes**	-160	-220	(i.e. 200 - 60 - 360)

With:

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

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

Where:

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

2.1.4.4 Travel Speed Update with MLAxisAbs

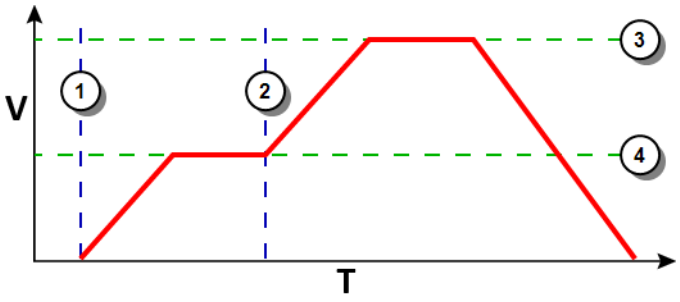
The travel speed of the generator can be updated using the function block "MLAxisGenWriteSpd" (➜ p. 129). Depending on the state of the generator, this speed is directly reflected on the current move or a future move.

- 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.

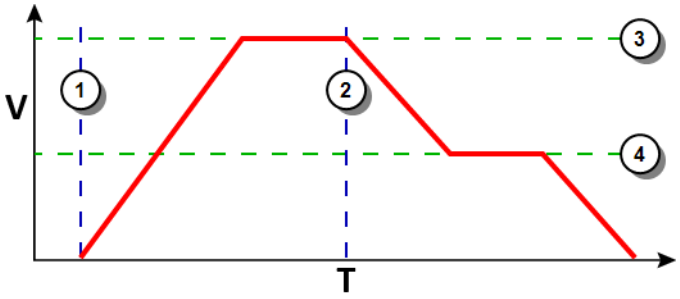
- If a MLAxisAbs is currently being executed and a new MLAxisAbs with a different target position is called, the new travel speed is taken into account.

Following are several examples.



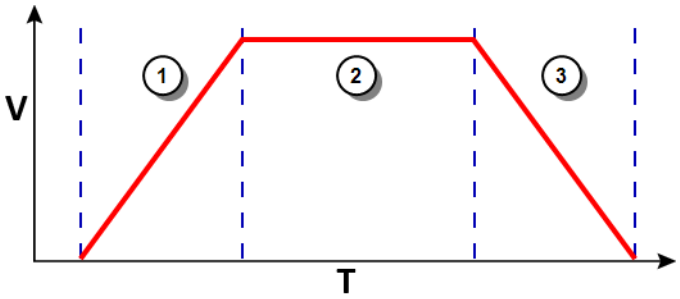
1. First MLAxisAbs
2. Second MLAxisAbs
3. Initial Speed
4. New Speed

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



1. First MLAxisAbs
2. Second MLAxisAbs
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 MLAxisAbs is triggered during acceleration or constant velocity

Output

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

Related Functions

MLAxisGenWriteSpd

MLAxisGenWriteDec

MLAxisGenWriteAcc

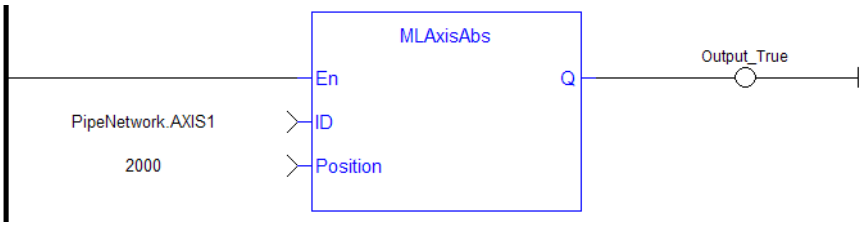
Example

See "Usage Example of Axis Functions" (➜ p. 165) for additional examples.

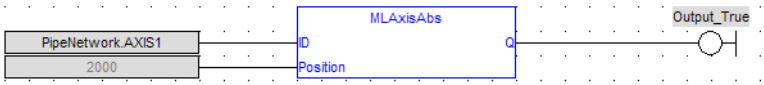
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.4.5 MLAxisAdd

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.

Arguments

Input

ID	**Description**	ID Name of the Axis block
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—
DeltaPosition	**Description**	Sets the Axis Delta Position to add to the endpoint of the previous move
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

Default (.Q)

Description	Returns true when function successfully executes, after the motion profile is complete
Data type	BOOL
Unit	n/a

Related Functions

MLAxisGenWriteAcc

MLAxisGenWriteDec

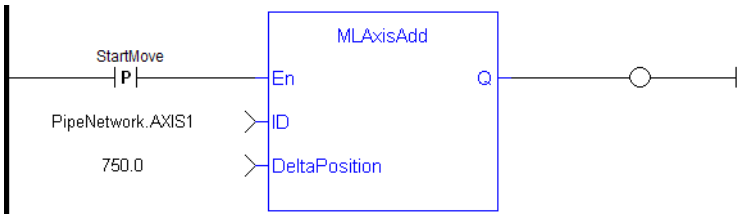
MLAxisGenWriteSpd

Example

Structured Text

```
MLAxisAdd(PipeNetwork.Axis1, LREAL#720.0 ) ;
```

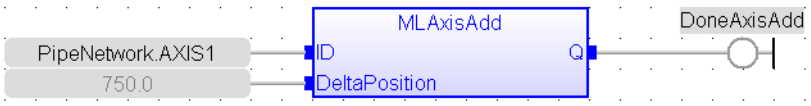
Ladder Diagram



NOTE

You must use a pulse contact to start the FB

Function Block Diagram



2.1.4.6 MLAxisAddress

Description

Returns the motion bus address of the axis

Arguments

Input

ID

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

Output

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

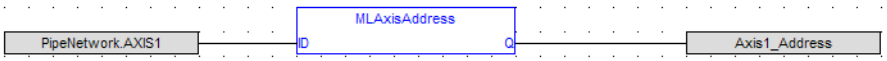
Example

Structured Text

```
MLAxisAddress( PipeNetwork.Axis1 );
```

Ladder Diagram

Function Block Diagram



2.1.4.7 MLAxisAddTq

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. 139) function has been invoked. Using the PDOPDO, it also requires IL.KBUSFFIL.KBUSFF value to be set to 1 in the drive.

Arguments

Input

ID	**Description**	Pipe network identifier of the axis block
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—
Torque	**Description**	Requested additive torque value in N.m (Newton meter).
	Data type	LREAL
	Unit	Rated torque units as used in the drive (i.e. Peak Motor Current times the Torque factor).

Output

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

Related Functions

"MLAxisRatedTq" (➔ p. 139)

Example

Structured Text

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

2.1.4.8 MLAxisCfgFastIn

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.

Arguments

Input

En	**Description**	Enables execution
	Data type	BOOL
	Unit	n/a
	Default	-
AxisID	**Description**	ID name of the Axis Block
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—
InputID	**Description**	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 AKD Capture Engine Configuration.
	Data type	DINT
	Range	[0, 1]
	Unit	n/a
	Default	—
Mode	**Description**	Configures the Fast Inputs as 0= Disabled, 1=Rising Edge, 2=Falling edge
	Data type	DINT
	Range	[0, 2]
	Unit	n/a
	Default	—

Output

Q	**Description**	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.
	Data type	BOOL
	Unit	n/a

Related Functions

MLAxisIsTrigged

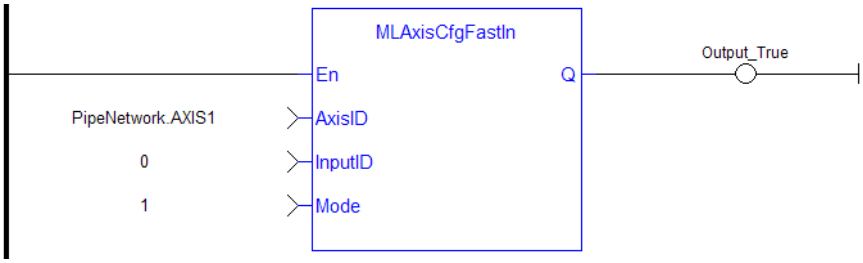
MLAxisRstFastIn

Example

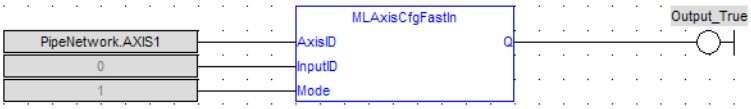
Structured Text

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

Ladder Diagram



Function Block Diagram



See also "Fast inputs" for more details.

2.1.4.9 MLAxisCmdPos

Description

Returns the reference position of the axis.

Arguments

Input

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

Output

OK	**Description**	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a
Position	**Description**	Returns the Axis reference position
	Data type	LREAL
	Unit	User unit

Related Functions

MLAxisReadActPos

MLAxisFBackPos

MLAxisGenPos

MLAxisPipePos

MLAxisWritePipPos

Previous Function Name

MLAxisRefPos

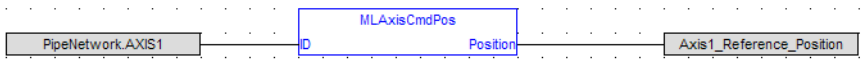
Example

Structured Text

```
MLAxisCmdPos(PipeNetwork.Axis1 ) ;
```

Ladder Diagram

Function Block Diagram



2.1.4.10 MLAxisCreate

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. 426) is called.

Arguments

Input

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

Output

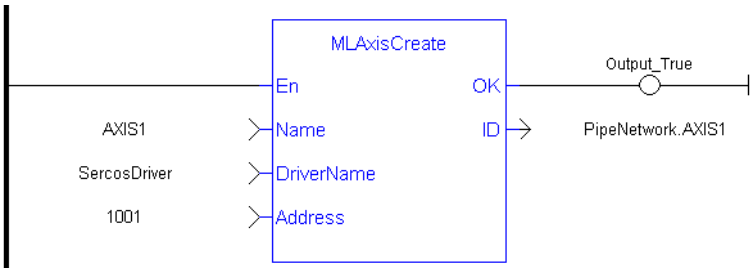
OK	**Description**	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a

Example

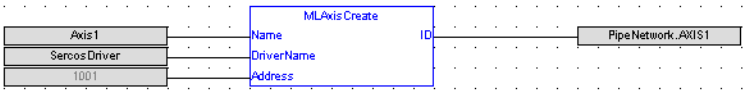
Structured Text

```
MLAxisCreate( 'AXIS1', 'MSBusDriver', 1001);
```

Ladder Diagram



Function Block Diagram



2.1.4.11 MLAxisFBackPos

Description

Returns the Feedback Position of the axis

Arguments

Input

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

Output

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

Related Functions

MLAxisReadActPos

MLAxisGenPos

MLAxisPipePos

MLAxisCmdPos

MLAxisWritePipPos

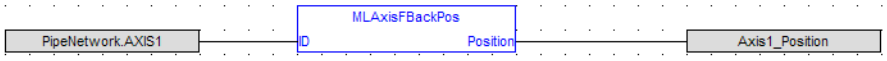
Example

Structured Text

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

Ladder Diagram

Function Block Diagram



2.1.4.12 MLAxisGenEN

Description

Enables or disables the internal TMP generator of the axis.

Arguments

Input

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

Output

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

Related Functions

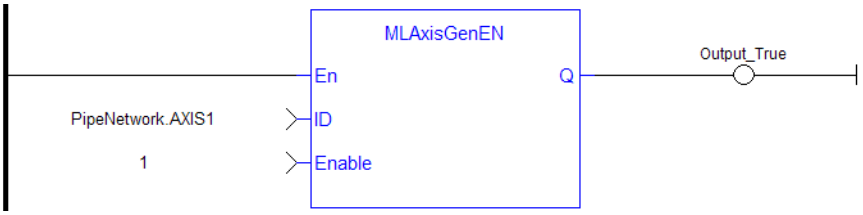
MLAxisGenIsEN

Example

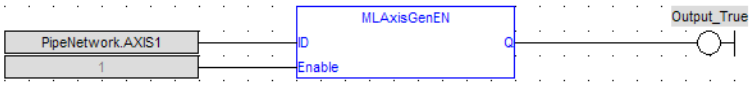
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.4.13 MLAxisGenIsEN

Description

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

Arguments

Input

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

Output

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

Related Functions

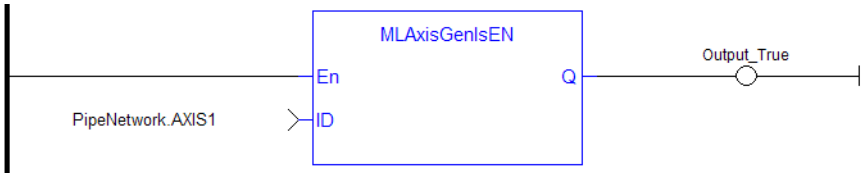
MLAxisGenIsRdy

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.4.14 MLAxisGenIsRdy

Description

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

Arguments

Input

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

Output

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

Related Functions

MLAxisGenIsEN

MLAxisStatus

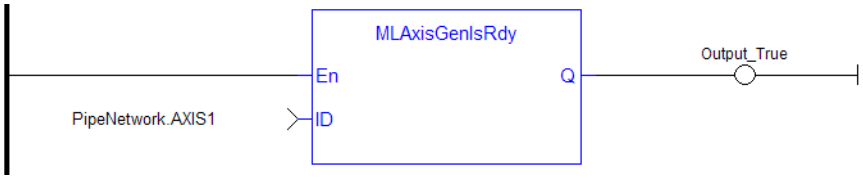
Example

See "Usage Example of Axis Functions" (➜ p. 165) for additional examples.

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.4.15 MLAxisGenPos

Description

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

Arguments

Input

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

Output

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

Related Functions

MLAxisReadActPos

MLAxisFBackPos

MLAxisPipePos

MLAxisCmdPos

MLAxisWritePipPos

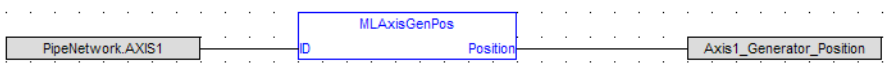
Example

Structured Text

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

Ladder Diagram

Function Block Diagram



2.1.4.16 MLAxisGenReadAcc

Description

Get the acceleration of the internal generator of an axis.

Arguments

Input

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

Output

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

Related Functions

MLAxisGenReadDec

MLAxisGenReadSpd

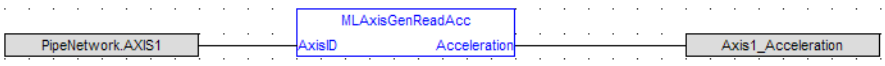
Example

Structured Text

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

Ladder Diagram

Function Block Diagram



2.1.4.17 MLAxisGenReadDec

Description

Get the Deceleration of the internal generator of an axis.

Arguments

Input

AxisID	**Description**	ID Name of the Axis block
	Data type	DINT

	Range	—
	Unit	n/a
	Default	—

Output

OK	**Description**	
	Data type	BOOL
	Unit	n/a
Deceleration	**Description**	Returns Axis Deceleration value
	Data type	LREAL
	Unit	User unit/sec^2

Related Functions

MLAxisGenReadAcc

MLAxisGenReadSpd

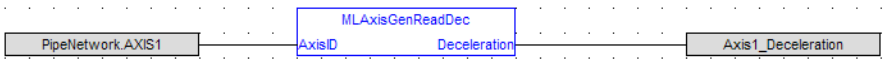
Example

Structured Text

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

Ladder Diagram

Function Block Diagram



2.1.4.18 MLAxisGenReadSpd

Description

Get the speed of the internal generator of an axis.

Arguments

Input

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

Output

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

Related Functions

MLAxisGenReadDec

MLAxisGenReadAcc

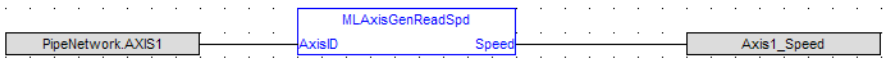
Example

Structured Text

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

Ladder Diagram

Function Block Diagram



2.1.4.19 MLAxisGenWriteAcc

Description

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

Arguments

Input

AxisID	**Description**	ID Name of the Axis block
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—
Acceleration	**Description**	Sets the generator Acceleration value

Data type	LREAL
Range	—
Unit	User unit/sec^2
Default	—

Output

Default (.Q)

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

Related Functions

MLAxisGenWriteDec

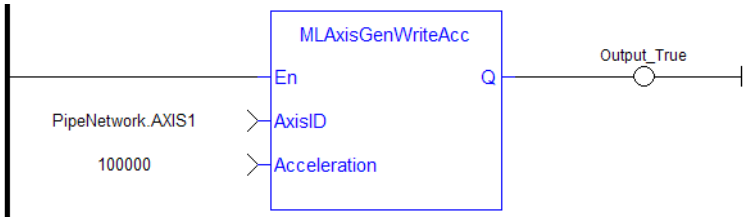
MLAxisGenWriteSpd

Example

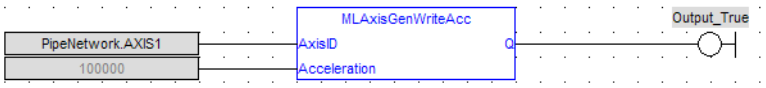
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.4.20 MLAxisGenWriteDec

Description

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

Arguments

Input

AxisID

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

	Default	—
Deceleration	**Description**	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.
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—

Output

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

Related Functions

MLAxisGenWriteAcc

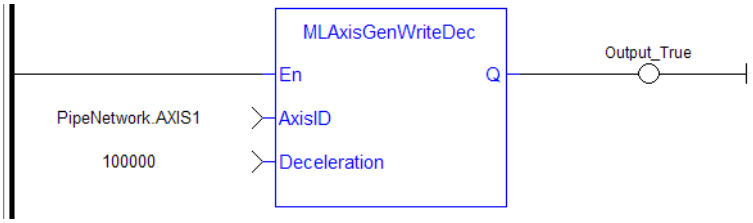
MLAxisGenWriteSpd

Example

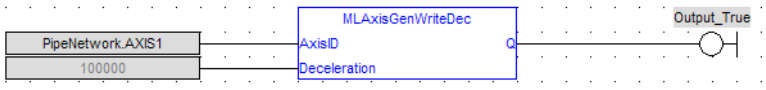
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.4.21 MLAxisGenWriteSpd

Description

Set the speed of the internal generator of an axis. Returns TRUE if the function succeeded.

Arguments

Input

AxisID	**Description**	ID Name of the Axis block
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—
Speed	**Description**	Sets the generator Speed value
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

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

Related Functions

MLAxisGenWriteAcc

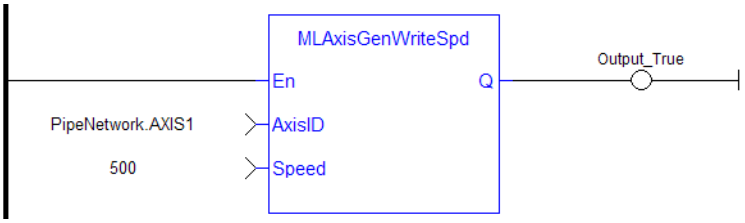
MLAxisGenWriteDec

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.4.22 MLAxisInit

Description

Initializes an axis object. Returns TRUE if the function succeeded

Arguments

Input

AxisID	**Description**	ID Name of the Axis block
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—
ModuloPosition	**Description**	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
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
UserUnitPerTurn	**Description**	Define the unit which is equivalent to one revolution of the physical motor
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—
FeedbackUnitPerTurn	**Description**	
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—
Speed	**Description**	Sets the Axis Speed
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
Acceleration	**Description**	Sets the Axis Acceleration value
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Deceleration	**Description**	Sets the Axis Deceleration value
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
InitialPosition	**Description**	Initial position value expressed in user logical units. Used only at the pipe activation to initialize the position starting point

	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
Modulo	**Description**	Define the mode which can be Modulo (True) or not (False)
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—

Output

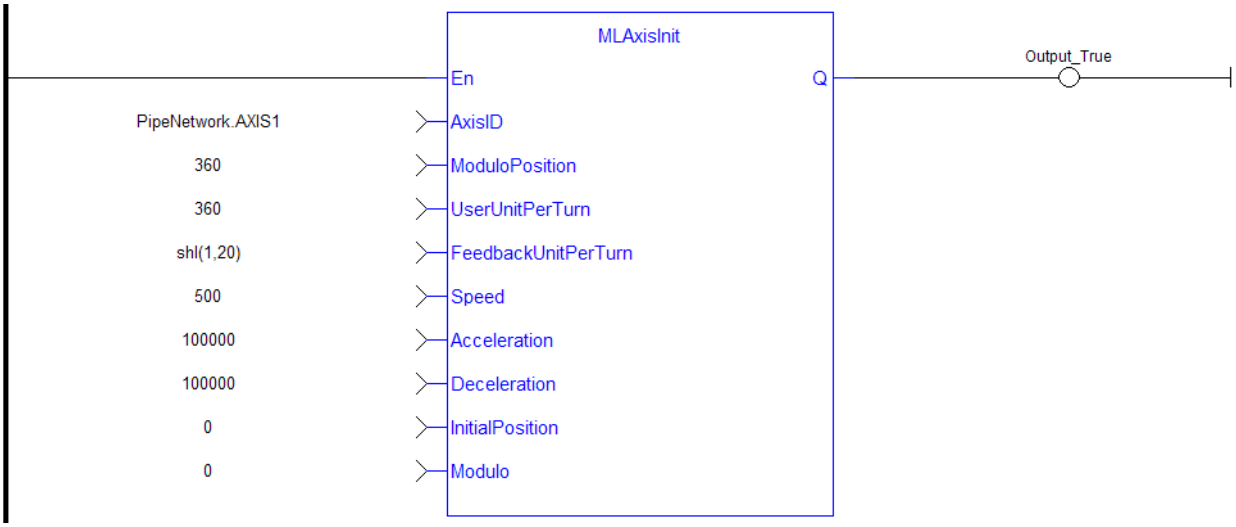
Default (.Q)	**Description**	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a

Example

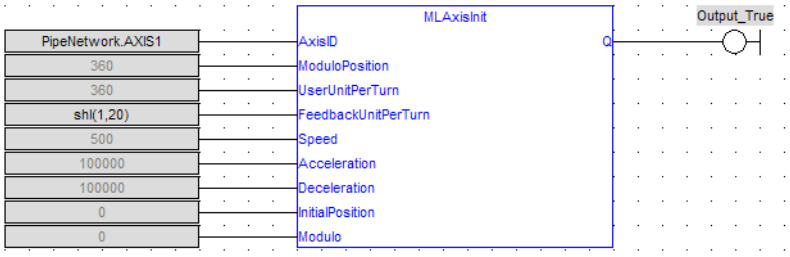
Structured Text

```
MLAxisInit( PipeNetwork.Axis1, 360.0, 360.0, SHL(1,20), 1000.0,
10000.0, 10000.0, 0.0, true ) ;
```

Ladder Diagram



Function Block Diagram



2.1.4.23 MLAxisIsCnctd

Description

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

Arguments

Input

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

Output

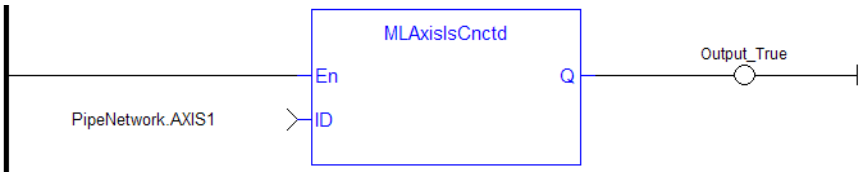
Default (.Q)	**Description**	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a

Example

Structured Text

```
MLAxisIsCnctd(PipeNetwork.Axis1 ) ;
```

Ladder Diagram



Function Block Diagram

2.1.4.24 MLAxisIsTrigged

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

Arguments

Input

ID	**Description**	ID Name of the Axis block
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—
InputID	**Description**	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 AKD Capture Engine Configuration.
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—
edge	**Description**	Configures the Inputs as 0= Disabled, 1=Rising Edge, 2=Falling edge
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—

Output

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

Related Functions

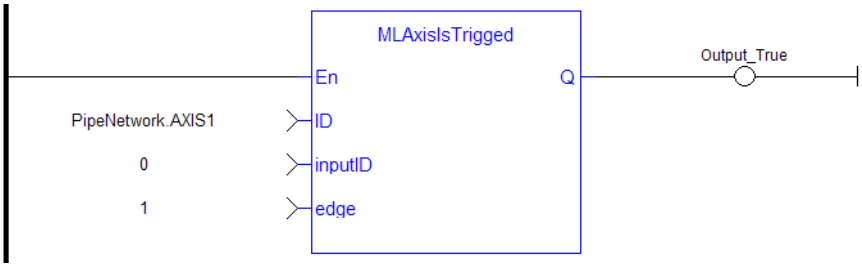
MLAxisRstFastIn

Example

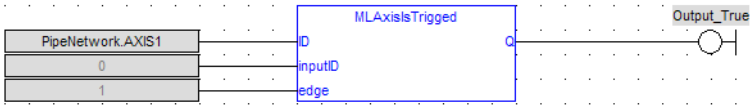
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.4.25 MLAxisMoveVel

Description

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

Arguments

Input

ID	**Description**	ID Name of the Axis block
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—
Speed	**Description**	Sets the Axis Speed
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—

Output

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

Related Functions

MLAxisGenWriteSpd

MLAxisGenWriteDec

MLAxisGenWriteAcc

Previous Function Name

MLAxisRun

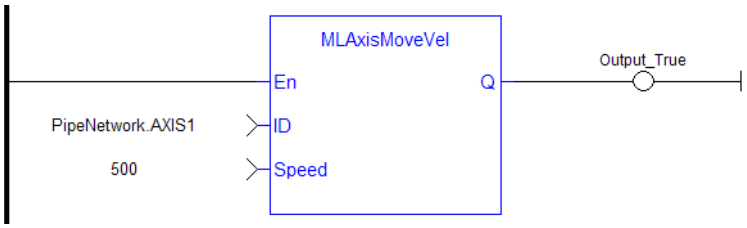
Example

See "Usage Example of Axis Functions" (➜ p. 165) for additional examples.

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.4.26 MLAxisPipePos

Description

Returns the pipe position of the axis.

Arguments

Input

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

Output

OK	**Description**	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a
Position	**Description**	
	Data type	LREAL
	Range	—
	Unit	User unit

Related Functions

MLAxisReadActPos

MLAxisFBackPos

MLAxisGenPos

MLAxisCmdPos

MLAxisWritePipPos

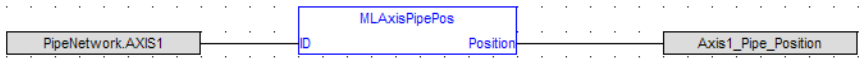
Example

Structured Text

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

Ladder Diagram

Function Block Diagram



2.1.4.27 MLAxisPower

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**.

Arguments

Input

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

Output

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

Related Functions

MLAxisPowerDOff

Previous Function Name

MLAxisPowerOn

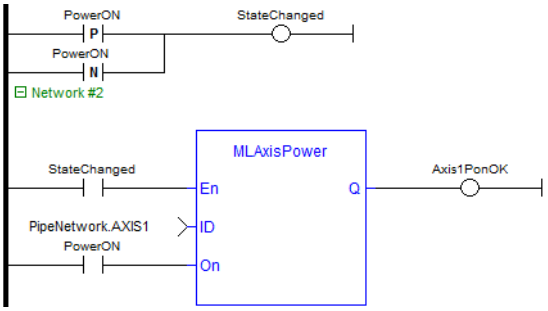
MLAxisPowerOff

Example

Structured Text

```
MLAxisPower( PipeNetwork.Axis1, PowerUp(*BOOL*)) ;
```

Ladder Diagram



Function Block Diagram

2.1.4.28 MLAxisPowerDOff

Description

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

Arguments

Input

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

Output

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

Related Functions

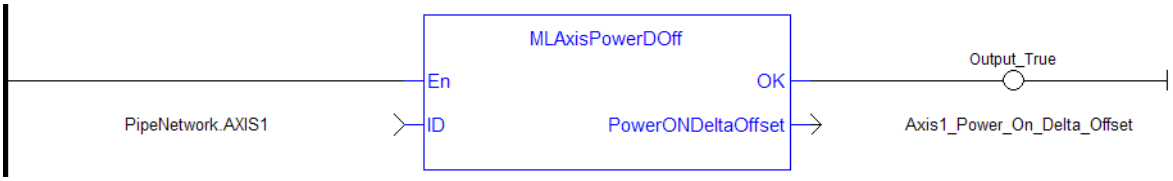
MLAxisPower

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.4.29 MLAxisRatedTq

Description

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

Arguments

Input

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

Torque	Description	Actual torque applied by the drive associated to the axis Rated torque = Peak Motor Current * Torque factor = MOTOR.IPEAK * MOTOR.KT About SDO MOTOR.IPEAK is obtained by SDO parameter: index 358Fh (sub-index 0) MOTOR.KT is obtained by SDO parameter: index 3593h (sub-index 0) For more details, refer to: • Communication SDOs • Manufacturer specific SDOs • Profile specific SDOs The actual units of MOTOR.IPEAK and MOTOR.KT are 1/1000 of the actual values if obtained by SDO. So the formula, if using the SDO values, is: Rated Torque = Torque = (SDO(MOTOR.IPEAK)/1000) * (SDO(MOTOR.KT)/1000)
	Data type	LREAL
	Unit	N.m (Newton meter)

Output

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

Related Functions

MLAxisReadTq

Example

Structured Text

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

2.1.4.30 MLAxisRead2ndFB

Description

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

Arguments

Input

ID	Description	Pipe network identifier of the axis block
	Data type	DINT
	Range	—

	Unit	n/a
	Default	—

Output

Position	Description	Position value returned by the secondary feedback
	Data type	LREAL
	Unit	User unit

Related Functions

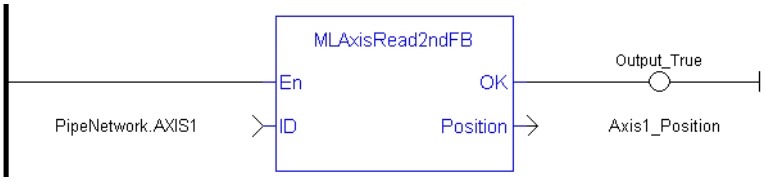
MLAxisReadActPos

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.4.31 MLAxisReadActPos

Description

Returns the Actual Position of the axis

Arguments

Input

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

Output

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

Position	Description	Returns the absolute position of the axis
	Data type	LREAL
	Unit	User unit

Related Functions

MLAxisFBackPos

MLAxisGenPos

MLAxisPipePos

MLAxisCmdPos

MLAxisWritePipPos

Previous Function Name

MLAxisActualPos

Example

Structured Text

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

Ladder Diagram

Function Block Diagram

2.1.4.32 MLAxisReadFBUnit

Description

Get the feedback units per revolution value of the axis

Arguments

Input

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

Output

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

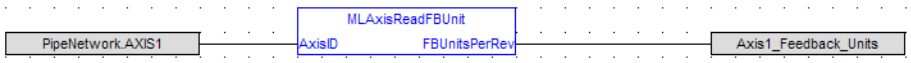
Example

Structured Text

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

Ladder Diagram

Function Block Diagram



2.1.4.33 MLAxisReadFEUU

Description

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

Arguments

Input

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

Output

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

Related Functions

MLAxisReadActPos

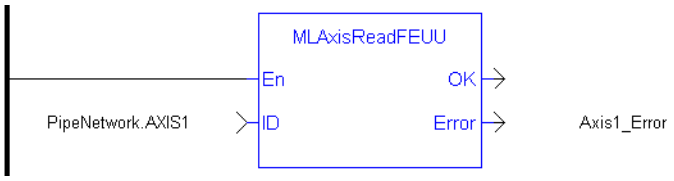
ECATGetStatus

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.4.34 MLAxisReadGenStatus

Description

Returns the status of the internal generator of the axis.

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

Arguments

Input

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

Output

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

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

Related Functions

MLAxisGenIsRdy

MLAxisStatus

Previous Function Name

MLAxisGenStatus

Example

Structured Text

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

Ladder Diagram

Function Block Diagram

2.1.4.35 MLAxisReadModPos

Description

Get the value period of the axis.

Arguments

Input

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

Output

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

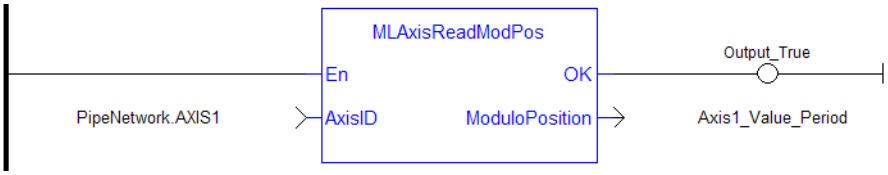
ModuloPosition	Description	Returns the Axis Value Period
	Data type	LREAL
	Unit	User unit

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.4.36 MLAxisReadTq

Description

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

Arguments

Input

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

Output

Torque	Description	Actual torque applied by the drive associated to the axis in N.m (Newton meter) If you have **not** previously invoked the "MLAxisRatedTq" (➜ p. 139) function, the Output value is **1/1000 rated motor torque** (1000.0 = rated torque)
	Data type	LREAL
	Unit	N.m (Newton meter)

Related Functions

MLAxisRatedTq

MLAxisReadActPos

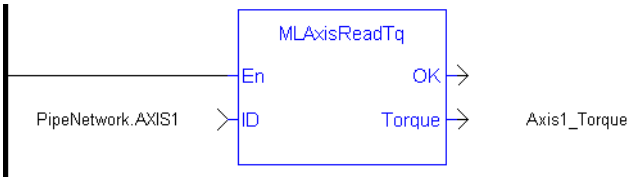
MLAxisReadVel

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.4.37 MLAxisReadUUnits

Description

Get the User units per revolution value of the axis

Arguments

Input

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

Output

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

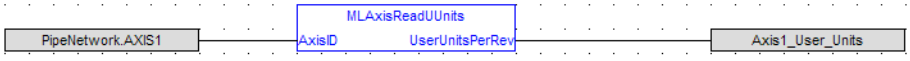
Example

Structured Text

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

Ladder Diagram

Function Block Diagram



2.1.4.38 MLAxisReadVel

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.

Arguments

Input

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

Output

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

Related Functions

MLAxisReadActPos

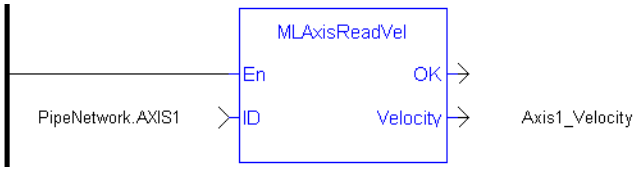
MLAxisReadTq

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.4.39 MLAxisReAlgnRdy

Description

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

Arguments

Input

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

Output

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

Related Functions

MLAxisReAlign

Example

Structured Text

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

Ladder Diagram

Function Block Diagram

2.1.4.40 MLAxisReAlign

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

Arguments

Input

ID	**Description**	ID Name of the Axis block
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—
Acceleration	**Description**	Sets the Realign Acceleration
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Deceleration	**Description**	Sets the Realign Deceleration rate
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Speed	**Description**	Sets the Axis Speed
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
DeltaPos	**Description**	Sets the Axis Delta Position, or the relative distance to be moved
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

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

Related Functions

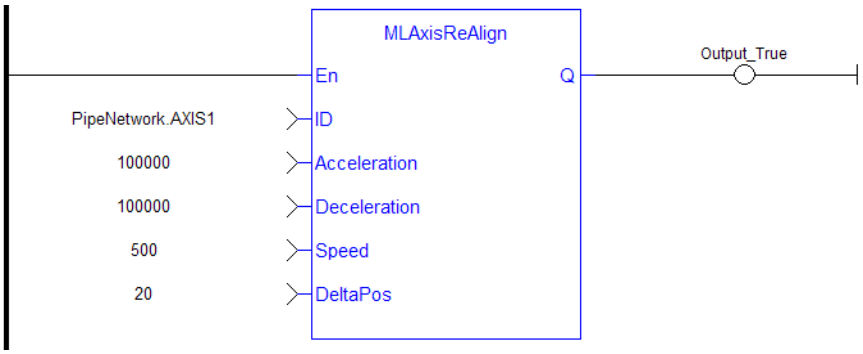
MLAxisReAlgnRdy

Example

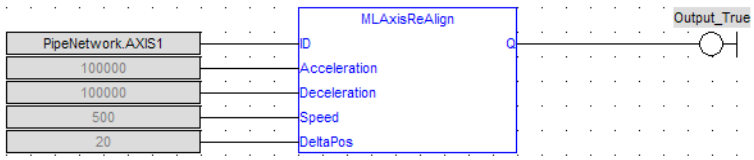
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.4.41 MLAxisRel

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 MLAxis 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 MLAxisRel can occur before moves have finished.

Arguments

Input

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

DeltaPosition	Description	Sets the Axis Delta Position, or the relative distance to be moved
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

Default (.Q)	Description	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.
	Data type	BOOL
	Unit	n/a

Related Functions

MLAxisGenWriteAcc

MLAxisGenWriteDec

MLAxisGenWriteSpd

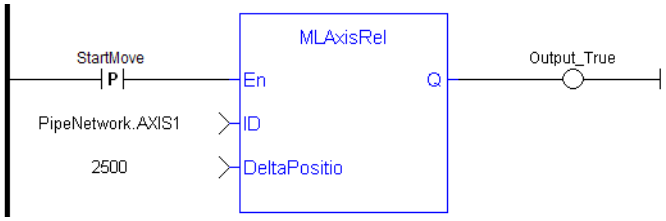
Example

See "Usage Example of Axis Functions" (➜ p. 165) for additional examples.

Structured Text

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

Ladder Diagram



NOTE

You must use a pulse contact to start the FB

Function Block Diagram

2.1.4.42 MLAxisResetErrors

Description

Clears errors of the specified axis

Arguments

Input

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

Output

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

Previous Function Name

MLAxisClrErrors

Example

Structured Text

```
MLAxisResetErrors( PipeNetwork.Axis1 ) ;
```

Ladder Diagram

Function Block Diagram

2.1.4.43 MLAxisRstFastIn

Description

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

Arguments

Input

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

InputID	Description	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 AKD Capture Engine Configuration.
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—

Output

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

Related Functions

MLAxisCfgFastIn

MLAxisIsTrigged

Example

Structured Text

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

Ladder Diagram

Function Block Diagram

2.1.4.44 MLAxisStatus

Description

Returns the status of the axis.

Arguments

Input

ID

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

Output

OK

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

Default (.Q)

Description Returns the status of the axis

Bit	Description
0	Initialized (1 if initialized)
1	Power (1 if power is on) Is linked to bit 1 (Switched on) of the Status Word 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 State Machine.
4	Configured (1 if configured) EtherCAT state is Safe-Operational, see State Machine.
5	Running (1 if running) EtherCAT state is Operational, see State Machine.
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

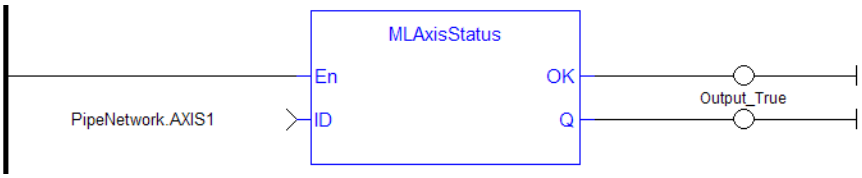
Data type	DINT
Unit	n/a

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.4.45 MLAxisStop

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.

Arguments

Input

ID	Description	ID Name of the Axis block
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—
Start	Description	
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Deceleration	Description	
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—

Output

Default (.Q)	Description	Comes true when the Axis is completely stopped.
	Data type	BOOL
	Unit	n/a
PipePos	Description	Corresponds to the Pipe Position input to the axis at the time the stop is triggered.
	Data type	LREAL
	Unit	User unit
GenPos	Description	Corresponds to the Generator Position input to the axis at the time the stop is triggered.
	Data type	LREAL
	Unit	User unit
RealignPos	Description	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.
	Data type	LREAL
	Unit	User unit
StopPos	Description	Corresponds to the last Reference Position sent to the drive at the time when the Axis is completely stopped. It is functionally different than the Actual Position because that position is the drive position converted to user units. The correct delta for the realign move to get in sync with the trajectory in order to realign the slave to the master is the current Reference Position minus the **Stop Position** for the realign move. After stopping, if the axis is disabled and the motor position is manually altered, this distance must be taken into account when performing the realign.
	Data type	LREAL
	Unit	User unit

Related Functions

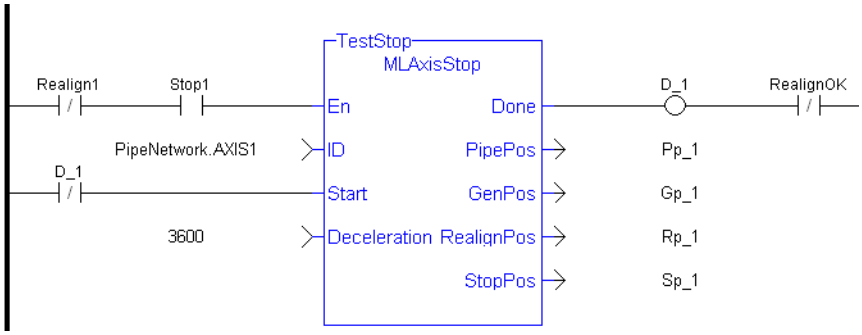
MLAxisReAlign

Example

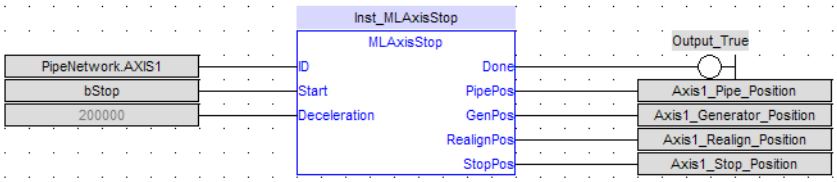
Structured Text

```
Inst_MLAxisStop(PipeNetwork.AXIS1, bStop, 200000) ;
If Inst_MLAxisStop.Done Then
StopPosition := Inst_MLAxisStop.StopPos;
End_if;
```

Ladder Diagram



Function Block Diagram



2.1.4.46 MLAxisTimeStamp

Description

Returns the timestamp of the triggered axis.

Arguments

Input

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

Output

OK	Description	Returns true when function successfully executes.
	Data type	BOOL
	Unit	n/a
Q	Description	Returns the time stamp value. This value is explained in How to interpret the timestamp.
	Data type	DINT
	Unit	microseconds

Related Functions

MLAxisCfgFastIn

MLAxisRstFastIn

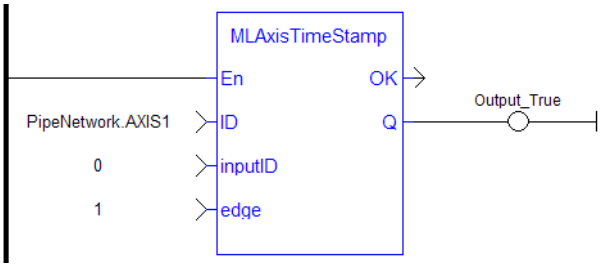
MLAxisIsTrigged

Example

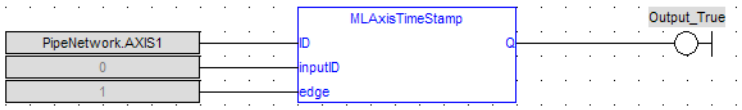
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.4.47 MLAxisWriteModPos

Description

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

Arguments

Input

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

ModuloPosition	Description	Sets the Axis Period Value when Mode is set to Modulo.
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

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

Example

Structured Text

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

Ladder Diagram

Function Block Diagram

2.1.4.48 MLAxisWritePipPos

Description

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

Arguments

Input

AxisID	Description	ID Name of the Axis block
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—
PipePosition	Description	Sets the Axis Pipe Position
	Data type	LREAL
	Range	—

	Unit	User unit
	Default	—

Output

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

Related Functions

MLAxisReadActPos

MLAxisFBackPos

MLAxisGenPos

MLAxisPipePos

MLAxisCmdPos

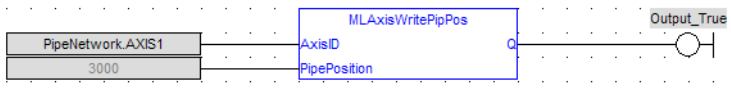
Example

Structured Text

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

Ladder Diagram

Function Block Diagram



2.1.4.49 MLAxisWritePos

Description

Used to set a position offset at the Axis when the Pipe Network is not yet connected.

- 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

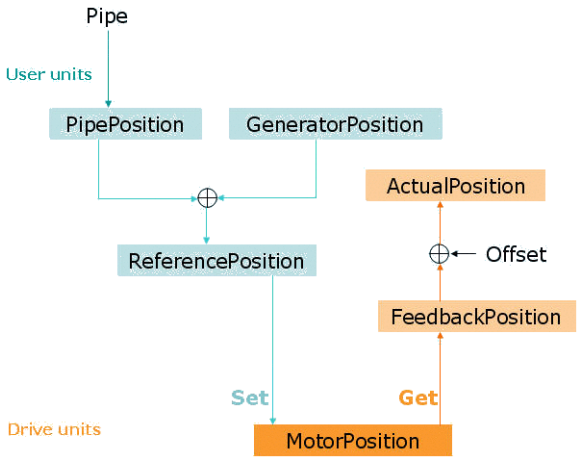
About associated data on Positions

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
ActualPosition	**Actual** 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 **Power ON Delta Offset**, and any zero-offset due to an MLWritePos function ("MLAxisWritePipPos" (➜ p. 160), "MLAxisWritePos" (➜ p. 161)). Normally the value of power on delta offset is zero. `ActualPos := FeedbackPos + ZeroOffset`
CurrentPosition	**Current** 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. `CurrentPos := ReferencePosition + ZeroOffset`
FeedbackPosition	**Feedback** Position is the “Position actual value” read from the drive. FeedbackPos relates to the TxPDO value of 'Actual position value'
GeneratorPosition	**Generator** 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 "MLAxisWritePos" (➜ p. 161)() being used to modify actual position via the zero offset value and the adjustment in commanded value to insure no steps in the **Current** 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.
MotorPosition	**Motor** position relates to the RxPDO value of 'Position demand value' `MotorPosition = CurrentPos + PowerOnDeltaOffset`
PipePosition	The output of the convertor block is written into the **PipePosition** value whenever the convertor block is connected to the axis and the pipe is active.
Power ON Delta Offset	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 “**ActualPosition** value” and the “Position demand value” last sent to the drive. It is then added to the **Current** position value when the “Position demand value” is updated. It is read in User Units without periodicity applied.
ReferencePosition	**Reference** position is the summation of **PipePosition** and **GeneratorPosition**. `ReferencePosition = Pipe Position + Generator Position`
Zero Offset	Affected by the "MLAxisWritePos" (➜ p. 161)() 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.



Arguments

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	—

Output

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

Previous Function Name

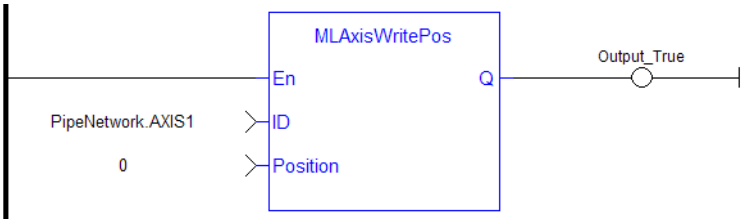
MLAxisSetZero

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.4.50 MLAxisWriteUUnits

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^2]

Arguments

Input

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

Output

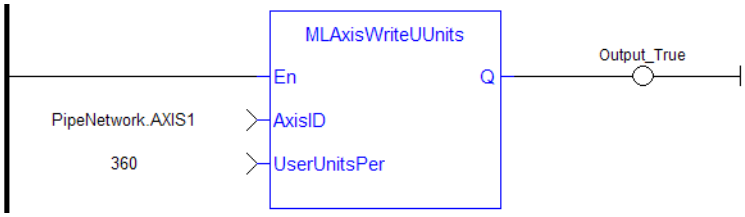
Default (.Q)	Description	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a
UserUnitsPerRev	Description	Sets the Axis User Units per revolution
	Data type	LREAL
	Unit	n/a

Example

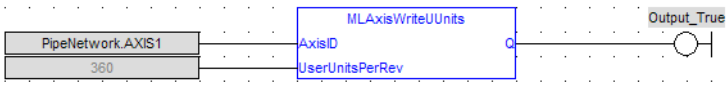
Structured Text

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

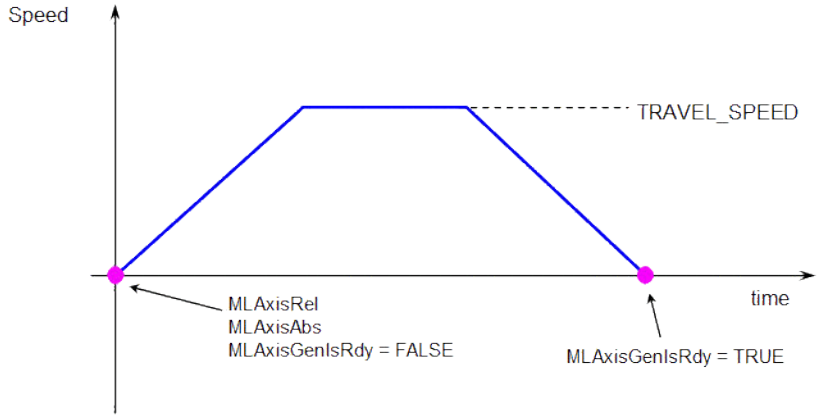
Ladder Diagram



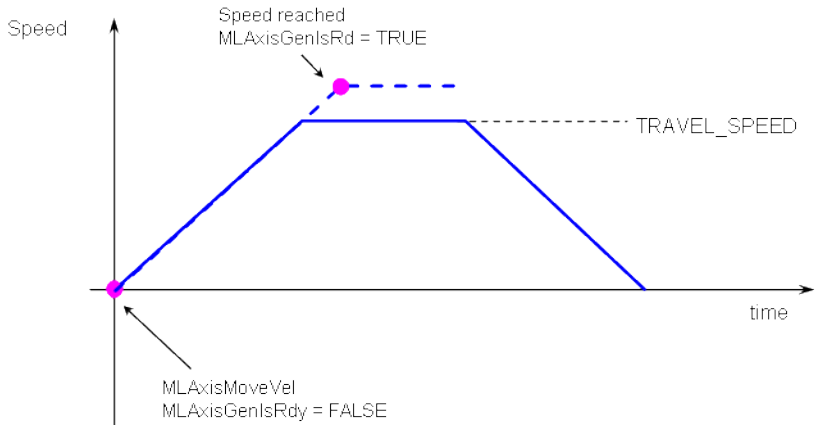
Function Block Diagram



2.1.4.51 Usage Example of Axis Functions



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



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

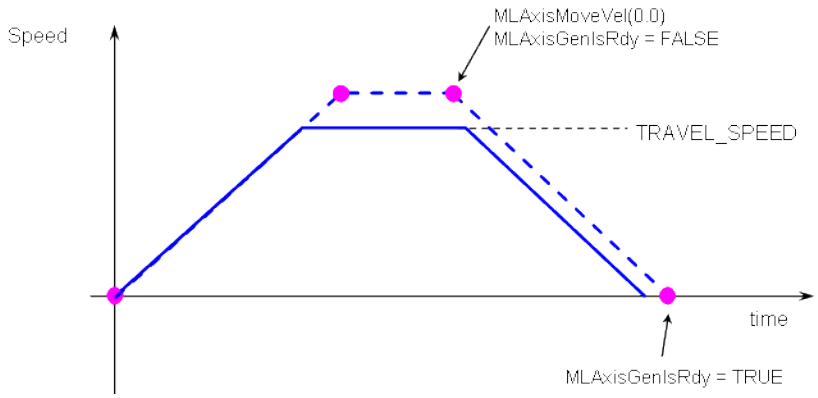


Figure 1-19: Axis Functions Usage

2.1.5 Motion Library - Cam Profile

Name	Description	Return type
"MLCamInit" (➜ p. 166)	Initializes a cam Pipe Block with user-defined settings	BOOL
"MLCamSwitch" (➜ p. 168)	Switches profiles of the selected cam object	BOOL
"MLPrfReadIOffset" (➜ p. 169)	Returns the Input Offset value of a selected cam profile	None
"MLPrfReadIScale" (➜ p. 171)	Returns the Input Ratio value of a selected cam profile	None
"MLPrfReadOOffset" (➜ p. 172)	Returns the Output Offset value of a selected cam profile	None
"MLPrfReadOScale" (➜ p. 173)	Returns the Output Ratio value of a selected cam profile	None
"MLPrfWriteIOffset" (➜ p. 174)	Sets the Input Offset value of a selected cam profile	BOOL
"MLPrfWriteIScale" (➜ p. 176)	Sets the Input Ratio value of a selected cam profile	BOOL
"MLPrfWriteOOffset" (➜ p. 177)	Sets the Output Offset value of a selected cam profile	BOOL
"MLPrfWriteOScale" (➜ p. 179)	Sets the Output Ratio value of a selected cam profile	BOOL
"MLProfileBuild" (➜ p. 413)	Builds a cam profile from application data	See "Output" (➜ p. 415)
"MLProfileCreate" (➜ p. 418)	Creates a new cam profile object	None
"MLProfileInit" (➜ p. 419)	Initializes a previously created cam profile object	BOOL
"MLProfileRelease" (➜ p. 421)	Removes a Profile so the Profile ID may be used by a different or new Profile.	See "Output" (➜ p. 422)

2.1.5.1 MLCamInit

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.

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

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.

Arguments

Input

BlockID	**Description**	ID number of a created Pipe Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	CAM
ProfileName	**Description**	Name of the current profile assigned to the cam. It must be a declared profile object
	Data type	STRING
	Range	—
	Unit	n/a
	Default	—
ModuloPosition	**Description**	Value of the period of the cam output values expressed in user units, for a cyclic system
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	360.0

Output

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

Return Type

BOOL

Related Functions

"MLProfileCreate" (➜ p. 418)

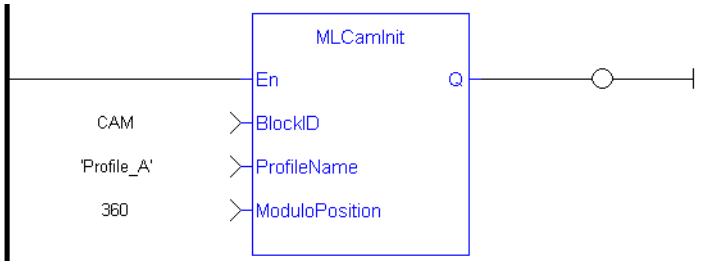
"MLProfileInit" (➜ p. 419)

Example

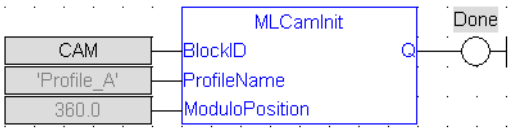
Structured Text

```
//Create and initialize a CAM Object
CAM := MLBlkCreate( 'CAM', 'CAM' );
MLCamInit( CAM, 'Profile_A', 360.0 );
```

Ladder Diagram



Function Block Diagram



2.1.5.2 MLCamSwitch

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.

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

"MLProfileCreate" (➜ p. 418)

"MLProfileInit" (➜ p. 419)

"MLPrfWriteIOffset" (➜ p. 174)

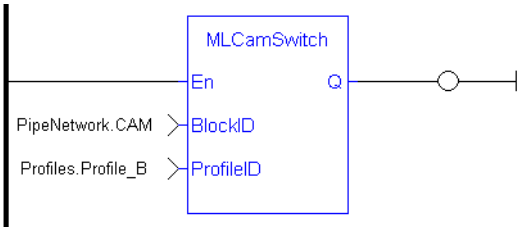
"MLPrfWriteOScale" (➜ p. 179)

Example

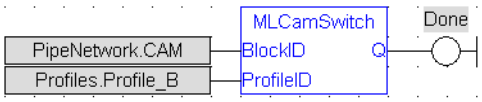
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.5.3 MLPrfReadIOffset

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.

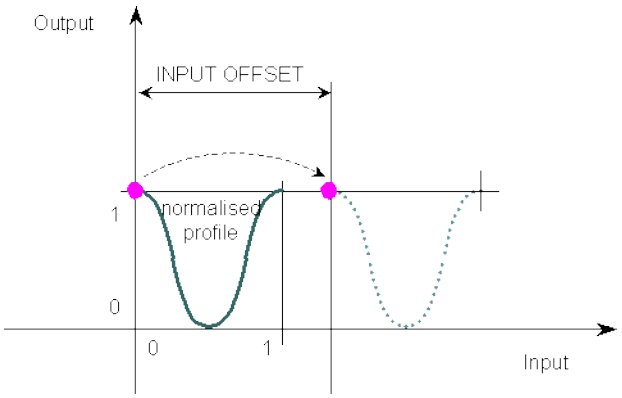


Figure 1-20: MLPrfReadIOffset

Arguments

Input

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

Output

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

Related Functions

"MLPrfWritelOffset" (➜ p. 174)

"MLProfileCreate" (➜ p. 418)

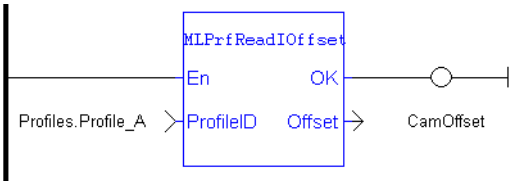
"MLProfileInit" (➜ p. 419)

Example

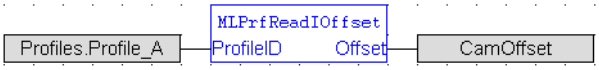
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.5.4 MLPrfReadIScale

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.

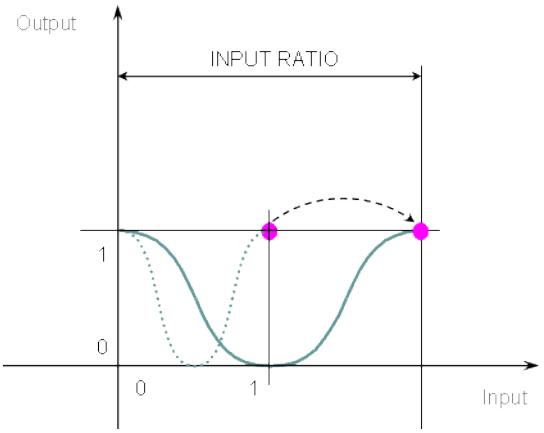


Figure 1-21: MLPrfReadIScale

Arguments

Input

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

Output

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

Related Functions

"MLPrfWriteIScale" (➜ p. 176)

"MLProfileCreate" (➜ p. 418)

"MLProfileInit" (➜ p. 419)

Previous Function Name

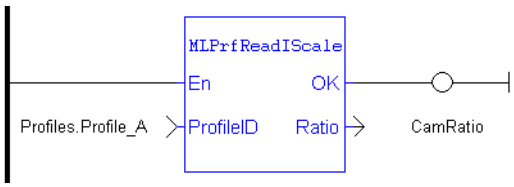
MLPrfGetIRatio

Example

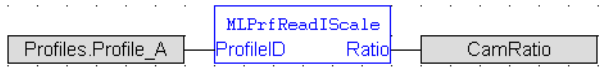
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.5.5 MLPrfReadOOffset

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.

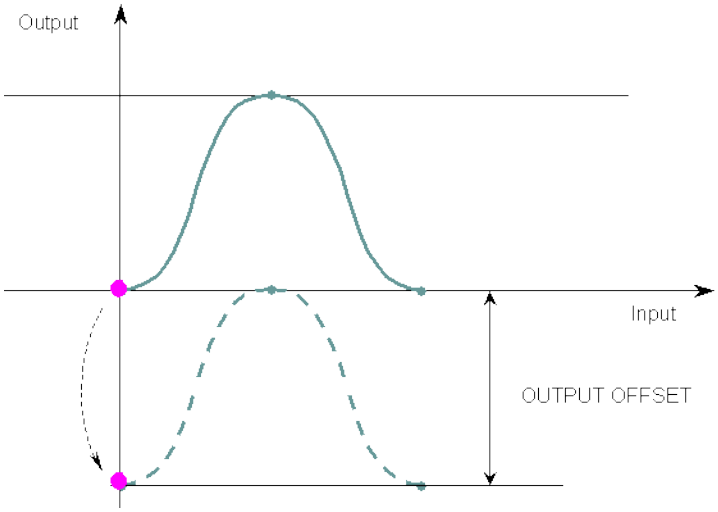


Figure 1-22: MLPrfReadOOffset

Arguments

Input

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

Output

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

Related Functions

"MLPrfWriteOOffset" (➜ p. 177)

"MLProfileCreate" (➜ p. 418)

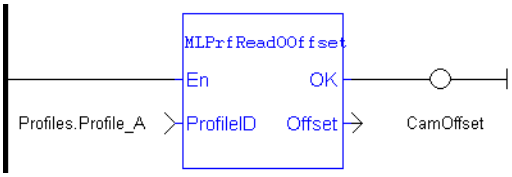
"MLProfileInit" (➜ p. 419)

Example

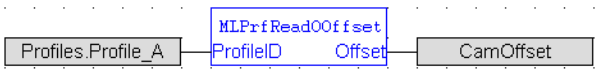
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.5.6 MLPrfReadOScale

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.

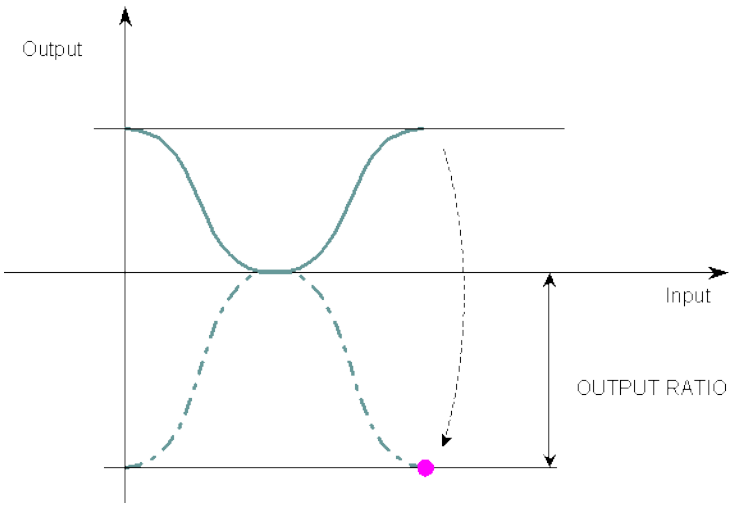


Figure 1-23: MLPrfReadOScale

Arguments

Input

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

Output

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

Related Functions

"MLPrfWriteOScale" (➜ p. 179)

"MLProfileCreate" (➜ p. 418)

"MLProfileInit" (➜ p. 419)

Previous Function Name

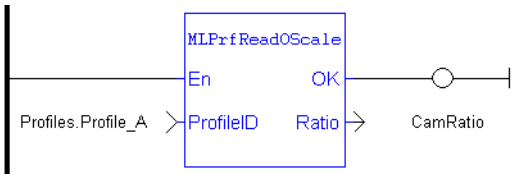
MLPrfGetORatio

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.5.7 MLPrfWritelOffset

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.

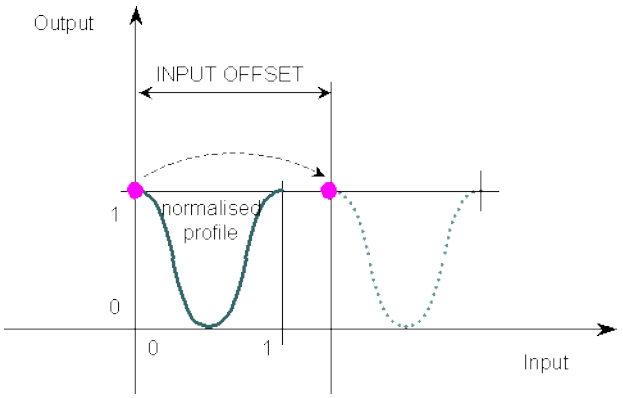


Figure 1-24: MLPrfWriteIOffset

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

"MLPrfReadIOffset" (➜ p. 169)

"MLProfileCreate" (➜ p. 418)

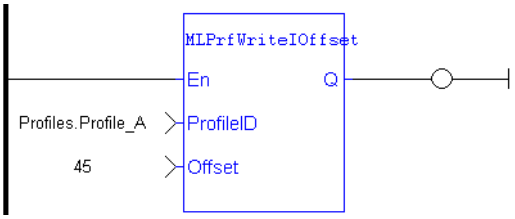
"MLProfileInit" (➜ p. 419)

Example

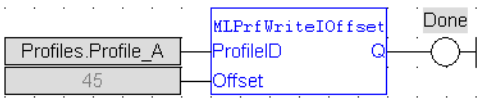
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.5.8 MLPrfWriteIScale

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.

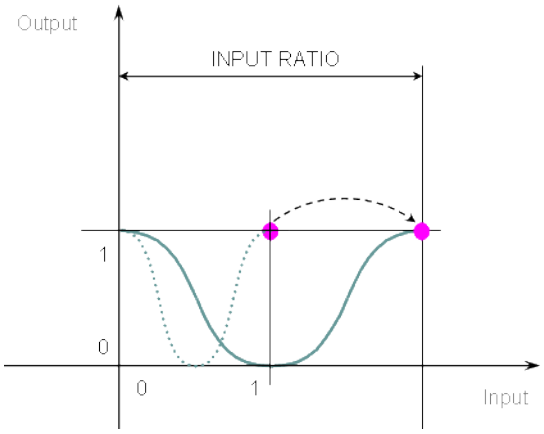


Figure 1-25: MLPrfWriteIScale

Arguments

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	—

Output

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

Return Type

BOOL

Related Functions

"MLPrfReadIScale" (➜ p. 171)

"MLProfileCreate" (➜ p. 418)

"MLProfileInit" (➜ p. 419)

Previous Function Name

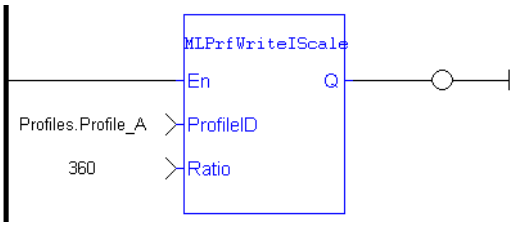
MLPrfSetIRatio

Example

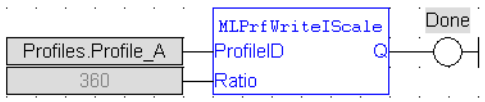
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.5.9 MLPrfWriteOOffset

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.

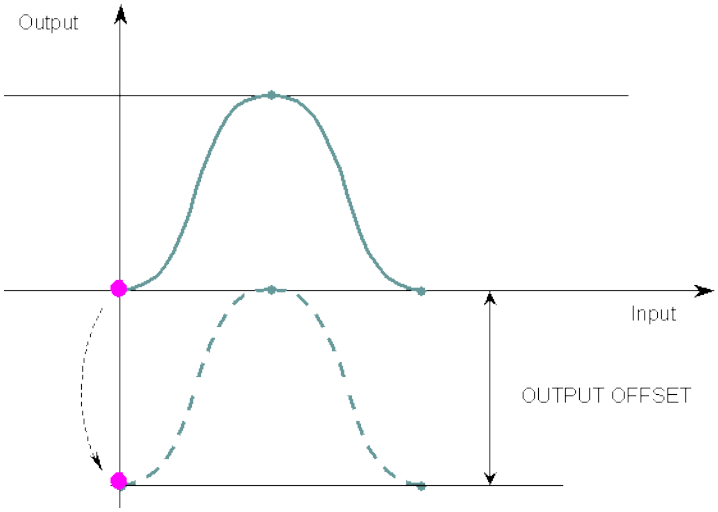


Figure 1-26: MLPrfWriteOOffset

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

"MLPrfReadOOffset" (➜ p. 172)

"MLProfileCreate" (➜ p. 418)

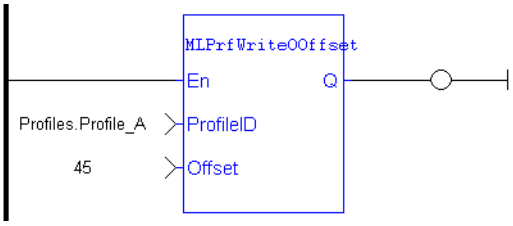
"MLProfileInit" (➜ p. 419)

Example

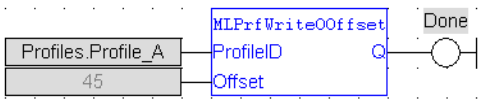
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.5.10 MLPrfWriteOScale

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.

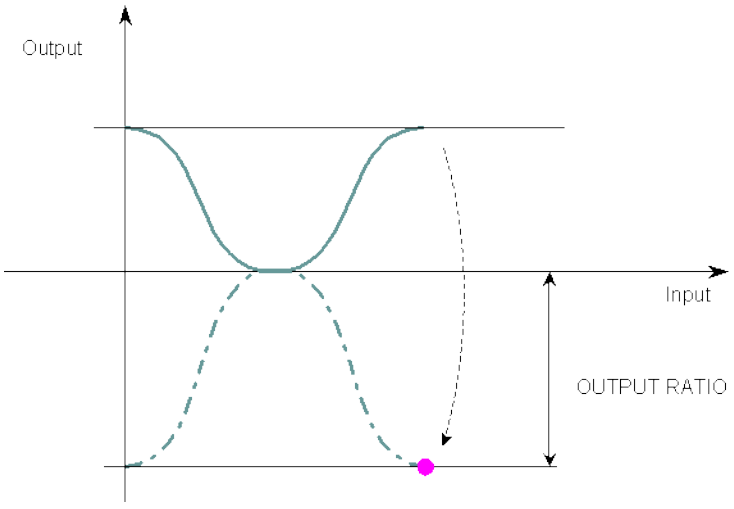


Figure 1-27: MLPrfWriteOScale

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

"MLPrfReadOScale" (➜ p. 173)

"MLProfileCreate" (➜ p. 418)

"MLProfileInit" (➜ p. 419)

Previous Function Name

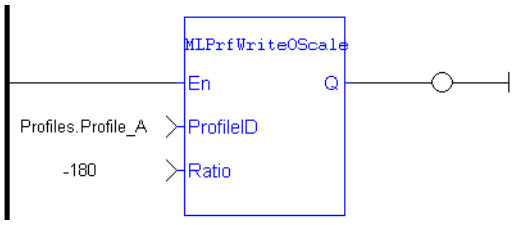
MLPrfSetORatio

Example

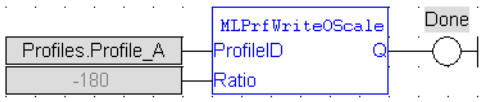
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.6 Motion Library - Comparator

TIP

For usage example about Comparator Functions, "" (➜ p. 188)

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

2.1.6.1 MLCompCheck

Description

Check if the reference of a comparator Pipe Block has been crossed. Returns the Transition Flag of a comparator object, which turns TRUE if the input position to the comparator is greater or equal to the reference. The Comparator Transition Flag stays TRUE until it is reset.

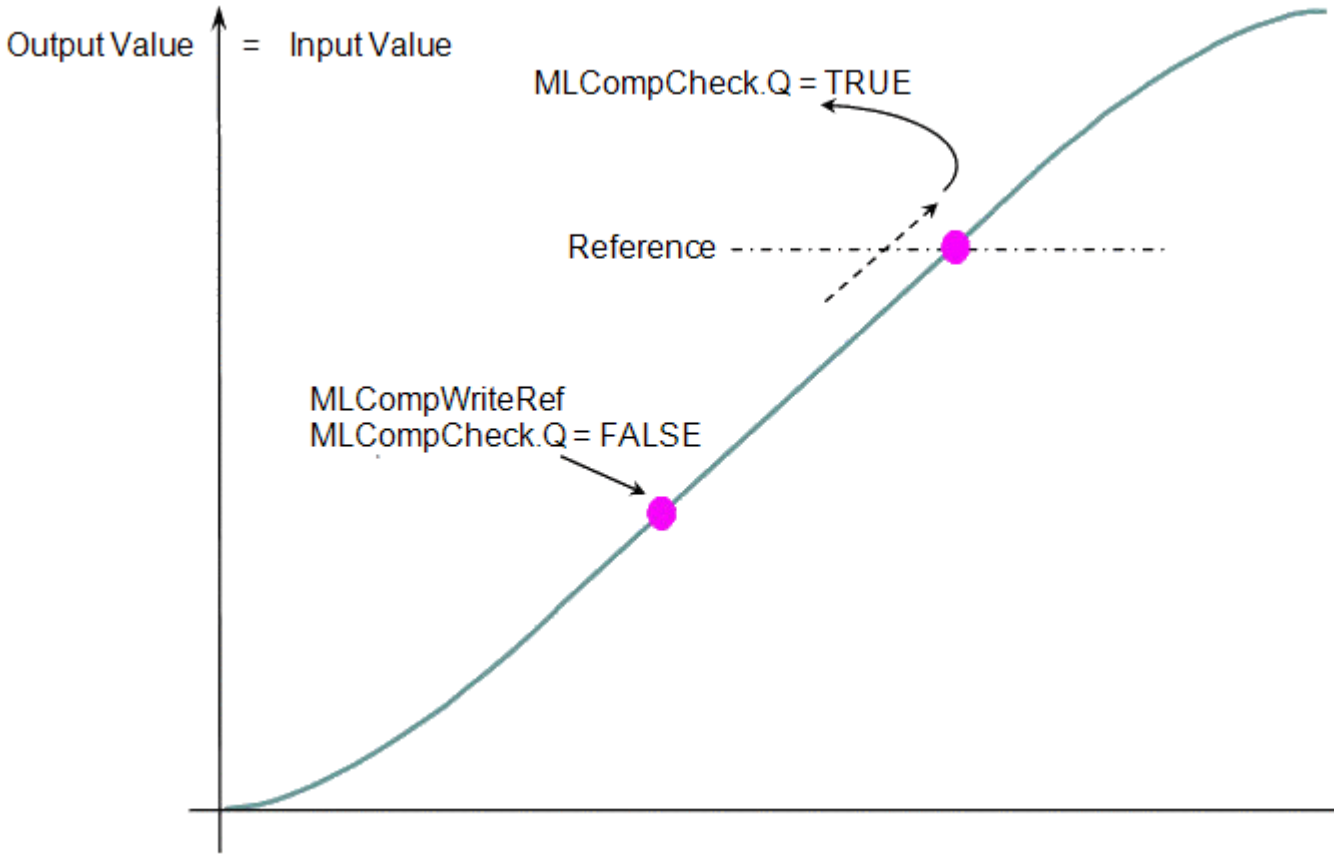


Figure 1-28: MLCompCheck

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLCompReset

MLCompWriteRef

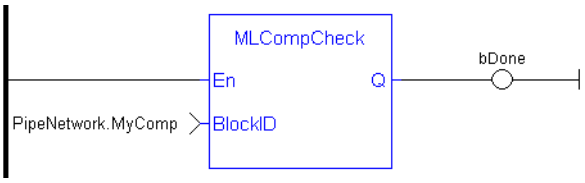
MLCompReadRef

Example

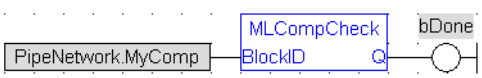
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.6.2 MLCompInit

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

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLBlkCreate

MLCompCheck

MLCompReset

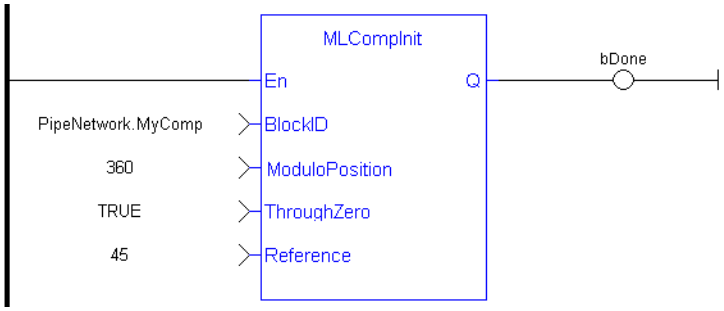
MLCompWriteRef

Example

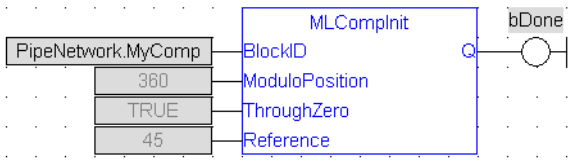
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.6.3 MLCompReadRef

Description

Returns the reference position of a comparator block. The Transition Flag of a comparator object turns TRUE if the input position to the comparator is greater or equal to the reference. The Comparator Transition Flag stays TRUE until it is reset.

Arguments

Input

BlockID	Description	ID number of an initiated Comparator object
	Data type	DINT
	Range	[-2147483648, 2147483648]

	Unit	n/a
	Default	—

Output

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

Related Functions

MLCompWriteRef

MLCompReset

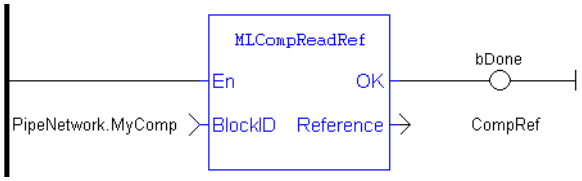
MLCompCheck

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.6.4 MLCompReset

Description

Clear the Transition Flag of a comparator Pipe Block. The Transition Flag of a comparator object turns TRUE if the input position to the comparator is greater or equal to the reference. The Comparator Transition Flag stays TRUE until it is reset.

Arguments

Input

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

	Default	—

Output

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

Return Type

BOOL

Related Functions

MLCompCheck

MLCompReadRef

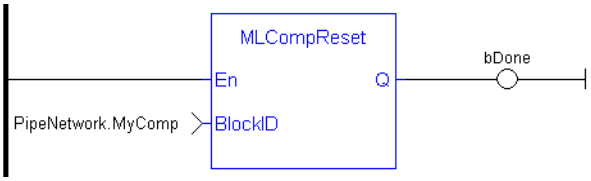
MLCompWriteRef

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.6.5 MLCompWriteRef

Description

Set the reference position of a comparator block. The Transition Flag of a comparator object turns TRUE if the input position to the comparator is greater or equal to the reference. The Comparator Transition Flag stays TRUE until it is reset.

If the input ThroughZero is set to TRUE, system must cross zero and then the reference position before the Transition Flag is set. If ThroughZero is FALSE, Transition Flag is set immediately if the input pipe position is greater then or equal to the Reference value.

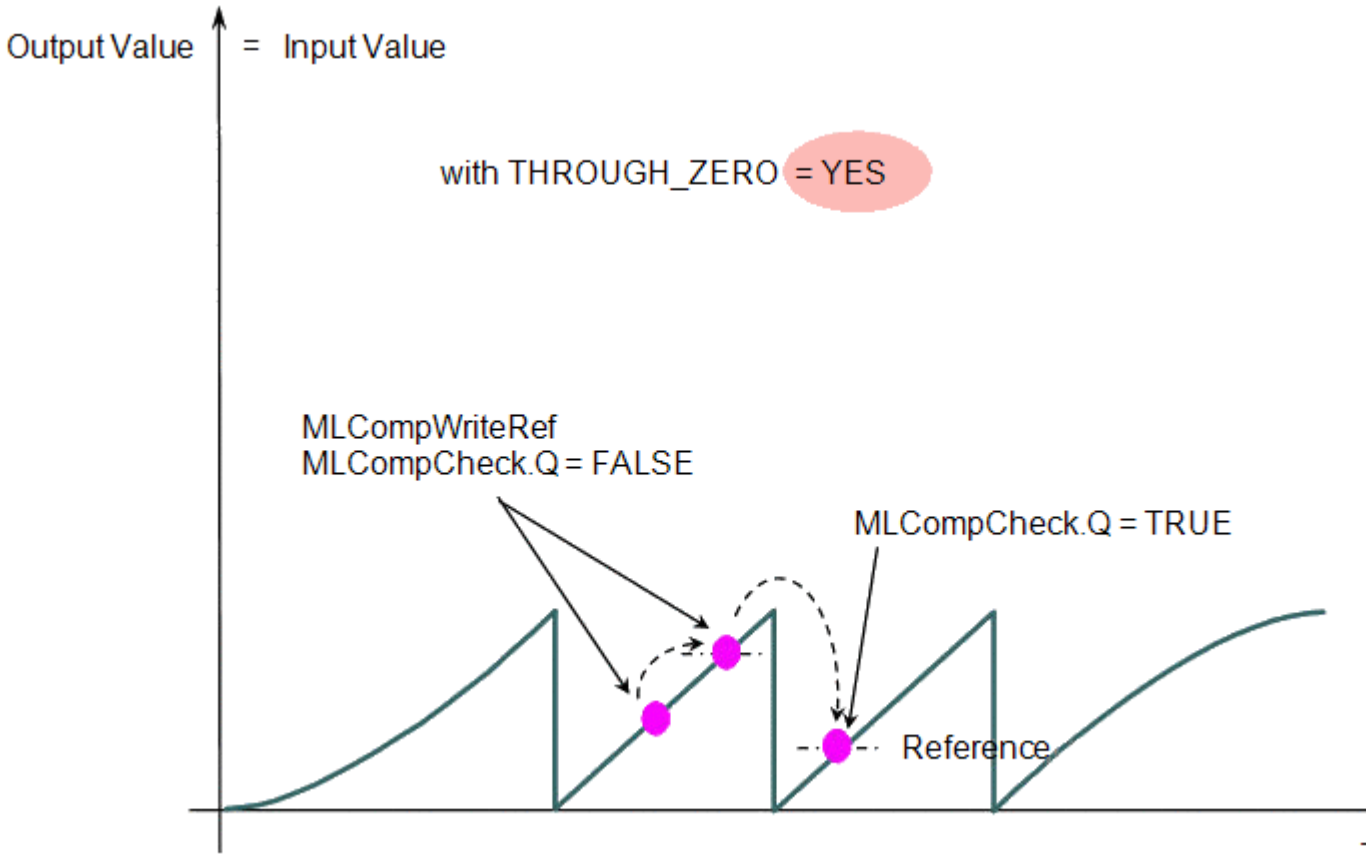


Figure 1-29: MLCompWriteRef

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLCompCheck

MLCompReadRef

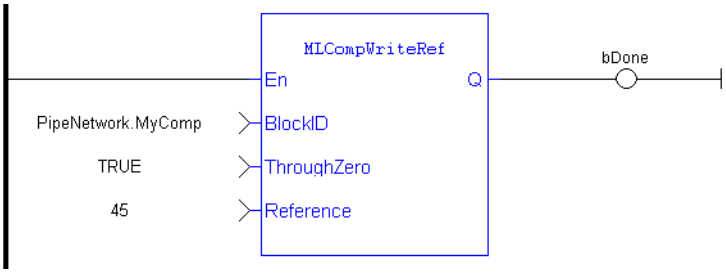
MLCompReset

Example

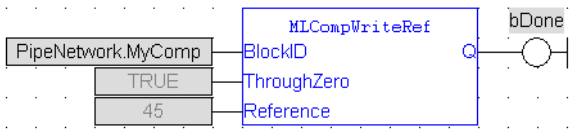
Structured Text

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

Ladder Diagram

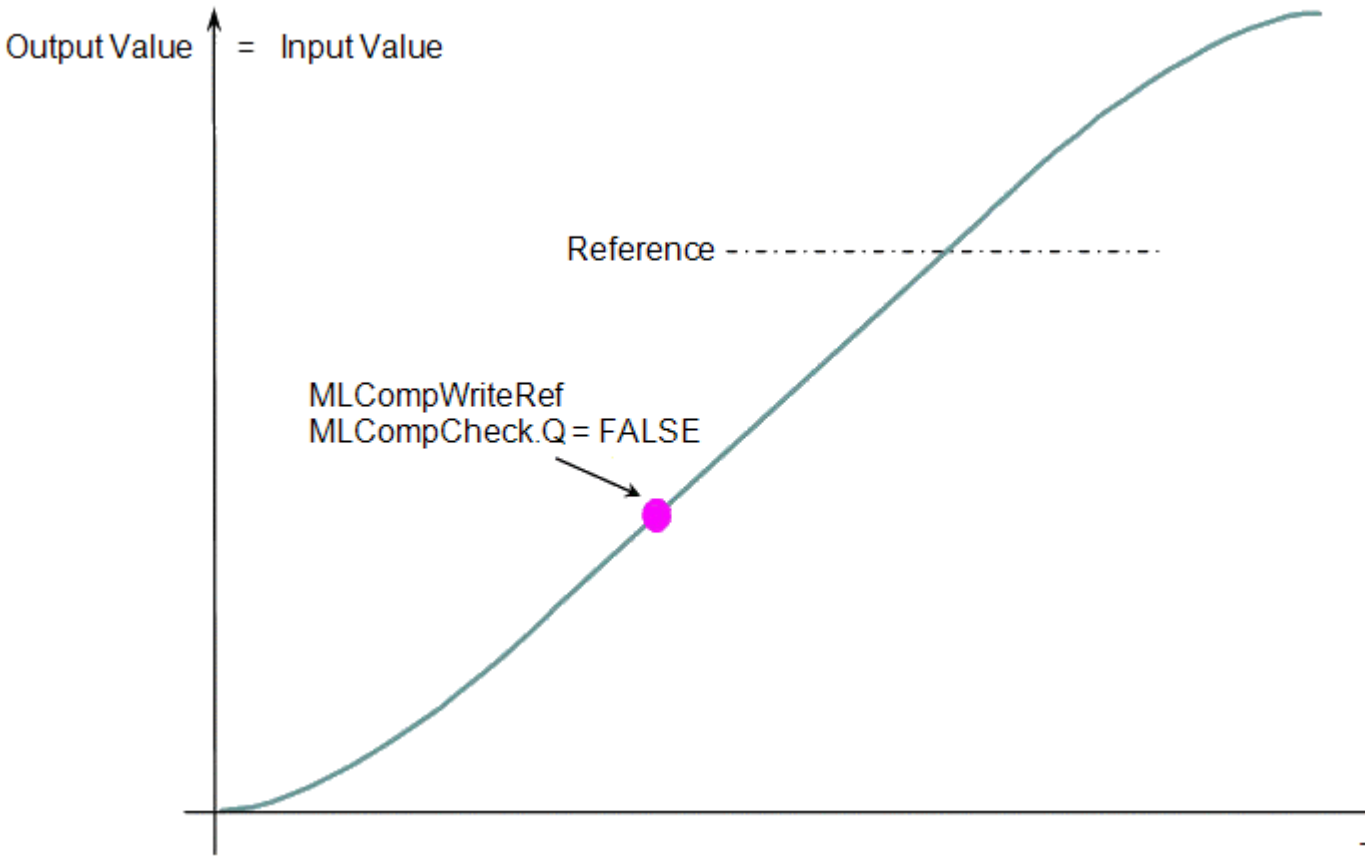


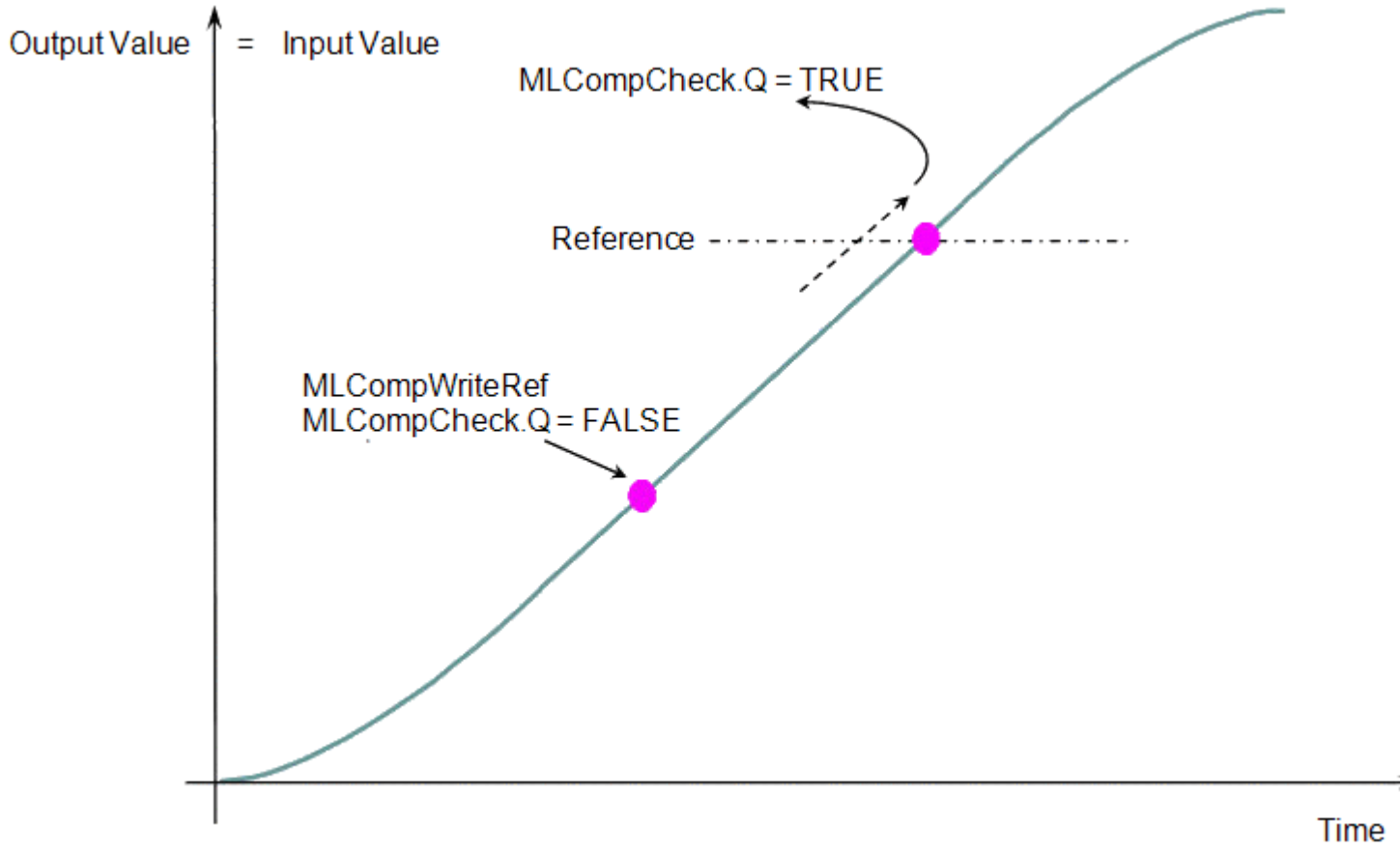
Function Block Diagram



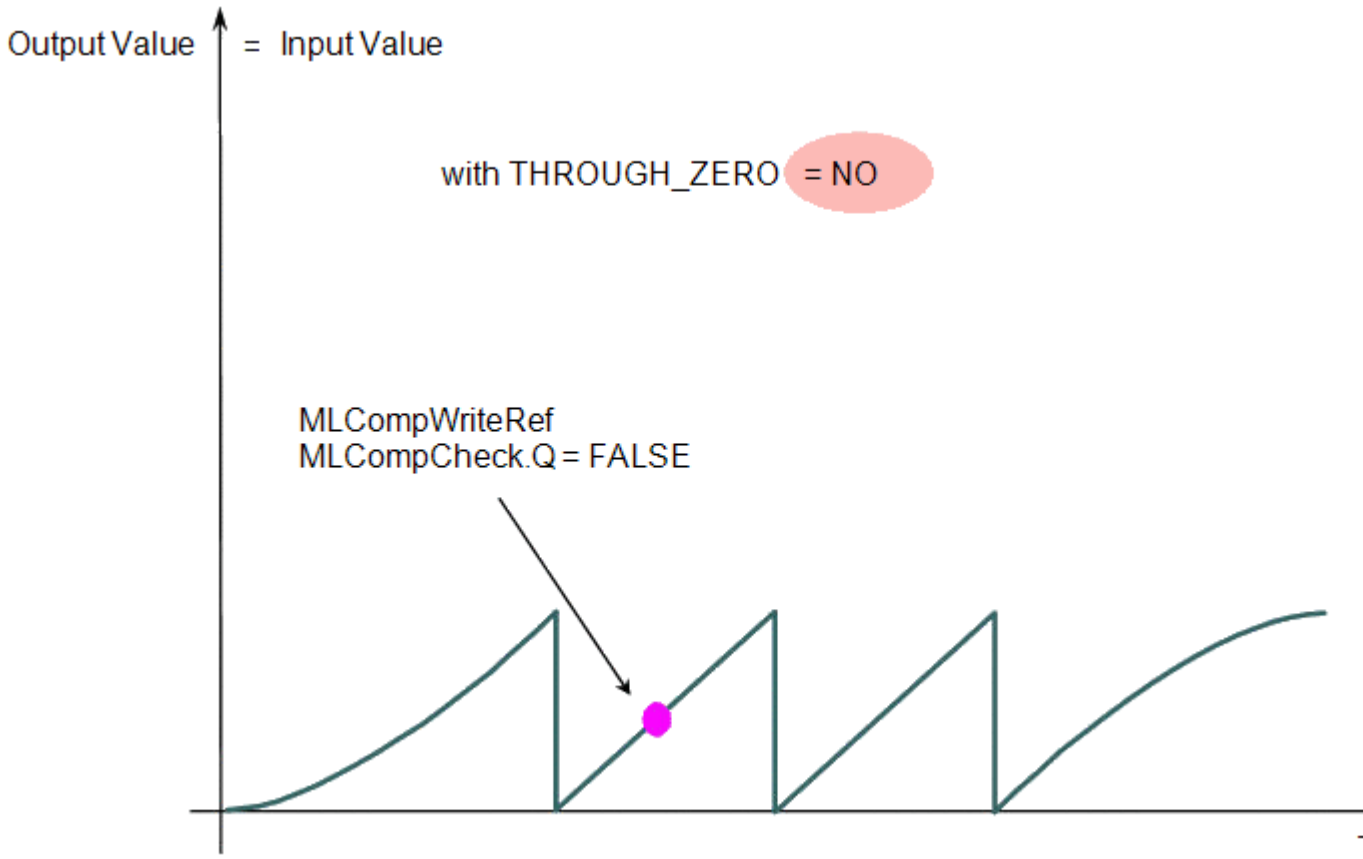
2.1.6.6 Usage example of Comparator Functions

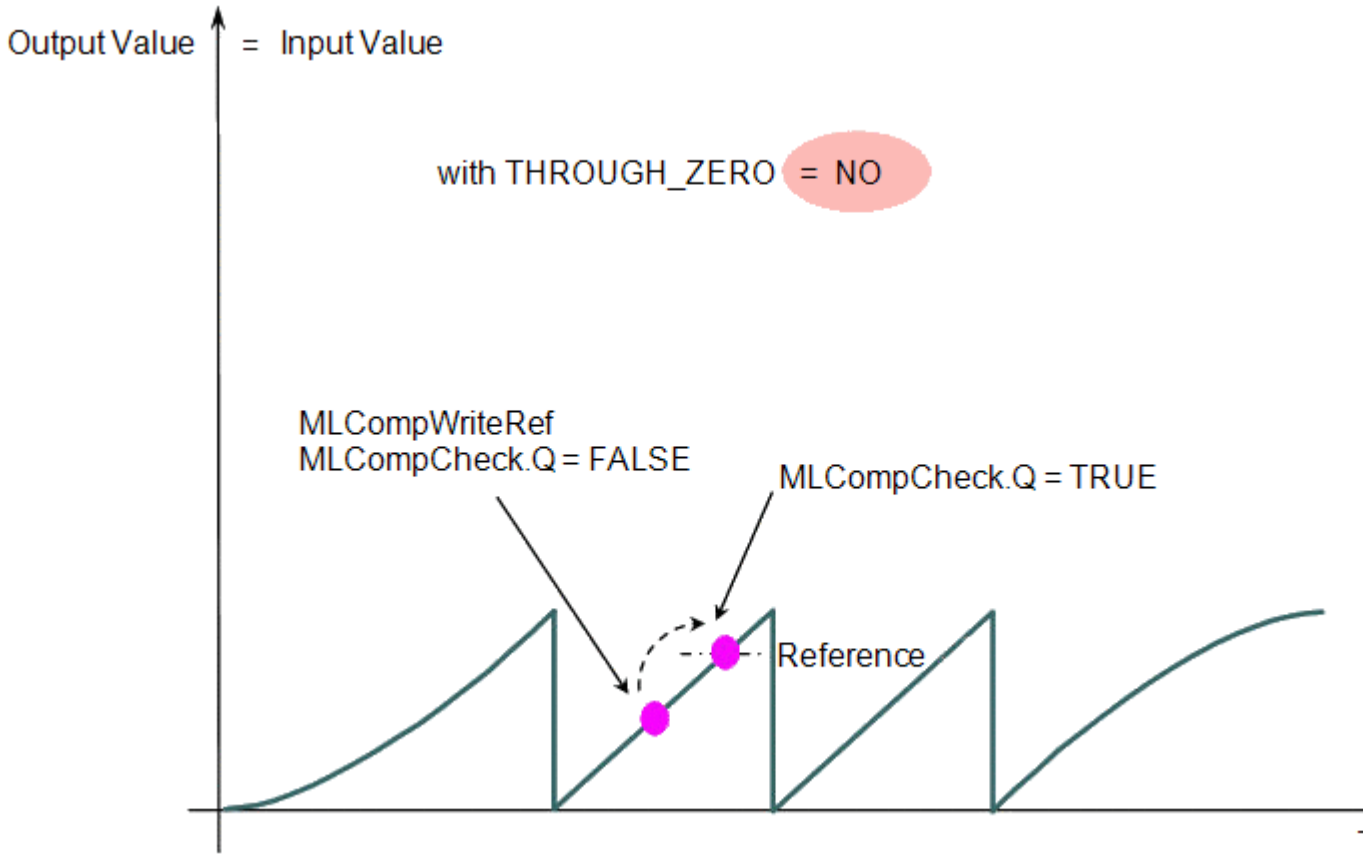
When you call the **MLCompWriteRef** function, the output for MLCompCheck becomes True as soon as the input value reaches the reference.


Output Value
=
Input Value
Reference
MLCompWriteRef
MLCompCheck.Q = FALSE

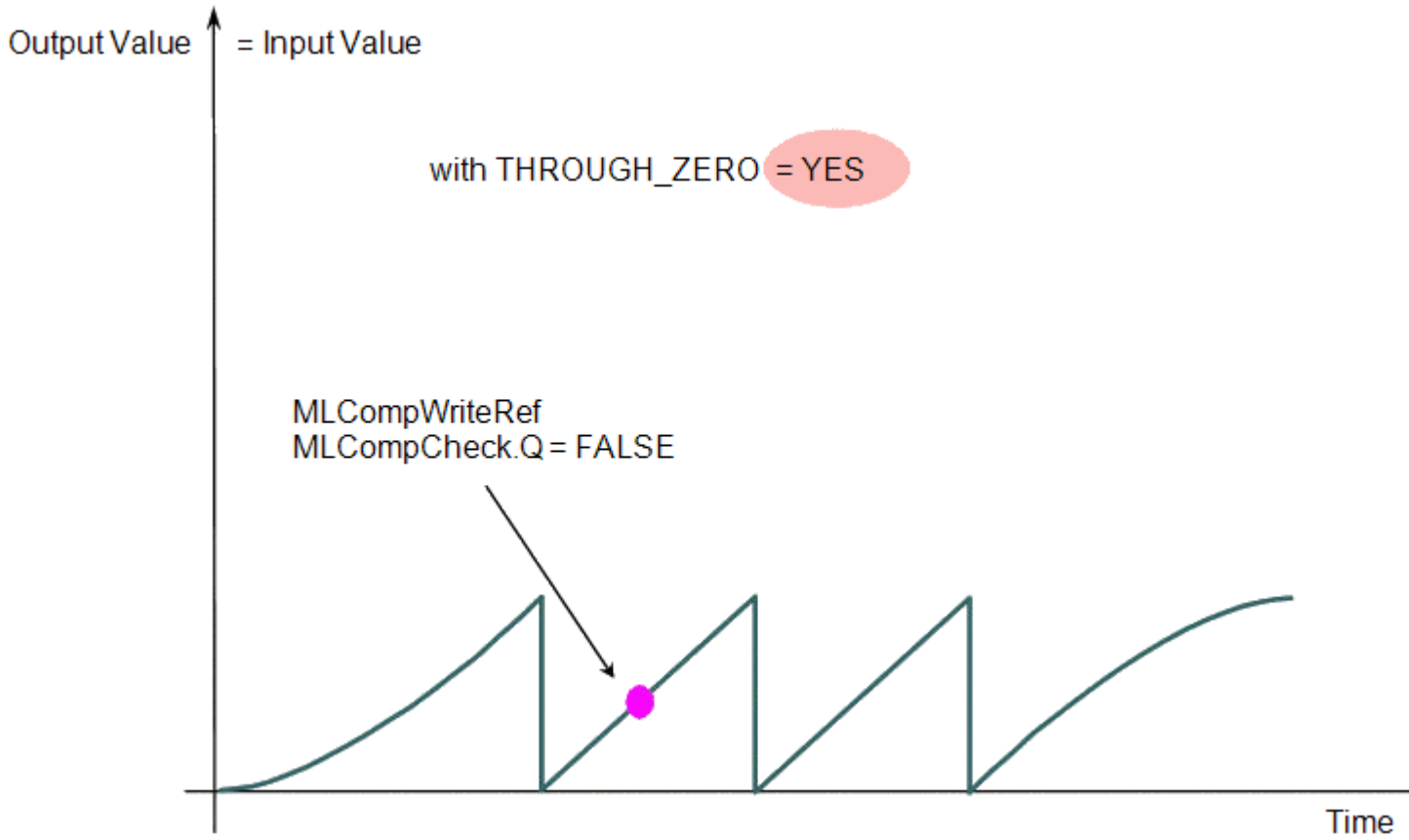


The same function can also be called for a cyclic input value.


Output Value
=
Input Value
with THROUGH_ZERO = NO
MLCompWriteRef
MLCompCheck.Q = FALSE


Output Value
=
Input Value
with THROUGH_ZERO = NO
MLCompWriteRef
MLCompCheck.Q = FALSE
MLCompCheck.Q = TRUE
Reference

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.



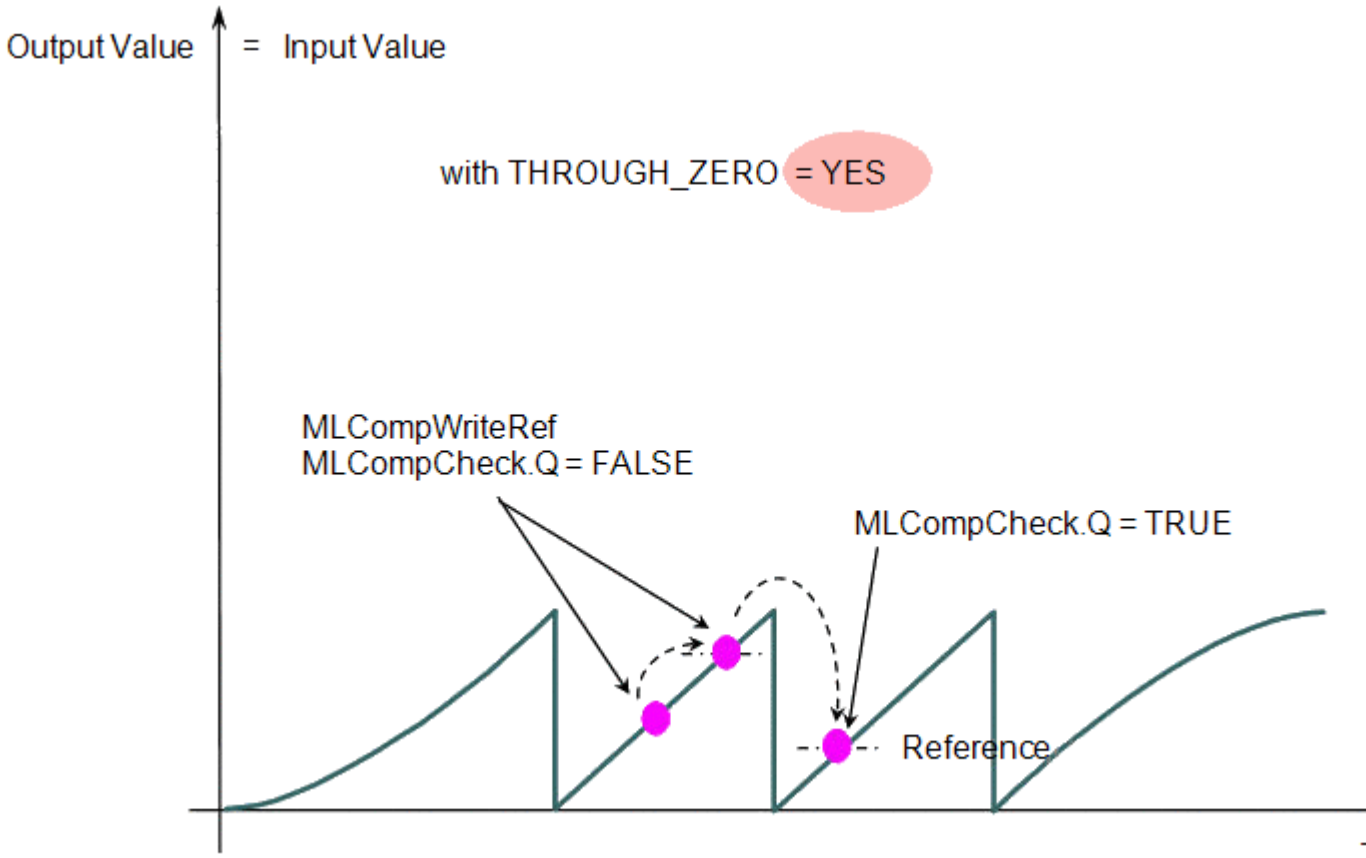


Figure 1-30: Comparator Functions Usage

2.1.7 Motion Library - Convertor

Name	Description	Return type
"MLCNVConnect" (➜ p. 193)	Connects a converter Pipe Block to the specified axis	BOOL
"MLCNVConnectEx" (➜ p. 195)	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. 199)	Disconnects a converter Pipe Block from its associated axis	BOOL
"MLCNVInit" (➜ p. 200)	Initializes a converter Pipe Block in Position or Speed mode	BOOL

2.1.7.1 MLCNVConnect

Description

Connect a converter Pipe Block to the specified axis. When using the Pipe Network for coordinated motion, Pipe Blocks have to be Activated, Connected, and then Powered On before move commands work.

The Converter block changes the incoming flow of values to continuous position output with no periodicity. If a converter block is not connected to an Axis, it does not send position output values to its assigned Axis. Every pipe branch must end in a converter, whether or not it is connected to a destination Axis object, as seen in Figure 1 below.

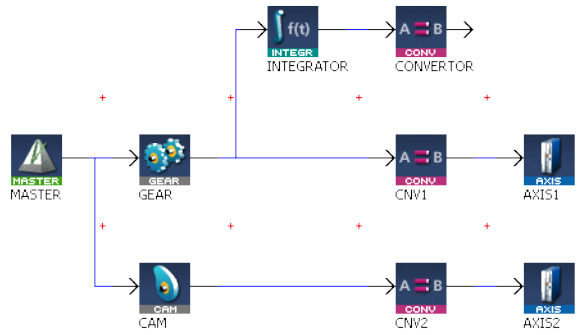


Figure 1-31: MLCNVConnect

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.

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

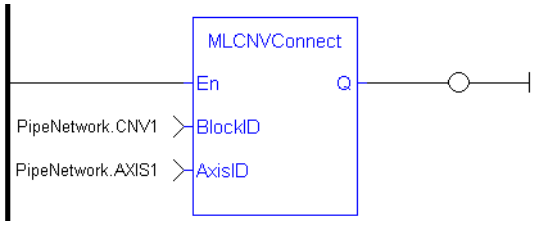
"MLCNVConnectEx" (➜ p. 195)
"MLCNVDisconnect" (➜ p. 199)
"MLCNVInit" (➜ p. 200)

Example

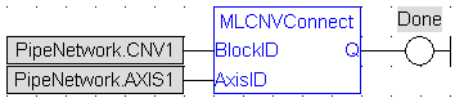
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.7.2 MLCNVConnectEx

Description

Connect a converter Pipe Block to the specified axis. This function connects the output of a pipe to an axis data other than the control position. With this function, several converter Pipe Blocks can connect to the same axis and acts on different data.

Normally a Converter block sends position values to an Axis. However, some cases exist that require additional information such as torque feed-forward (IDN 3056) that needs to be provided by a second converter.

NOTE

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

NOTE

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

Arguments

Input

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

	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
ValueID	**Description**	Specify the following constant: • EC_ADDITIVE_TORQUE_VALUE (for torque feed-forward) • EC_ANALOG_OUTPUT (for control of Analog Output: AKD parameter: "AOUT.VALUEU") If the Analog Output is mapped to a PLC variable, the connection to the analog output by EC_ANALOG_OUTPUT will not work as the output value will be overwritten by the PLC mapped variable data. In order to function properly the AOUT.MODE must be set to "User Mode (mode = 0)". See the TIP below for more information.
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
ValueInfo	**Description**	This value is ignored and must be set to **zero**
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

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.

Output

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

Return Type

BOOL

Related Functions

"MLCNVConnect" (➜ p. 193)

"MLCNVDisconnect" (➜ p. 199)

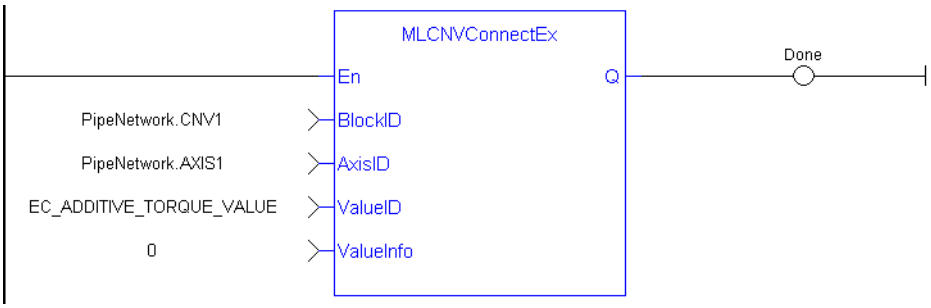
"MLCNVInit" (➜ p. 200)

Example

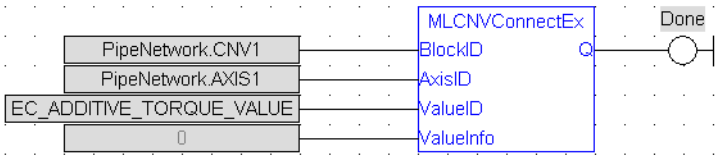
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.7.3 MLCNVConECAT

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.

Arguments

Input

BlockID	**Description**	The convertor block whose output value will be written to the PDO object. For example: PipeNetwork:CNV1
	Data type	DINT
	Range	n/a
	Unit	n/a
	Default	—
DeviceAddr	**Description**	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.
	Data type	INT

	Range	n/a
	Unit	n/a
	Default	—
Index	**Description**	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.)
	Data type	UINT
	Range	n/a
	Unit	n/a
	Default	—
SubIndex	**Description**	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.)
	Data type	USINT
	Range	n/a
	Unit	n/a
	Default	—

Output

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

Related Functions

"MLCNVDisconnect" (➜ p. 199)

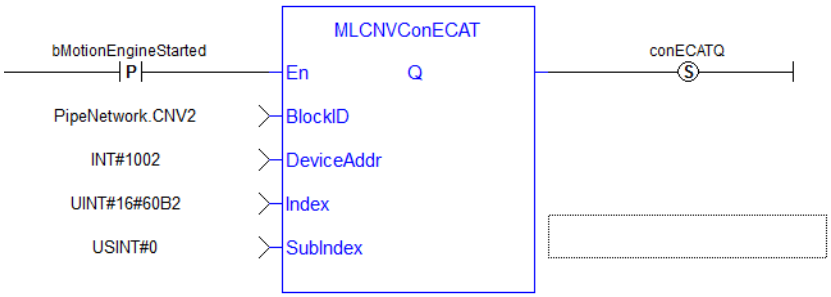
"MLCNVInit" (➜ p. 200)

Example

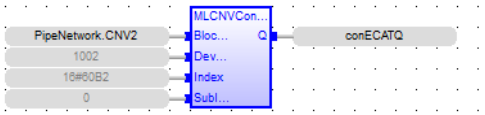
Structured Text

```
//Connect a converter Pipe Block to Axis1
MLCNVConECAT( PipeNetwork.CNV2(*DINT*), 1002(*INT*), 16#60B2(*UINT), 0
(*USINT*) );
```

Ladder Diagram



Function Block Diagram



2.1.7.4 MLCNVDisconnect

Description

Disconnect a converter Pipe Block from its associated axis.

If a converter block is not connected to an Axis, it does not send position output values to its assigned Axis. Can disconnect one or multiple Axis from the Pipe Network and still send single-axis motion commands. Axis can be disconnected while the Pipe Positions are reset to different values or if coordinated motion is only not needed with every axis in the project in a certain state.

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

"MLCNVConnect" (➜ p. 193)

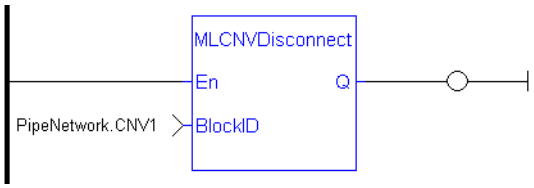
"MLCNVInit" (➜ p. 200)

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.7.5 MLCNVInit

Description

Initializes a converter Pipe Block. Function block is automatically called if a Convertor Block is added to the Pipe Network, with the input mode (position or speed) entered in the Pipe Blocks Properties screen. The Converter block changes the incoming flow of speed or position values to continuous position output with no periodicity.

NOTE

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

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLBlkCreate

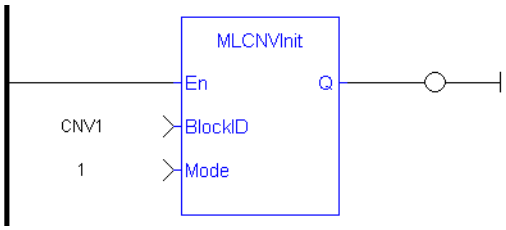
"MLCNVConnect" (➜ p. 193)

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.8 Motion Library - Delay

Name	Description	Return type
"MLDelayInit" (➜ p. 201)	Initializes a delay object	BOOL

2.1.8.1 MLDelayInit

Description

Initializes a delay object. Returns TRUE if the function succeeded. This FB is automatically created in the compiled code of a Pipe Network. It is included in the MLPN_CREATE_OBJECT (created in ST) which is typically executed in a project as part of the startup sequence of the Pipe Network.

Arguments

Input

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

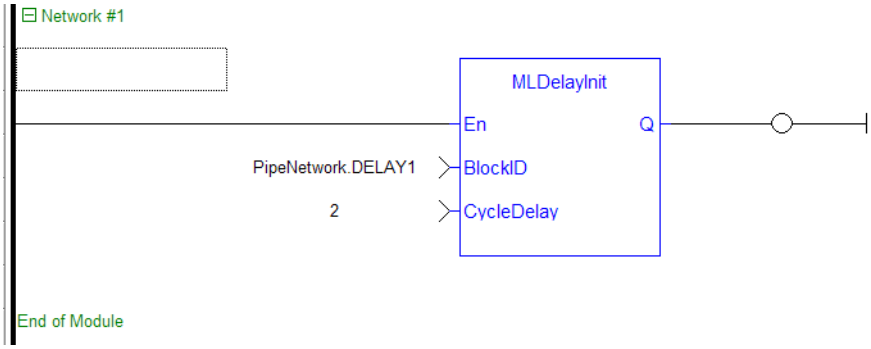
	Unit	n/a
	Default	—
CycleDelay	Description	Number of delay cycles
	Data type	DINT
	Range	[0 , 9]
	Unit	Cycle
	Default	0

Example

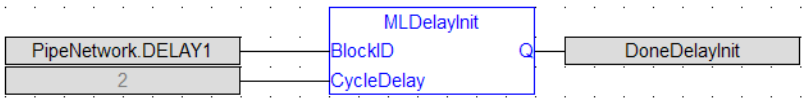
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.9 Motion Library - Derivator

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

2.1.9.1 MLDerInit

Description

Initializes an derivator object. Function block is automatically called if a Derivator Block is added to the Pipe Network, with user-defined settings entered in the Pipe Blocks Properties screen. Input ModuloPosition is defined to manage the periodicity (modulo) of the input values.

NOTE

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

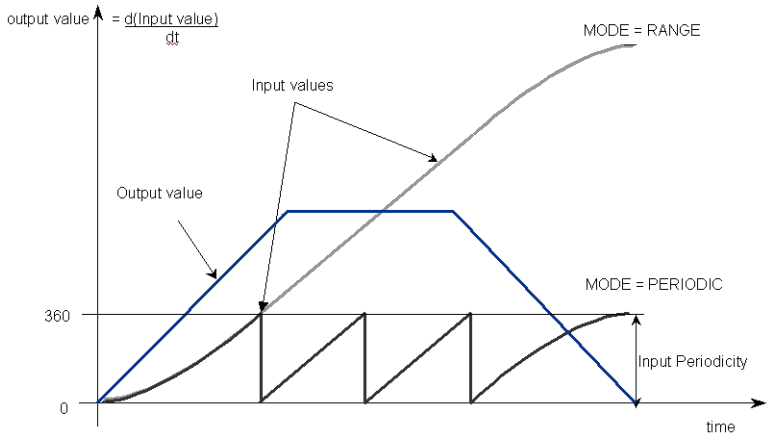


Figure 1-32: MLDerInit

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLBlkCreate

MLDerReadInModPos

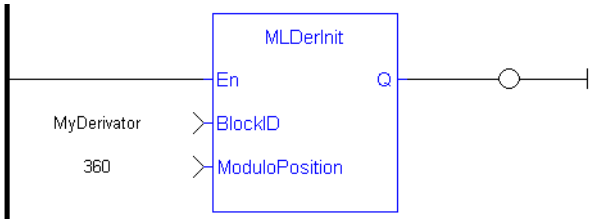
MLDerWriteInModPos

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.9.2 MLDerReadInModPos

Description

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

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

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

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

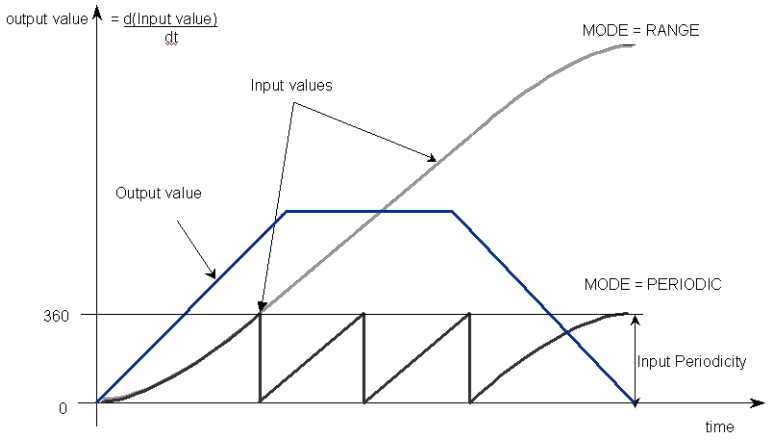


Figure 1-33: MLDerReadInModPos

NOTE

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

Arguments

Input

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

Output

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

Related Functions

MLDerWriteInModPos

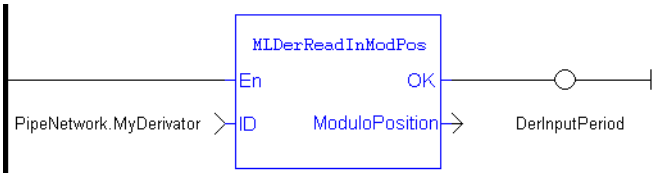
MLDerInit

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.9.3 MLDerWriteInModPos

Description

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

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

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

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

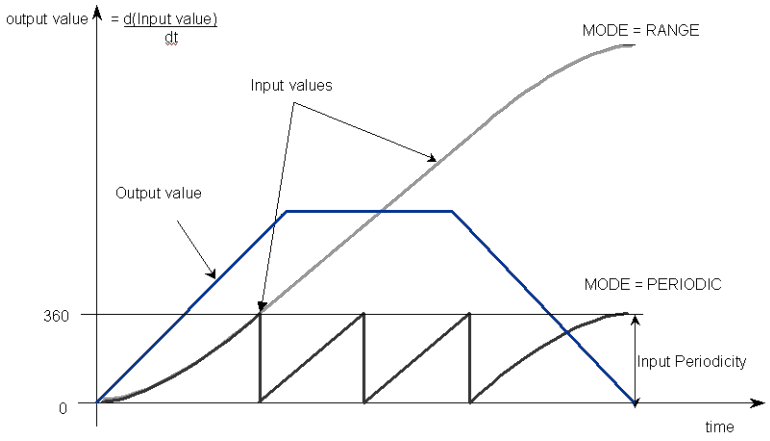


Figure 1-34: MLDerWriteInModPos

NOTE

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

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLDerReadInModPos

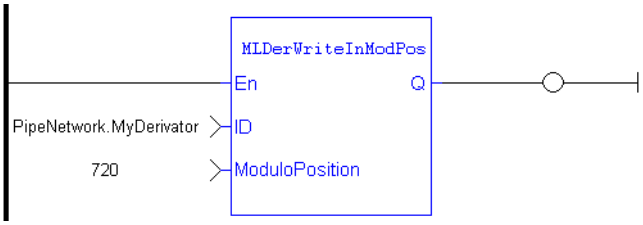
MLDerInit

Example

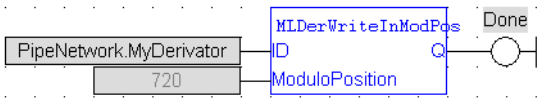
Structured Text

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

Ladder Diagram



Function Block Diagram

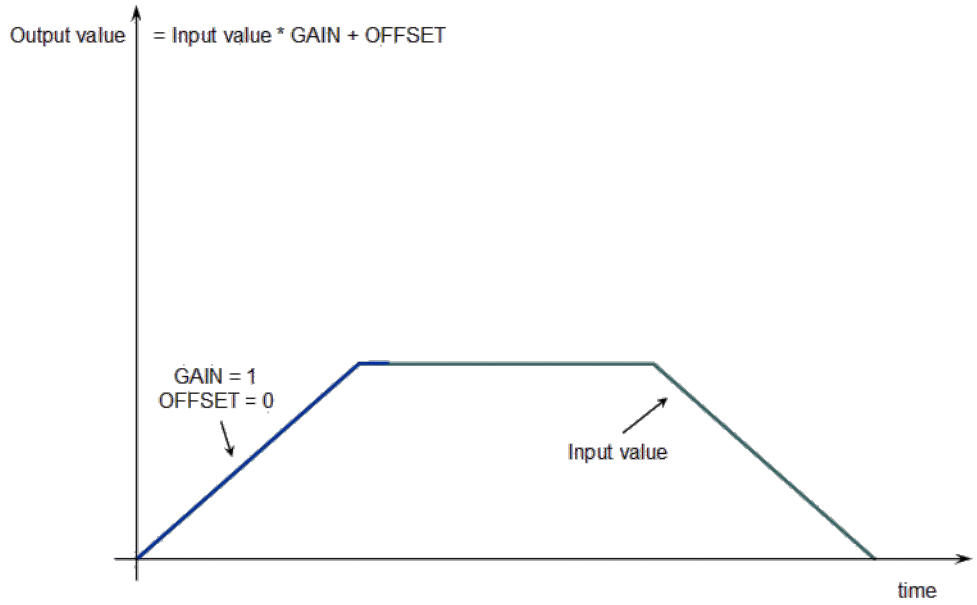


2.1.10 Motion Library - Gear

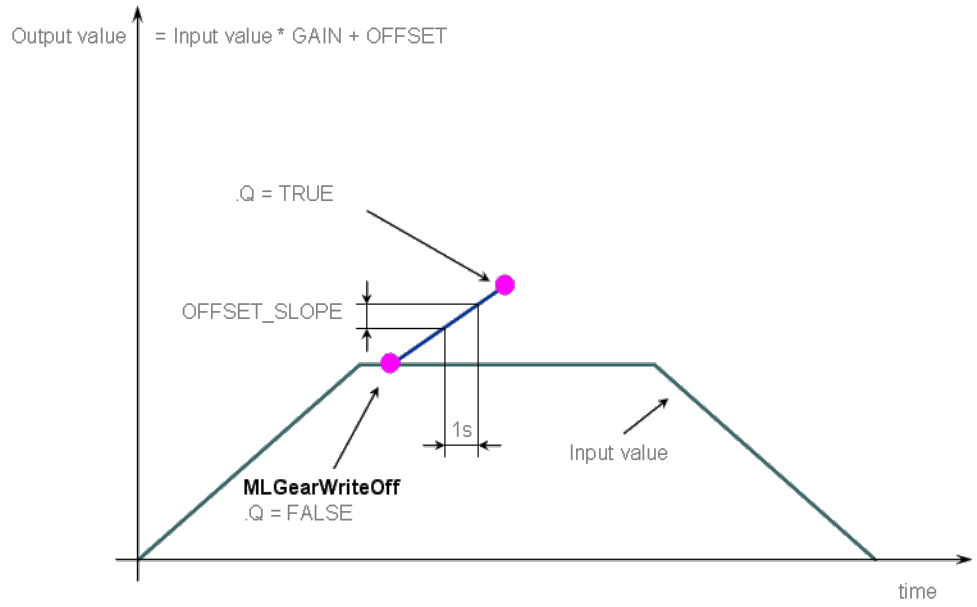
Name	Description	Return type
"MLGearInit" (➜ p. 209)	Initializes a Gear Pipe Block with user-defined settings	BOOL
"MLGearReadOffset" (➜ p. 212)	Returns the offset value of selected Gear Block	None
"MLGearReadOffSlp" (➜ p. 213)	Returns the Offset Slope value of selected Gear Block	None
"MLGearReadRatio" (➜ p. 214)	Returns the ratio value of a gear block	None
"MLGearReadRatSlp" (➜ p. 215)	Returns the ratio slope value of a gear block	None
"MLGearWriteOff" (➜ p. 216)	Sets the Offset value of a selected Gear Pipe Block	BOOL
"MLGearWriteOSlp" (➜ p. 217)	Sets the Offset Slope value of a selected Gear Pipe Block	BOOL
"MLGearWriteRatio" (➜ p. 219)	Sets the Ratio value of a selected Gear Pipe Block	BOOL
"MLGearWriteRatSlp" (➜ p. 220)	Sets the Ratio Slope value of a selected Gear Pipe Block	BOOL

2.1.10.1 Usage example of Gear Functions

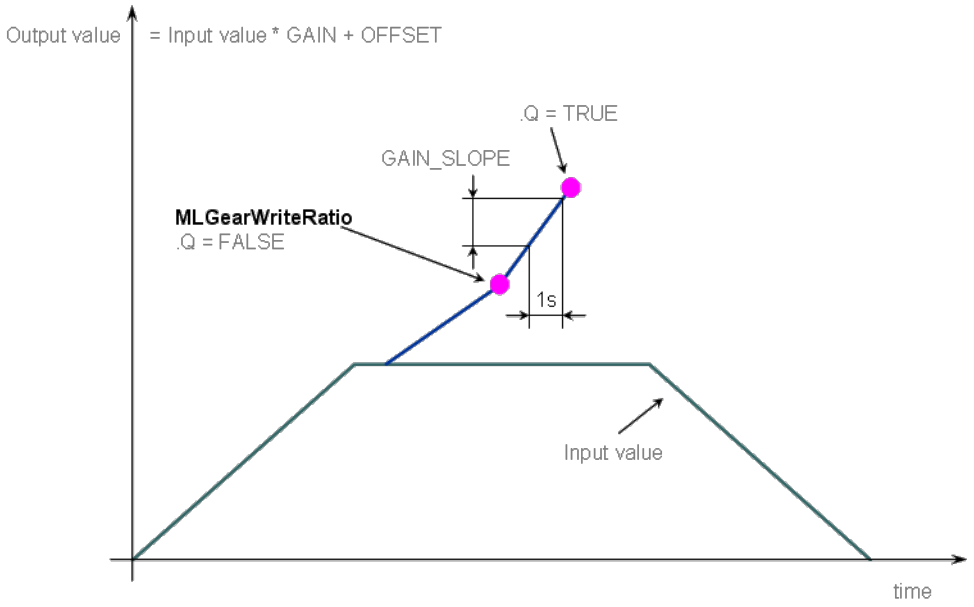
The output value starts with offset = 0 and gain = 1 (blue line)



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).

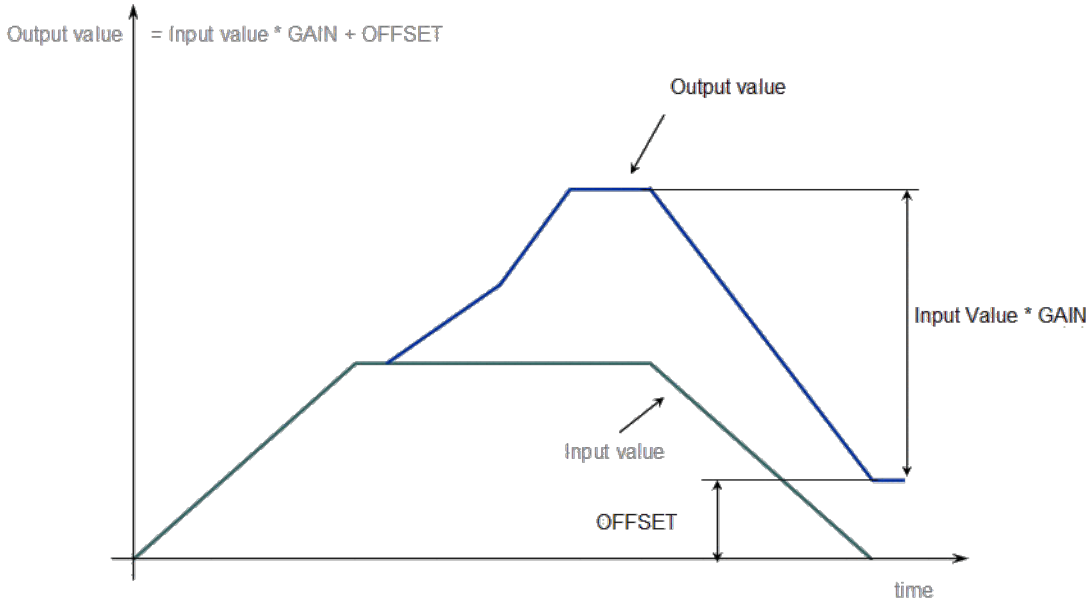


Figure 1-35: Gear Functions Usage

2.1.10.2 MLGearInit

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 default slope value when creating a Gear Block is Max or infinite.

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

NOTE

Be sure to set `RatioSlope < (Ratio * 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.

Arguments

Input

BlockID	**Description**	ID number of a created Pipe Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	GEAR
Ratio	**Description**	Ratio of new Gear Pipe Block. Values lower then 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	1.0
Offset	**Description**	Offset of new Gear Pipe Block. Values lower then 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	0.0
UseUserRatioSlope	**Description**	FALSE to use default MAX or Infinite Slope, TRUE to use user-defined RatioSlope
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	FALSE
RatioSlope	**Description**	User-defined limit at which step changes in Ratio are implemented. Values lower then 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)
	Data type	LREAL
	Range	—
	Unit	1/sec
	Default	0.0
UseUserOffsetSlope	**Description**	FALSE to use default MAX or Infinite Slope, TRUE to use user-defined OffsetSlope

	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	FALSE
OffsetSlope	**Description**	User-defined limit at which step changes in Offset are implemented. Values lower then 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	0.0
Modulo	**Description**	TRUE when mode is modulo. Modulo mode adapts the output values according to the ModuloPosition (modulo)
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	FALSE
Offset	**Description**	Offset of new Gear Pipe Block. Values lower then 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	0.0
RatioSlope	**Description**	User-defined limit at which step changes in Ratio are implemented. Values lower then 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)
	Data type	LREAL
	Range	—
	Unit	1/sec
	Default	0.0

Output

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

Return Type

BOOL

Related Functions

MLBlkCreate

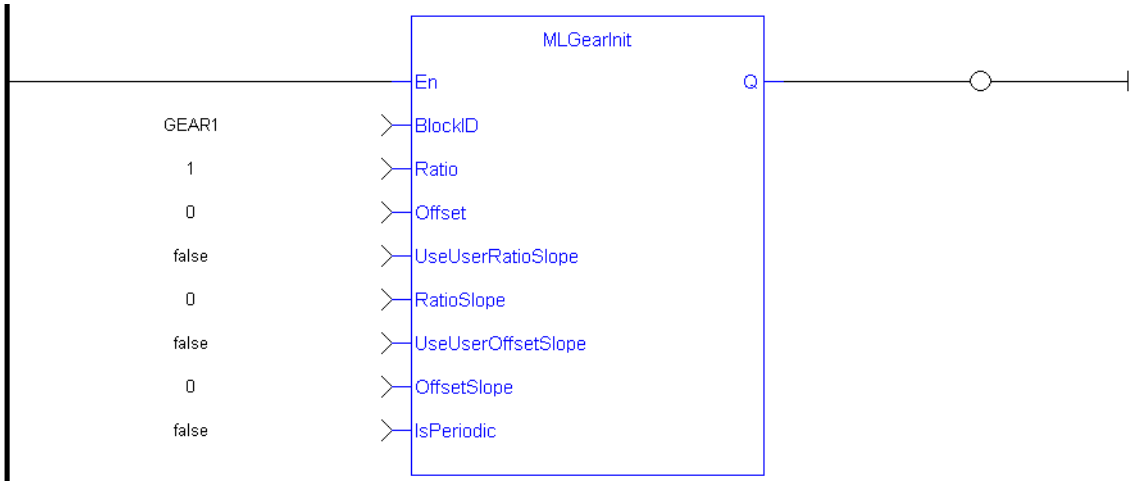
"MLGearWriteRatio" (➜ p. 219)

Example

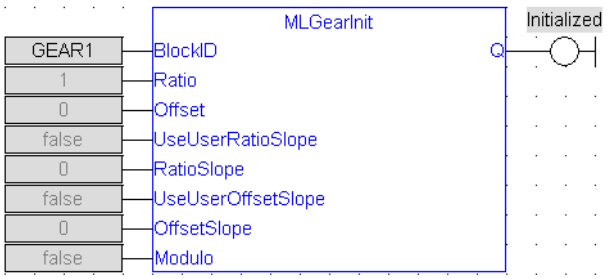
Structured Text

```
//Creates and Initializes a Gear Pipe Block with default values
GEAR1 := MLBlkCreate( 'GEAR1', 'GEAR' );
MLGearInit( GEAR1, 1.0, 0.0, false, 0.0, false, 0.0, false );
```

Ladder Diagram



Function Block Diagram



2.1.10.3 MLGearReadOffset

Description

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

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

Arguments

Input

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

Output

Offset	**Description**	The offset value currently assigned to the selected Gear Pipe Block
	Data type	LREAL
	Unit	User unit

Related Functions

"MLGearWriteOff" (➜ p. 216)

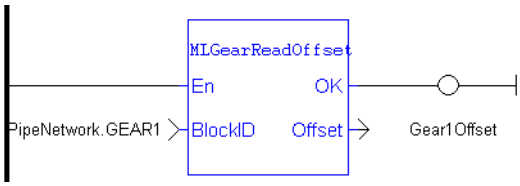
"MLGearInit" (➜ p. 209)

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.10.4 MLGearReadOffSlp

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.

Arguments

Input

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

Output

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

Related Functions

"MLGearWriteOSlp" (➜ p. 217)

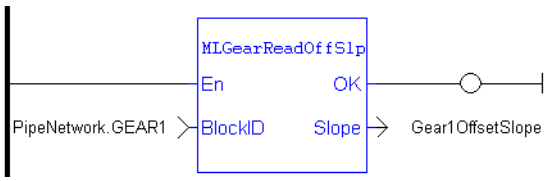
"MLGearInit" (➜ p. 209)

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.10.5 MLGearReadRatio

Description

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

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

Arguments

Input

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

Output

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

Related Functions

"MLGearWriteRatio" (➜ p. 219)

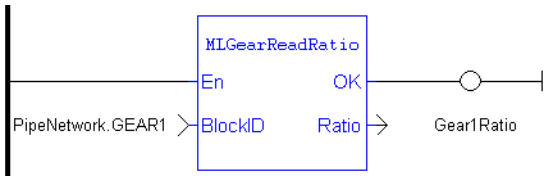
"MLGearInit" (➜ p. 209)

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.10.6 MLGearReadRatSlp

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^{-1}) at which step changes in Ratio are implemented. The default value when creating a Gear Block is RATIO_SLOPE_MAX or infinite.

Arguments

Input

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

Output

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

Related Functions

"MLGearWriteRatSlp" (➜ p. 220)

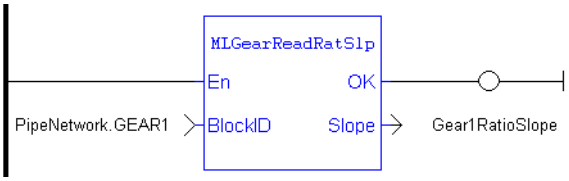
"MLGearInit" (➜ p. 209)

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.10.7 MLGearWriteOff

Description

Sets the Offset value of a selected Gear Pipe Block.

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

TIP

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

Arguments

Input

BlockID	**Description**	ID number of an initialized Gear Pipe Block
	Data type	DINT
	Range	[-2147483648, 2147483648]

	Unit	n/a
	Default	—
Offset	**Description**	New Offset value to be assigned to selected Gear Pipe Block. Values lower then 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

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

Return Type

BOOL

Related Functions

"MLGearReadOffset" (➜ p. 212)

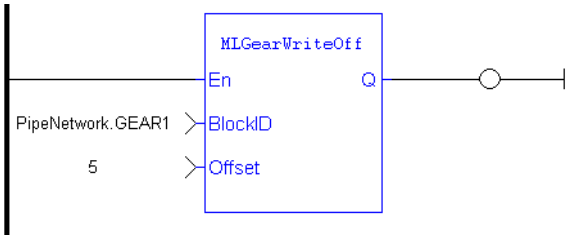
"MLGearInit" (➜ p. 209)

Example

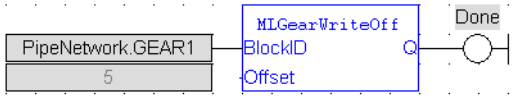
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.10.8 MLGearWriteOSlp

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

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

"MLGearReadOffSlp" (➜ p. 213)

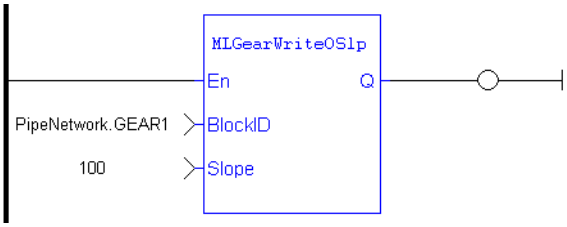
"MLGearInit" (➜ p. 209)

Example

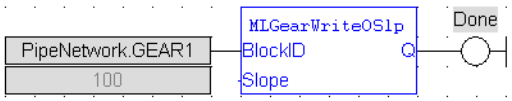
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.10.9 MLGearWriteRatio

Description

Set the Ratio value of a selected Gear Pipe Block.

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

TIP

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

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

"MLGearReadRatio" (➜ p. 214)

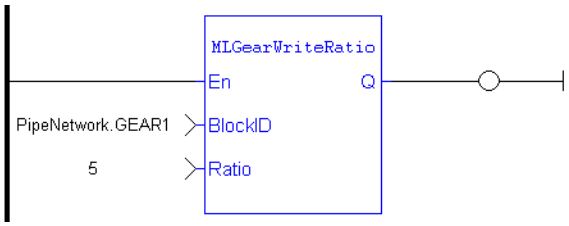
"MLGearInit" (➜ p. 209)

Example

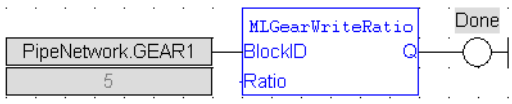
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.10.10 MLGearWriteRatSlp

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. The default value when creating a Gear Block is RATIO_SLOPE_MAX or infinite.

NOTE

Be sure to set `RatioSlope < (Ratio * 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 then 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. 221) in the Examples below.

Arguments

Input

BlockID	**Description**	ID number of an initialized Gear Pipe Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a

	Default	—
Slope	**Description**	New Ratio Slope value to be assigned to selected Gear Pipe Block. Values lower then 1.0 can be entered, but require a leading zero (for example 0.8 instead of .8)
	Data type	LREAL
	Range	—
	Unit	1/sec
	Default	—

Output

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

Return Type

BOOL

Related Functions

"MLGearReadOffSlp" (➜ p. 213)

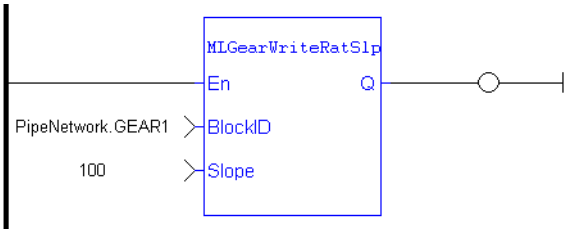
"MLGearInit" (➜ p. 209)

Example

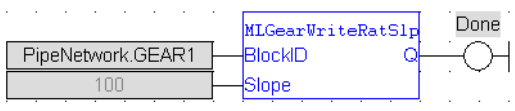
Structured Text

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

Ladder Diagram



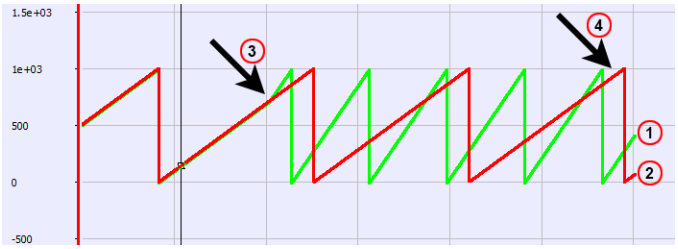
Function Block Diagram



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.

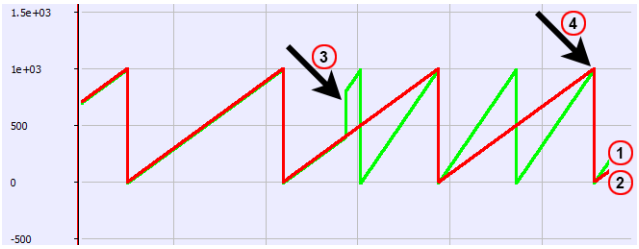


1. Green line: PN Gear Block Output
2. Red line: PN Gearbox Input
3. When the ratio is changed
4. Phase difference

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. Synched

2.1.11 Motion Library - Integrator

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

2.1.11.1 MLIntInit

Description

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

Integrator object can operate in Modulo or not modulo mode. While in Modulo mode, the output values are adapted according to the entered ModuloPosition value.

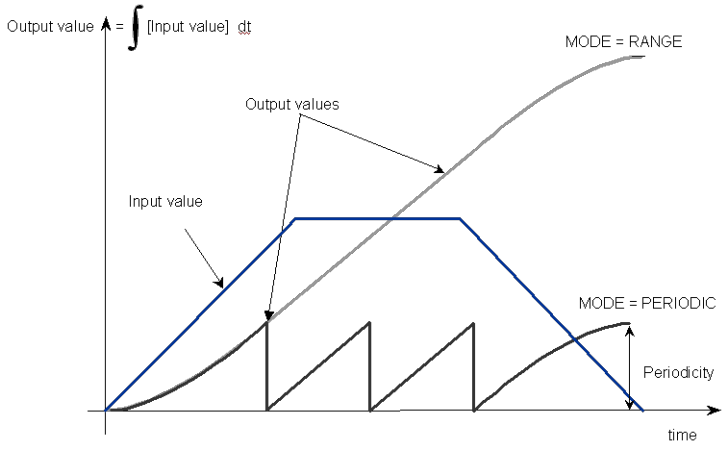


Figure 1-36: MLIntInit

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.

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLBlkCreate

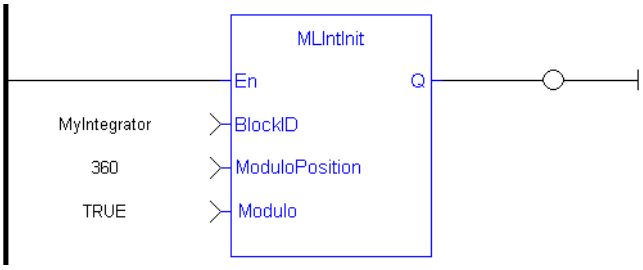
MLIntWriteOutVal

Example

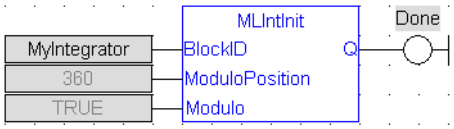
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.11.2 MLIntWriteOutVal

Description

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

NOTE

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

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

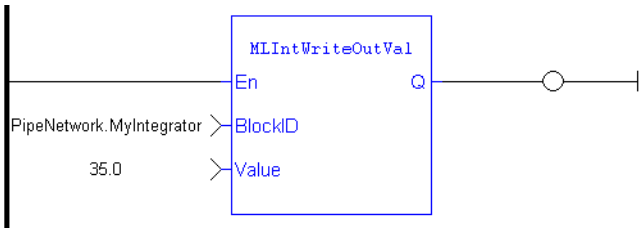
MLIntInit

Example

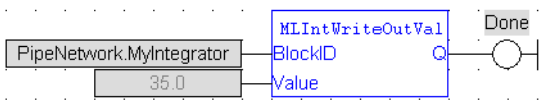
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.12 Motion Library - Master

TIP

For usage example about Master Functions, "" (➜ p. 246)

Function sorted by types:

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

Functions sorted in alphabetical order:

Name	Description	Return type
MLMstAbs	Does an absolute move	BOOL
MLMstAdd	Does an additive move relative for a specified distance from the endpoint of the previous move	BOOL
MLMstForcePos	Forces the specified position. Possible only when the block is **not** moving.	BOOL
MLMstInit	Initializes a master object (TMP generator)	BOOL
MLMstReadAccel	Gets the present acceleration value of a master block	None

Name	Description	Return type
MLMstReadDecel	Gets the present deceleration value of a master block	None
MLMstReadInitPos	Gets the initial position of a master block	None
MLMstReadSpeed	Gets the speed of a master block	None
MLMstRel	Does an Relative move for a specified distance from the current position	BOOL
MLMstRun	Jogs at the specified speed. Returns TRUE if the function succeeded	BOOL
MLMstStatus	Returns the status of the generator	DINT
MLMstWriteAccel	Sets the acceleration of a master block	BOOL
MLMstWriteDecel	Sets the deceleration of a master block	BOOL
MLMstWriteInitPos	Sets the initial position of a master block	BOOL
MLMstWriteSpeed	Sets the speed of a master block	BOOL

2.1.12.1 MLMstAbs

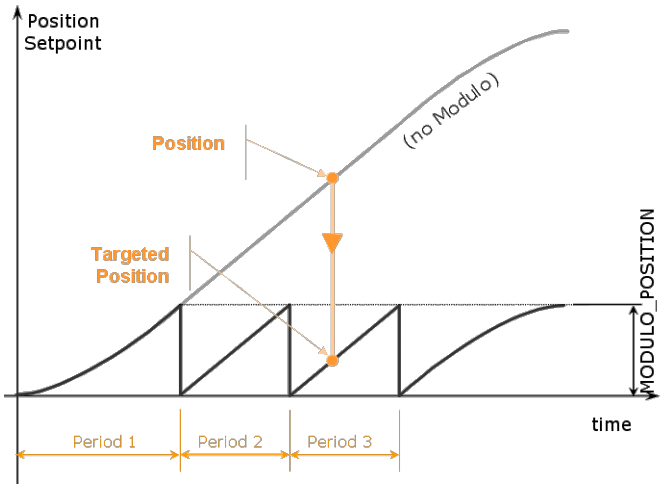
Description

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

Arguments

Input

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

Position	Description	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: • If the Position input is between 0 and the Modulo Position, then the Master Block moves within the **current** period (no position rollover). • If the Position input is greater than the Modulo Position, then the Master Block moves during one of the **next** period (positive position rollover).  The Master Block works similarly for negative positions: if the Position input is less than zero, then the Master Block moves during one of the **previous** period (negative position rollover).
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

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

Related Functions

MLMstWriteSpeed

MLMstWriteDecel

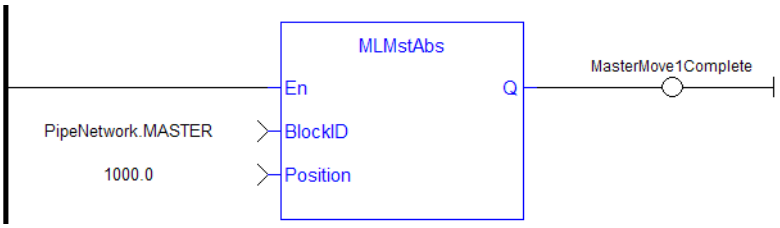
MLMstWriteSpeed

Example

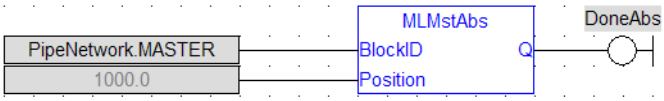
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.12.2 MLMstAdd

Description

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

Arguments

Input

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

Output

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

Related Functions

MLMstWriteSpeed

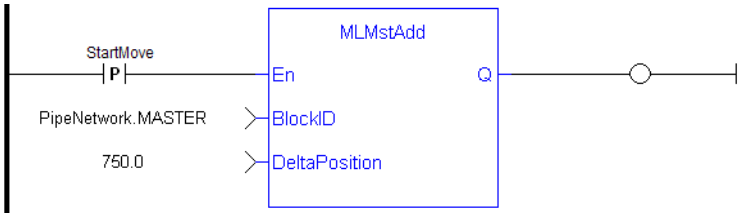
MLMstWriteDecel

Example

Structured Text

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

Ladder Diagram



NOTE

You must use a pulse contact to start the FB

Function Block Diagram

2.1.12.3 MLMstForcePos

Description

Forces the position of a Master Block to a specified position. This block can only be executed when motion is not occurring. It can be used to force the master starting position to the desired values from which to start motion.

Arguments

Input

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

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

Output

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

Related Functions

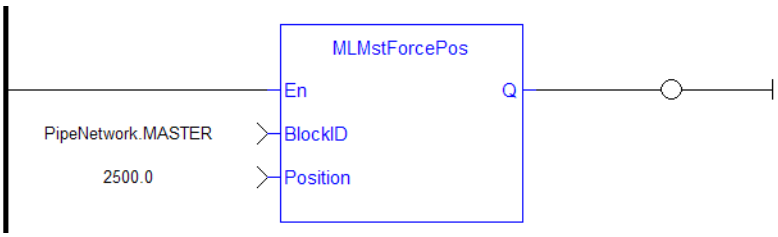
MLMstReadInitPos

Example

Structured Text

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

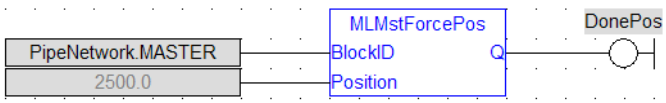
Ladder Diagram



NOTE

You must use a pulse contact to start the FB

Function Block Diagram



2.1.12.4 MLMstInit

Description

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

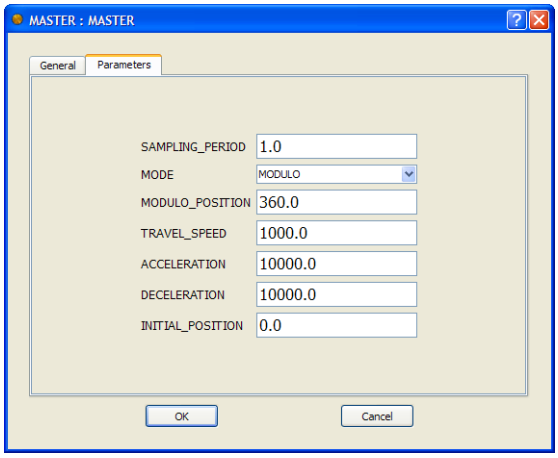


Figure 1-37: TMP Initialization

Arguments

Input

Block ID	Description	ID name of the Master Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
ModuloPosition	Description	Modulo Position for cyclic motion systems expressed in user logical units (Position Rollover Value)
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
Period	Description	Sampling period of the generator expressed according to the update cycle (e.g. 2.0 means the sampling is done once every 2 cycles)
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
Speed	Description	Travel speed value expressed in user logical units per second. The travel speed value is used to set the constant speed part of the trapezoidal motion profile
	Data type	LREAL
	Range	—
	Unit	User unitsec
	Default	—
Acceleration	Description	Acceleration value expressed in user logical units per second squared. The acceleration value is always used to generate the first part of the trapezoidal motion profile
	Data type	LREAL
	Range	—

	Unit	User unit/sec^2
	Default	—
Deceleration	Description	Deceleration value expressed in user logical units per second squared. The deceleration value is always used to generate the last part of the trapezoidal motion profile
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Initial Position	Description	Initial position value expressed in user logical units. Used only at the pipe activation to initialize the position starting point
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
Modulo	Description	The available modes are Modulo (True) or No modulo (False)
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—

Output

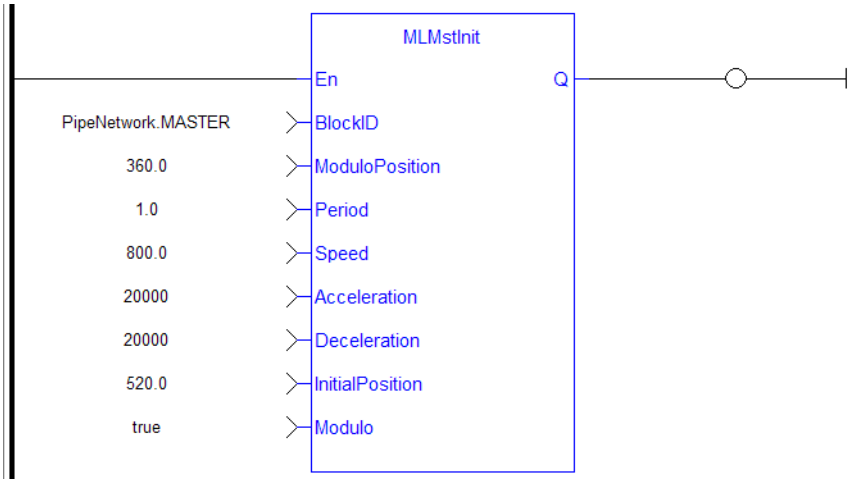
Default (.Q)	Description	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a

Example

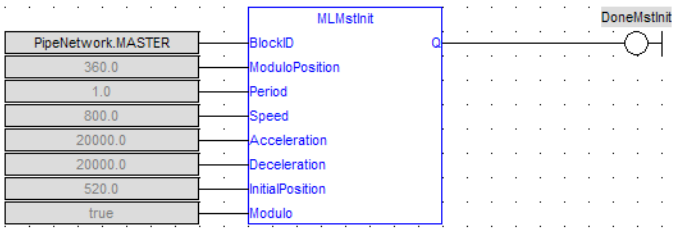
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.12.5 MLMstReadAccel

Description

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

Arguments

Input

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

Output

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

Acceleration	Description	Returns Acceleration value
	Data type	LREAL
	Unit	User unit/sec^2

Related Functions

MLMstReadSpeed

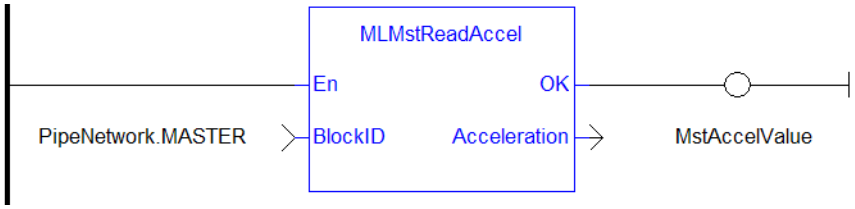
MLMstReadDecel

Example

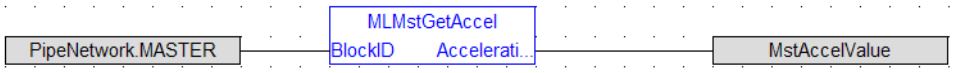
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.12.6 MLMstReadDecel

Description

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

Arguments

Input

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

Output

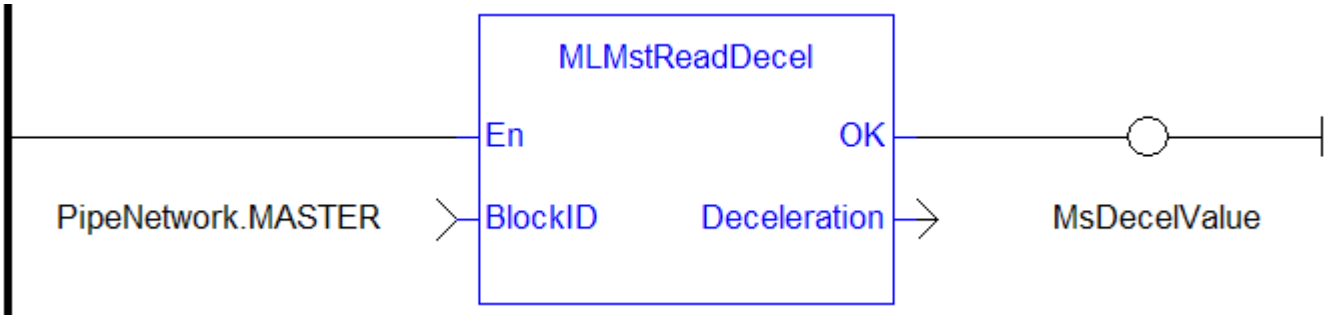
OK	Description	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a
Deceleration	Description	Returns Deceleration value
	Data type	LREAL
	Unit	User unit/sec^2

Example

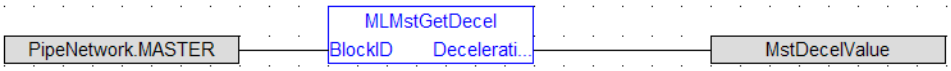
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.12.7 MLMstReadInitPos

Description

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

Arguments

Input

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

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

Output

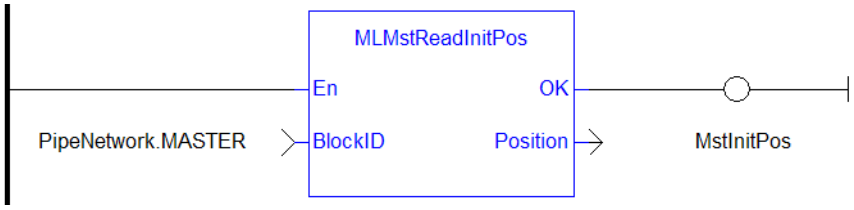
OK	Description	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a
Position	Description	Returns Intial Postion
	Data type	LREAL
	Unit	User unit

Example

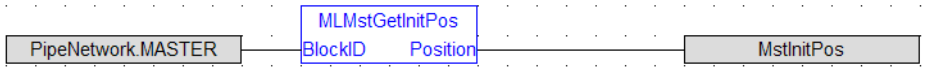
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.12.8 MLMstReadSpeed

Description

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

Arguments

Input

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

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

Output

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

Related Functions

MLMstReadAccel

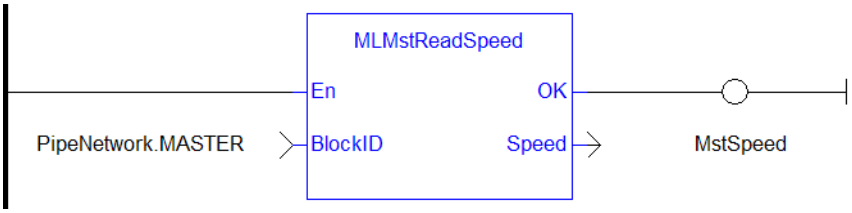
MLMstReadDecel

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.12.9 MLMstRel

Description

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

Arguments

Input

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

Output

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

Related Functions

MLMstWriteSpeed

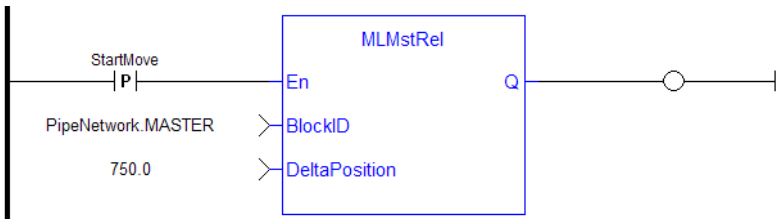
MLMstWriteDecel

Example

Structured Text

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

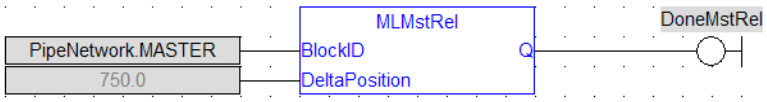
Ladder Diagram



NOTE

You must use a pulse contact to start the FB

Function Block Diagram



2.1.12.10 MLMstRun

Description

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

Arguments

Input

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

Output

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

Related Functions

MLMstWriteSpeed

MLMstWriteDecel

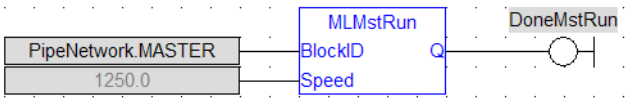
Example

Structured Text

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

Ladder Diagram

Function Block Diagram



2.1.12.11 MLMstStatus

Description

The value returned is the state being executed by the TMP generator as it processes the various motion commands. Some states are transitory, others are stable until the next event takes place. The following terms are relevant to the returned values.

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

Status	Definition
0	(New speed programmed) is entered when a 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.

Arguments

Input

EN	Description	Enables FB to be executed
	Data type	BOOL

	Range	0, 1
	Unit	n/a
	Default	—
Block ID	Description	ID name of the Master Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

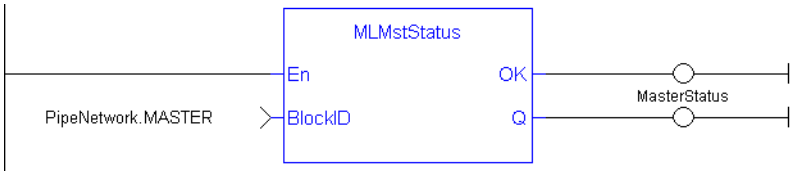
OK	Description	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a
Default (.Q)	Description	Returns the status of the generator
	Data type	DINT
	Unit	n/a

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.12.12 MLMstWriteAccel

Description

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

Arguments

Input

EN	Description	Enables FB to be executed
	Data type	BOOL

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

Output

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

Related Functions

MLMstAbs

MLMstRel

MLMstWriteSpeed

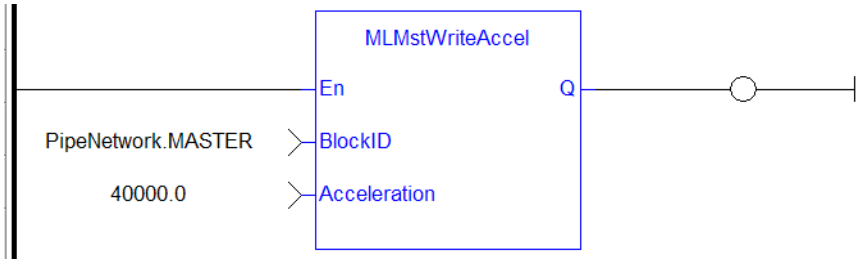
MLMstWriteDecel

Example

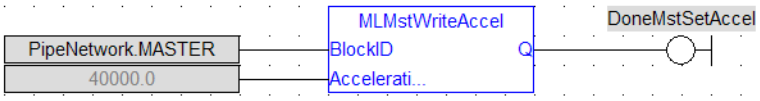
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.12.13 MLMstWriteDecel

Description

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

Arguments

Input

EN	Description	Enables FB to be executed
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Block ID	Description	ID name of the Master Block
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
Deceleration	Description	Deceleration value
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—

Output

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

Related Functions

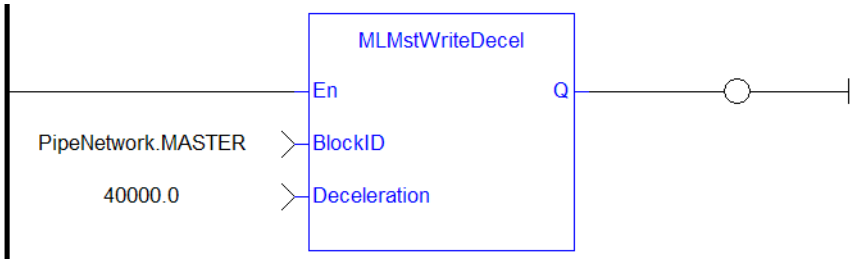
MLMstWriteSpeed

Example

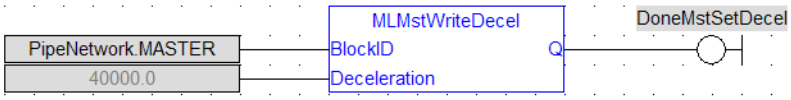
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.12.14 MLMstWriteInitPos

Description

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

Arguments

Input

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

Output

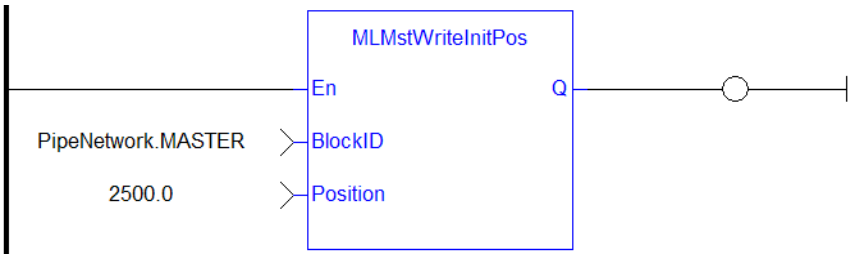
Default (.Q)	Description	Returns true when function successfully executes
	Data type	BOOL
	Unit	n/a

Example

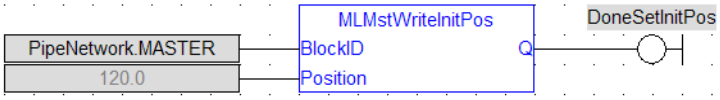
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.12.15 MLMstWriteSpeed

Description

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

Arguments

Input

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

Output

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

Data type	BOOL
Unit	n/a

Related Functions

MLMstWriteSpeed

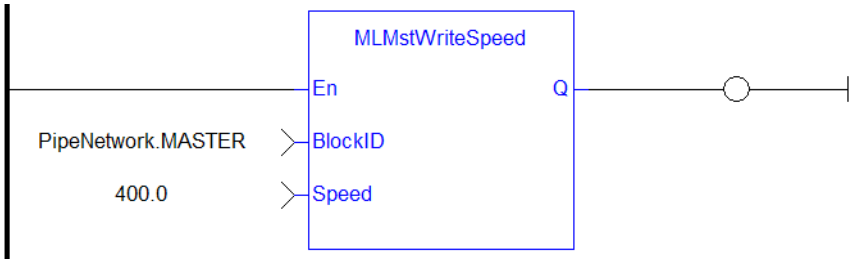
MLMstWriteDecel

Example

Structured Text

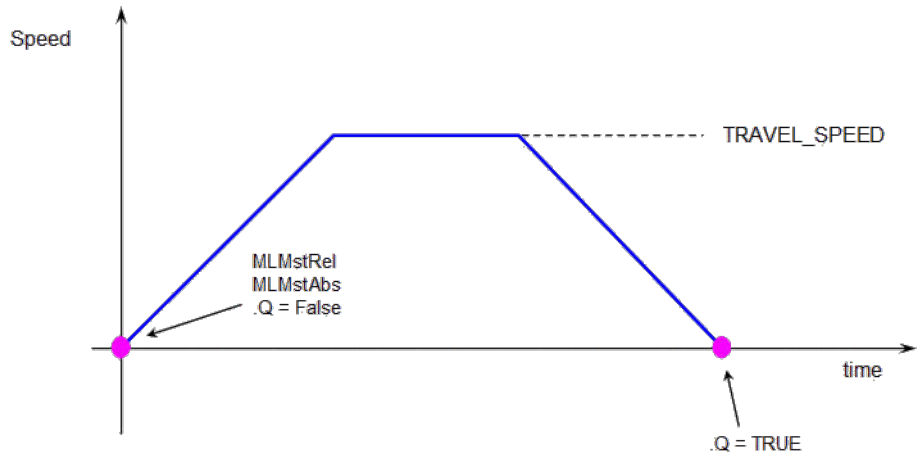
```
MLMstWriteSpeed( PipeNetwork.MASTER, 400.0 );
```

Ladder Diagram



Function Block Diagram

2.1.12.16 Usage example of Master Functions



MLMstRun(0.0) reduce the speed down to 0.

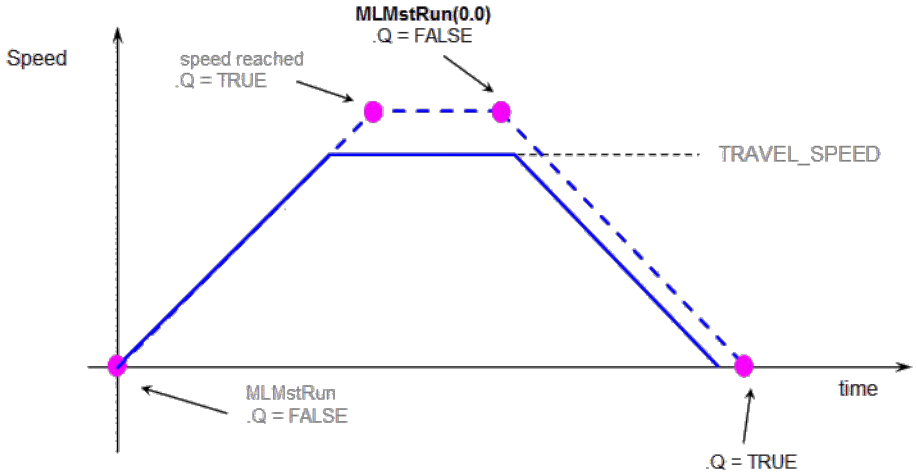


Figure 1-38: Master Functions Usage

2.1.13 Motion Library - Phaser

TIP

For usage example about Phaser Functions, "Usage example of Phaser Functions" (➜ p. 247)

Names	Description	Return type
"MLPhaInit" (➜ p. 248)	Initializes a phaser Pipe Block	BOOL
"MLPhaReadPhase" (➜ p. 251)	Gets the phase value of a phaser block	None
"MLPhaReadSlope" (➜ p. 252)	Gets the phase slope value of a phaser block	None
"MLPhaWritePhase" (➜ p. 253)	Sets the phase value of a phaser block	BOOL
"MLPhaWriteSlope" (➜ p. 254)	Sets the phase slope value of a phaser block	BOOL
"MLPhaReadActPhase" (➜ p. 250)	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.

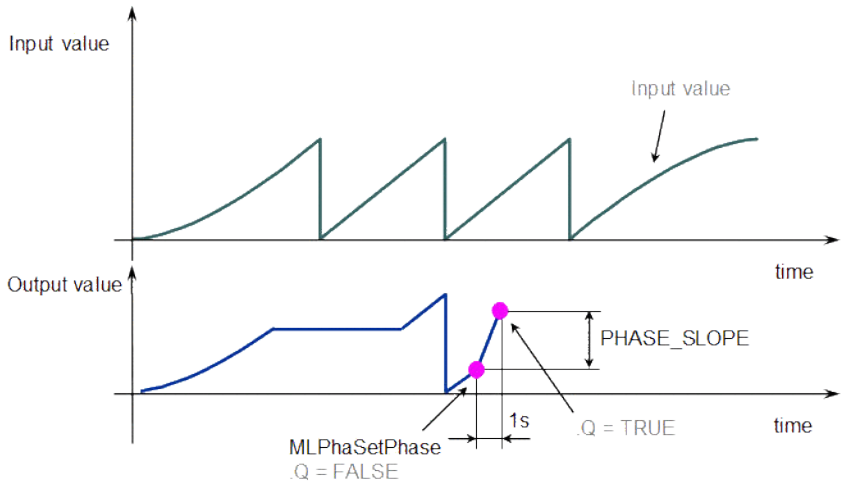
When executing, the phaser block is in one of three states: **Standby**, **Changing phase**, or **Applying phase**.

Standby	Entered when the Block is initialized. Exits to the State "Changing Phase" when the Phase value is changed via the "MLPhaWritePhase" (➜ p. 253) command
Changing Phase	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.
Applying Phase	Entered when the programmed Phase value is reached. Exits to the Changing phase state whenever a new value is programmed via the "MLPhaWritePhase" (➜ p. 253) function changes the Phase Offset target.

2.1.13.1 Usage example of Phaser Functions

You can call "MLPhaWritePhase" (➜ p. 253) function to modify the Phase value..

You can call "MLPhaWriteSlope" (➜ p. 254) to modify the rate of change of phase, or slope, applied when the Phase value is changed.



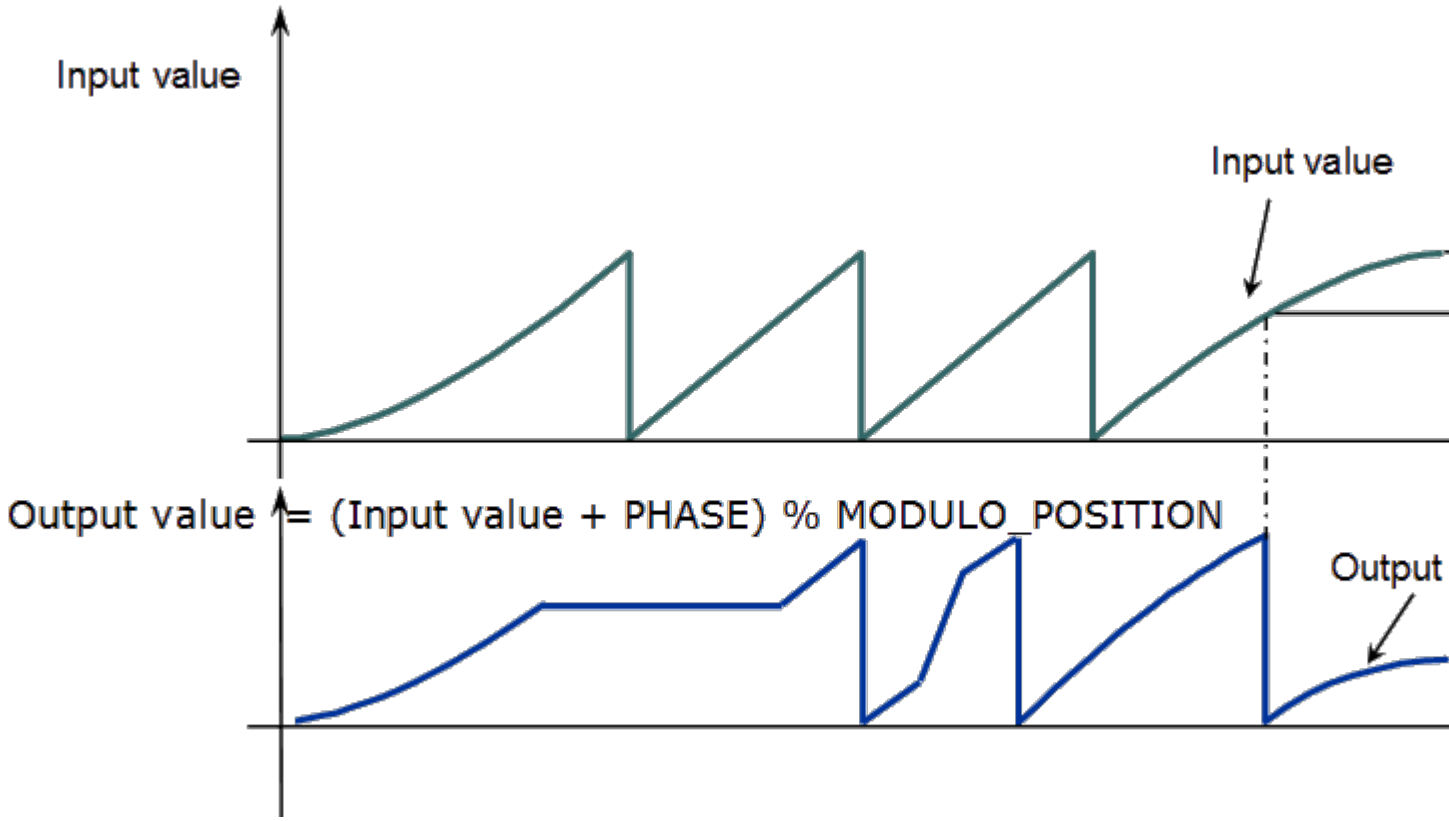


Figure 1-39: Phaser Functions Usage

NOTE

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

2.1.13.2 MLPhaInit

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.

Arguments

Input

BlockID	**Description**	ID Name of a Phaser function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
ModuloPosition	**Description**	Rollover Position of the Phaser block
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	360.0
Phase	**Description**	Amount of Phase adjustment
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	0.0
UseUserSlope	**Description**	Setting determines if Max Slope or user-defined slope is used
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	Max Slope
PhaseSlope	**Description**	User-defined slope for making the phase adjustment
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	0.0
StandbyValue	**Description**	This value is output from the Phaser Block, when the pipe is active, until the "MLPhaWritePhase" (➜ p. 253) function is executed.
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	0.0

Output

Default (.Q)	**Description**	Returns True if the function block is successfully executing
	Data type	BOOL
	Unit	n/a

Related Functions

"MLPhaReadPhase" (➜ p. 251)

"MLPhaReadSlope" (➜ p. 252)

MLPhaInit

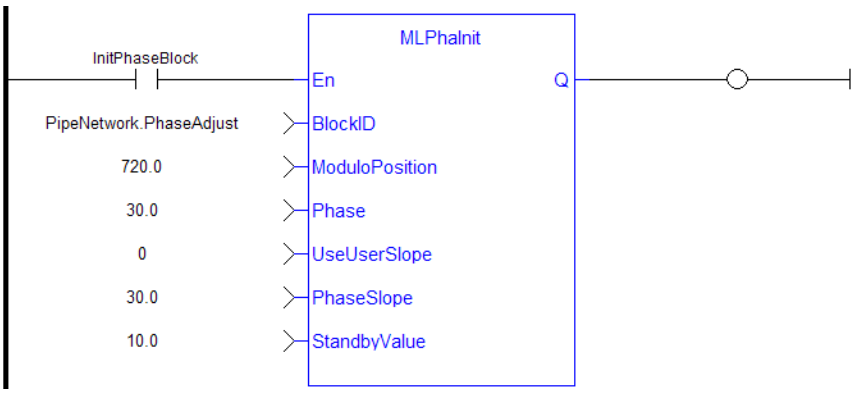
"MLPhaWritePhase" (➜ p. 253)

Example

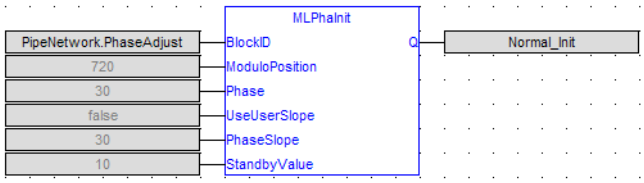
Structured Text

```
MLPhaInit( PipeNetwork.PhaseAdjust , 720, 30, false, 30 , 10 );
```

Ladder Diagram



Function Block Diagram



2.1.13.3 MLPhaReadActPhase

Description

Get the actual phase value of a phaser block.

If a “PHASE_SLOPE_USER” (refer to "MLPhaReadSlope" (➜ p. 252) and "MLPhaWriteSlope" (➜ p. 254)) value is being used, the new phase (refer to "MLPhaWritePhase" (➜ p. 253)) 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. 251) 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.

Arguments

Input

Enable	**Description**
	Data type

	Range	
	Unit	
	Default	
BlockID	**Description**	ID Name of a Phaser function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

OK	**Description**	
	Data type	
	Unit	
Phase	**Description**	
	Data type	LREAL
	Unit	

Related Functions

MLPhaReadPhase, MLPhaWritePhase, MLPhaReadSlope, MLPhaWriteSlope

Example

Structured Text

Ladder Diagram

Function Block Diagram

2.1.13.4 MLPhaReadPhase

Description

Get the phase value of a phaser block.

Arguments

Input

BlockID	Description	ID Name of a Phaser function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

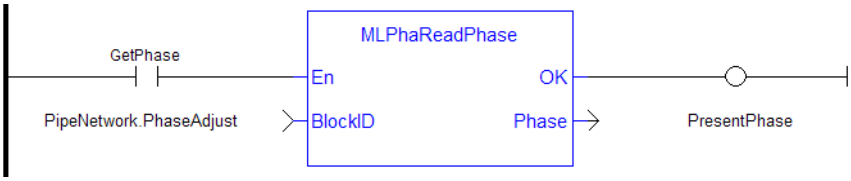
Phase	Data type	LREAL

Example

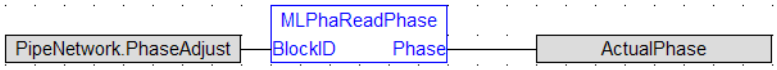
Structured Text

```
PresentPhase := MLPhaReadPhase( PipeNetwork.PhaseAdjust );
```

Ladder Diagram



Function Block Diagram



2.1.13.5 MLPhaReadSlope

Description

Get the phase slope value of a phaser block.

Arguments

Input

BlockID	Description	ID Name of a Phaser function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Slope	Description	Present Slope value
	Data type	LREAL
	Unit	User unit/sec
	Default	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.

Related Functions

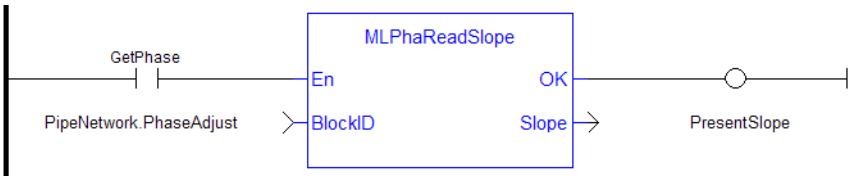
MLPhaReadSlope

Example

Structured Text

```
PresentSlope :=MLPhaReadSlope( PipeNetwork.PhaseAdjust );
```

Ladder Diagram



Function Block Diagram

2.1.13.6 MLPhaWritePhase

Description

Set the phase value of a phaser block.

Arguments

Input

BlockID	Description	ID Name of a Phaser function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
Phase	Description	Phase value
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	Value defined in the setup of a phaser Block within a Pipe Network. It is in the “PHASE” field in the parameter tab

Output

Default (.Q)	Description	Returns True if the function block is successfully executing
	Data type	BOOL
	Unit	n/a

Related Functions

"MLPhaReadPhase" (➜ p. 251)

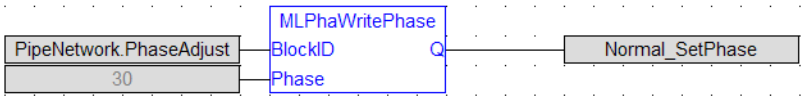
Example

Structured Text

```
MLPhaWritePhase( PipeNetwork.PhaseAdjust , 30 );
```

Ladder Diagram

Function Block Diagram



2.1.13.7 MLPhaWriteSlope

Description

Set the phase value of a phaser block. Returns TRUE if the function succeeded.

Arguments

Input

BlockID	Description	ID Name of a Phaser function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
Slope	Description	Set slope of phase adjust
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	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.

Output

Default (.Q)	Description	Returns True if the function block is successfully executing
	Data type	BOOL
	Unit	n/a

Related Functions

"MLPhaReadSlope" (➜ p. 252)

Example

Structured Text

```
MLPhaWriteSlope( PipeNetwork.PhaseAdjust , 10 );
```

Ladder Diagram

Function Block Diagram

2.1.14 Motion Library - PMP

Name	Description	Return type
"MLPmpAbs" (➜ p. 256)	Moves to an Absolute Position	BOOL
"MLPmpForcePos" (➜ p. 257)	Forces the specified position. Possible only when the block is **not** moving.	BOOL
"MLPmpInit" (➜ p. 258)	Initializes a PMP object (Parabolic Motion Profile generator) with user-defined settings	BOOL
"MLPmpReadAccel" (➜ p. 261)	Gets the Acceleration parameter of a PMP block	None
"MLPmpReadFstSpd" (➜ p. 262)	Gets the FirstTravelSpeed parameter of a PMP block	None
"MLPmpReadInitPos" (➜ p. 263)	Gets the InitialPosition parameter of a PMP block	None
"MLPmpReadJerk" (➜ p. 264)	Gets the Jerk parameter of a PMP block	None
"MLPmpReadLstSpd" (➜ p. 265)	Gets the LastTravelSpeed parameter of a PMP block	None
"MLPmpRel" (➜ p. 266)	Does two subsequent relative moves	BOOL
"MLPmpRun" (➜ p. 267)	Jog the generator at the specified speed	BOOL
"MLPmpStatus" (➜ p. 268)	Returns the status of the PMP block generator	None

Name	Description	Return type
"MLPmpWriteAccel" (➜ p. 270)	Sets the acceleration parameter of a PMP block	BOOL
"MLPmpWriteFstSpd" (➜ p. 271)	Sets the FirstTravelSpeed parameter of a PMP block	BOOL
"MLPmpWriteJerk" (➜ p. 272)	Sets the jerk parameter of a PMP block	BOOL
"MLPmpWriteLstSpd" (➜ p. 273)	Sets the LastTravelSpeed parameter of a PMP block	BOOL

2.1.14.1 MLPmpAbs

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.

Arguments

Input

BlockID	**Description**	ID Name of the PMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
Position	**Description**	Absolute Position of motor/load to be at after this FB is complete
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

Default (.Q)	**Description**	Returns True if the function block is successfully executing
	Data type	BOOL
	Unit	n/a

Related Functions

"MLPmpWriteAccel" (➜ p. 270)

"MLPmpWriteJerk" (➜ p. 272)

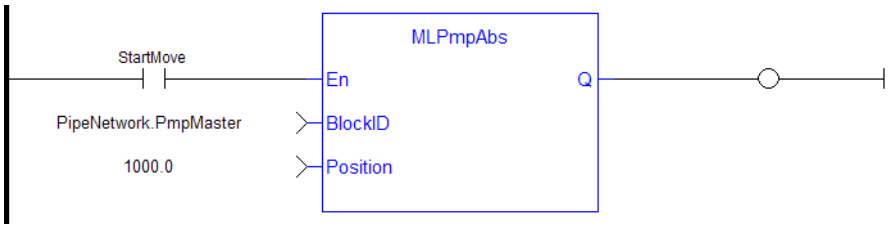
"MLPmpWriteFstSpd" (➜ p. 271)

Example

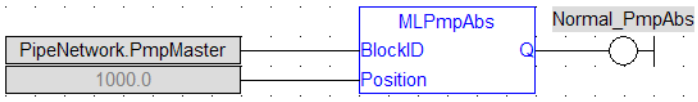
Structured Text

```
MLPmpAbs( PipeNetwork.PmpMaster, 1000.0 ) ;
```

Ladder Diagram



Function Block Diagram



2.1.14.2 MLPmpForcePos

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.

Arguments

Input

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

Output

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

Data type	BOOL
Unit	n/a

Related Functions

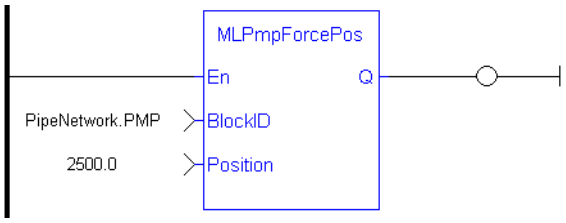
"MLPmpReadInitPos" (➜ p. 263)

Example

Structured Text

```
MLPmpForcePos( PipeNetwork.PMP, 2500.0 );
```

Ladder Diagram



NOTE

You must use a pulse contact to start the FB

Function Block Diagram

2.1.14.3 MLPmpInit

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

Arguments

Input

BlockID	**Description**	ID Name of a PMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
ModuloPosition	**Description**	Modulo Position for cyclic motion systems expressed in user logical units (Position Rollover Value)
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	360.0
Period	**Description**	Sampling period of the generator expressed according to the update cycle (e.g. 2.0 means the sampling is done once every 2 cycles)
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	1.0
FirstTravelSpeed	**Description**	First Travel Speed of the motion
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	100.0
LastTravelSpeed	**Description**	Last Travel Speed of the motion
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	100.0
Acceleration	**Description**	Acceleration of the Pmp block motion
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	1000.0
Jerk	**Description**	Jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec^3
	Default	0
InitialPosition	**Description**	Initial Position of the Pmp block when the Pipe Network is start up
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	0

Modulo	**Description**	The available modes are Modulo (True) or No modulo (False)
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—

Output

Default (.Q)	**Description**	Returns True if the function block is successfully executing
	Data type	BOOL
	Unit	n/a

Related Functions

"MLPmpReadAccel" (➜ p. 261)

"MLPmpReadFstSpd" (➜ p. 262)

"MLPmpReadInitPos" (➜ p. 263)

"MLPmpReadJerk" (➜ p. 264)

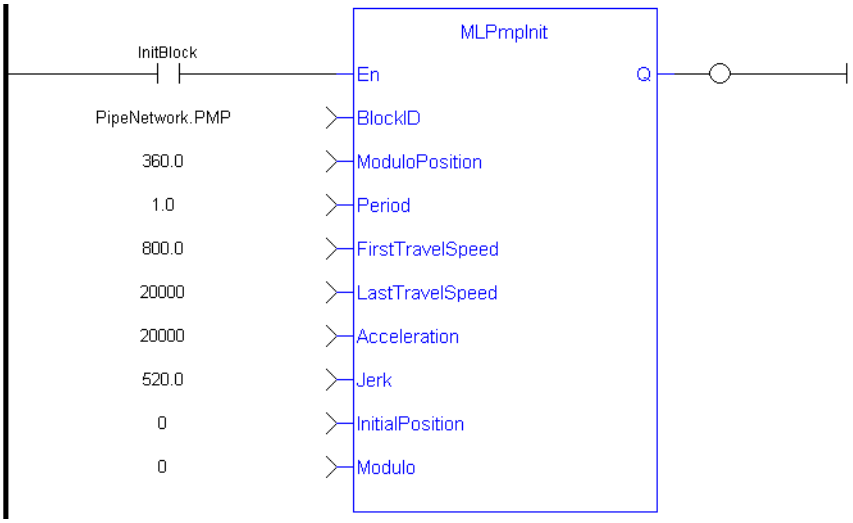
"MLPmpReadLstSpd" (➜ p. 265)

Example

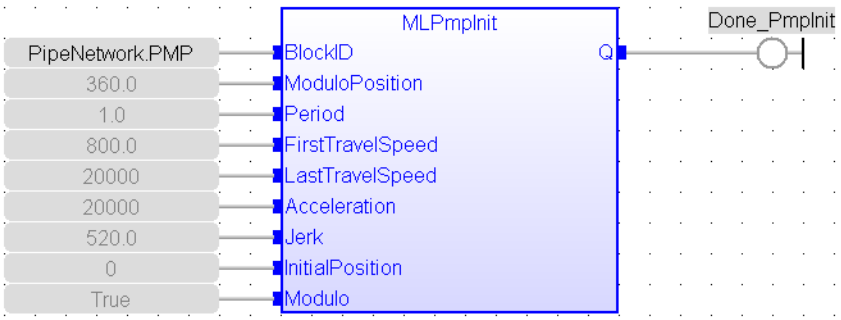
Structured Text

```
MLPmpInit( PipeNetwork.PmpMaster , 360.0, 1.0, 800.0, 20000.0, 20000.0,
520.0, 0, true  ) ;
```

Ladder Diagram



Function Block Diagram



2.1.14.4 MLPmpReadAccel

Description

Get the Acceleration parameter of a PMP block used in both the MLPmpAbs and MLPmpRel function block.

Arguments

Input

BlockID	**Description**	ID Name of the PMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Acceleration	**Description**	Present Acceleration of the PMP PipeNetork Function Block
	Data type	LREAL
	Unit	User unit/sec^2
	Default	Value defined PMP Block when creating a Pipe Network. It is in the “ACCELERATION” field in the parameter tab.

Related Functions

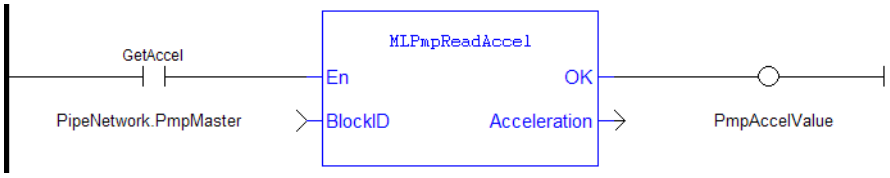
"MLPmpReadFstSpd" (➜ p. 262)
"MLPmpReadInitPos" (➜ p. 263)
"MLPmpReadJerk" (➜ p. 264)
"MLPmpReadLstSpd" (➜ p. 265)

Example

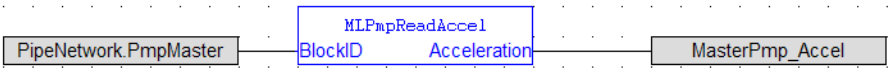
Structured Text

```
PmpAccelValue :=  MLPmpReadAccel( PipeNetwork.PmpMaster ) ;
```

Ladder Diagram



Function Block Diagram



2.1.14.5 MLPmpReadFstSpd

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.

Arguments

Input

BlockID	**Description**	ID Name of a PMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

FirstTravelSpeed	**Description**	Present first travel velocity of the PMP PipeNetork Function Block
	Data type	LREAL
	Unit	User unit/sec
	Default	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

Related Functions

"MLPmpReadAccel" (➜ p. 261)

MLPmpReadFstSpd

"MLPmpReadInitPos" (➜ p. 263)

"MLPmpReadJerk" (➜ p. 264)

"MLPmpReadLstSpd" (➜ p. 265)

"MLPmpWriteLstSpd" (➜ p. 273)

Example

Structured Text

```
FirstSpeedValue := MLPmpReadFstSpd( PipeNetwork.PmpMaster ) ;
```

Ladder Diagram

Function Block Diagram

2.1.14.6 MLPmpReadInitPos

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.

Arguments

Input

BlockID	Description	ID Name of a PMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

InitialPosition	Description	Present Initial Position of the PMP PipeNetork Function Block
	Data type	LREAL
	Unit	User unit
	Default	Value defined in the setup of a PMP Block within a Pipe Network. It is in the “INITIAL_POSITION” field in the parameter tab.

Related Functions

"MLPmpInit" (➜ p. 258)

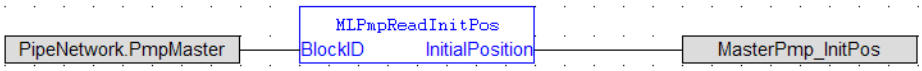
Example

Structured Text

```
PmpInitPos := MLPmpReadInitPos( PipeNetwork.PmpMaster ) ;
```

Ladder Diagram

Function Block Diagram



2.1.14.7 MLPmpReadJerk

Description

Get the Jerk parameter of a PMP block used in both the MLPmpAbs and MLPmpRel function block.

Arguments

Input

BlockID	Description	ID Name of a PMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Jerk	Description	Jerk of the PMP PipeNetork Function Block
	Data type	LREAL
	Unit	User unit/sec^3
	Default	Value defined in the setup of a PMP Block within a Pipe Network. It is in the “JERK” field in the parameter tab.

Example

Structured Text

```
PmpJerkValue := MLPmpReadJerk( PipeNetwork.PmpMaster ) ;
```

Ladder Diagram

Function Block Diagram

2.1.14.8 MLPmpReadLstSpd

Description

Get the LastTravelSpeed parameter of a PMP block used in the MLPmpRel function block.

Arguments

Input

BlockID	**Description**	ID Name of a PMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

LastTravelSpeed	**Description**	Last Travel Speed of the PMP PipeNetork Function Block
	Data type	LREAL
	Unit	User unit/sec
	Default	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.

Related Functions

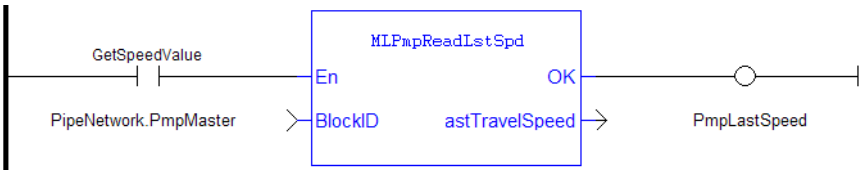
"MLPmpReadAccel" (➜ p. 261)
"MLPmpReadFstSpd" (➜ p. 262)
"MLPmpReadInitPos" (➜ p. 263)
"MLPmpReadJerk" (➜ p. 264)

Example

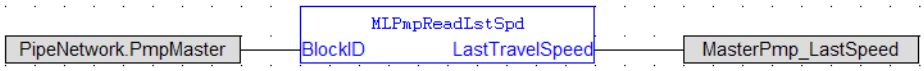
Structured Text

```
PmpLastSpeed := MLPmpReadLstSpd( PipeNetwork.PmpMaster ) ;
```

Ladder Diagram



Function Block Diagram



2.1.14.9 MLPmpRel

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.

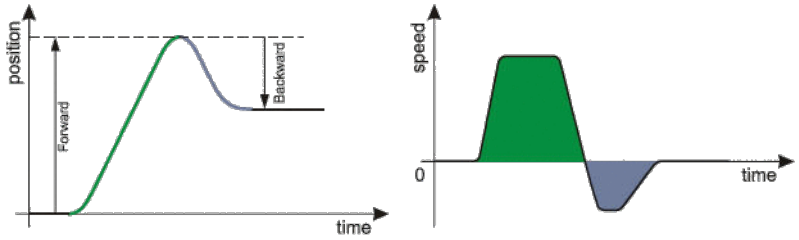


Figure 1-40: PMP Generator Forward & Backward Motion Profile

Arguments

Input

BlockID	**Description**	ID Name of the PMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
DeltaFirst	**Description**	Length of first Move
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	Value defined in the setup of a PMP Block within a Pipe Network. It is the "FIRST_TRAVEL_SPEED" field in the parameter tab.
DeltaSecond	**Description**	Length of second (return) Move
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	Value defined in the setup of a PMP Block within a Pipe Network. It is the "LAST_TRAVEL_SPEED" field in the parameter tab.

Output

Default (.Q)	**Description**	Returns True if the MLPmlRel successfully completed
	Data type	BOOL
	Unit	n/a

Related Functions

"MLPmpWriteAccel" (➜ p. 270)

"MLPmpWriteFstSpd" (➜ p. 271)

"MLPmpWriteJerk" (➜ p. 272)

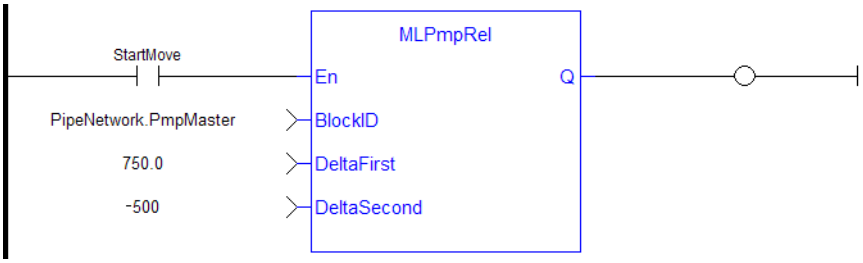
"MLPmpWriteLstSpd" (➜ p. 273)

Example

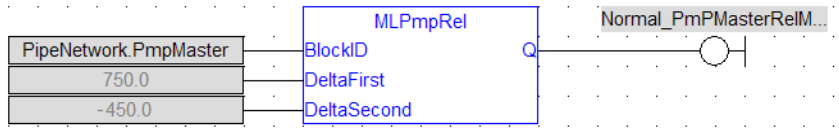
Structured Text

```
MLPmpRel( PipeNetwork.PmpMaster, 4000 , -2500 ) ;
```

Ladder Diagram



Function Block Diagram



2.1.14.10 MLPmpRun

Description

Jog the generator at the requested speed.

Arguments

Input

EN	**Description**	Enables FB to be executed. Is only recognized if the PMP generator is Idle or at constant velocity as determined from the "MLPmpStatus" (➜ p. 268) function.
	Data type	BOOL

	Range	0, 1
	Unit	n/a
	Default	—
BlockID	**Description**	ID Name of the PMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
Speed	**Description**	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).
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—

Output

Default (.Q)	**Description**	Returns True if the MLPmpRun successfully completed.
	Data type	BOOL
	Unit	n/a

Related Functions

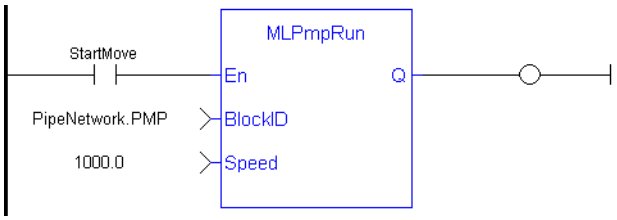
"MLPmpStatus" (➜ p. 268)

Example

Structured Text

```
MLPmpRun( PipeNetwork.PmpMaster, 1000.0 ) ;
```

Ladder Diagram



Function Block Diagram

2.1.14.11 MLPmpStatus

Description

Returns the status of the PMP block generator.

Arguments

Input

EN	Description	Enables FB to be executed.
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
BlockID	Description	ID Name of the PMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

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

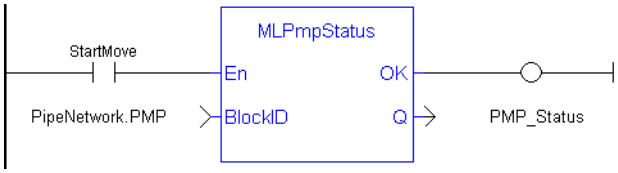
Value	Description
0	Indicates that the PMP block is idle. No command is currently running in the generator. It can be used to determine that a previous move is complete.
1	Indicates that the PMP block is either accelerating to a position or speed, or is decelerating to a position or speed.
2	Indicates that the PMP block is running at a constant speed.

Example

Structured Text

```
PMP_Status := MLPmpStatus ( PipeNetwork.PmpMaster ) ;
Done :=TRUE;
```

Ladder Diagram



Function Block Diagram

2.1.14.12 MLPmpWriteAccel

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.

Arguments

Input

BlockID	**Description**	ID Name of the PMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
Acceleration	**Description**	Acceleration Value
	Data type	LREAL
	Range	> 0
	Unit	User unit/sec^2
	Default	Value defined in the setup of a PMP Block within a Pipe Network. It is the “ACCELERATION” field the parameter tab.

Output

Default (.Q)	**Description**	Returns True if the function block is successfully executing
	Data type	BOOL
	Unit	n/a

Related Functions

"MLPmpAbs" (➜ p. 256)

"MLPmpRel" (➜ p. 266)

"MLPmpWriteFstSpd" (➜ p. 271)

"MLPmpWriteJerk" (➜ p. 272)

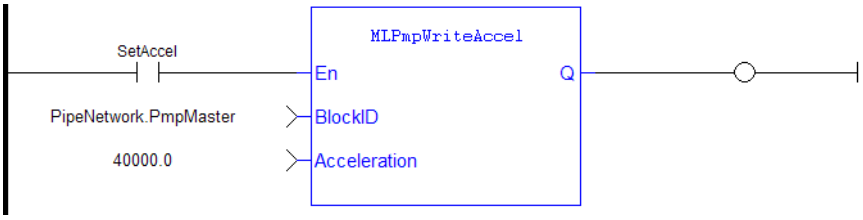
"MLPmpWriteLstSpd" (➜ p. 273)

Example

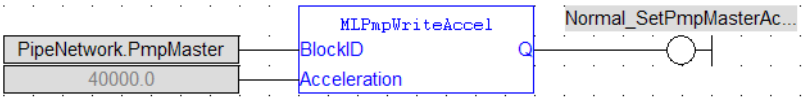
Structured Text

```
MLPmpWriteAccel( PipeNetwork.PmpMaster, 40000.0 ) ;
```

Ladder Diagram



Function Block Diagram



2.1.14.13 MLPmpWriteFstSpd

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.

Arguments

Input

BlockID	**Description**	ID Name of the PMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
First Travel Speed	**Description**	First Travel Speed Value
	Data type	LREAL
	Range	> 0
	Unit	User unit/sec
	Default	Value defined in the setup of a PMP Block within a Pipe Network. It is the “FIRST_TRAVEL_SPEED” field in the parameter tab.

Output

Default (.Q)	**Description**	Returns True if the function block is successfully executing
	Data type	BOOL
	Unit	n/a

Related Functions

"MLPmpAbs" (➜ p. 256)

"MLPmpRel" (➜ p. 266)

"MLPmpWriteAccel" (➜ p. 270)

"MLPmpWriteJerk" (➜ p. 272)

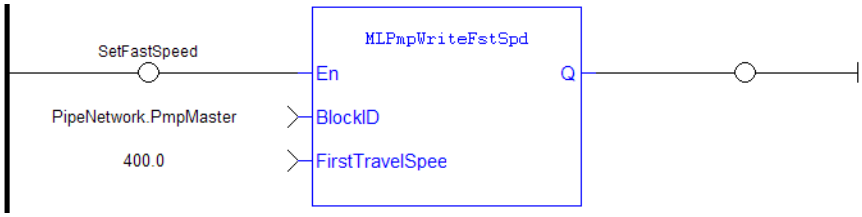
"MLPmpWriteLstSpd" (➜ p. 273)

Example

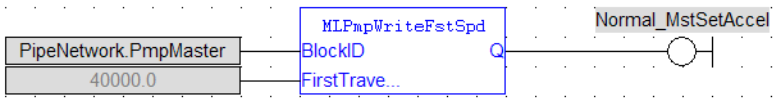
Structured Text

```
MLPmpWriteFstSpd( PipeNetwork.PmpMaster, 300.0 ) ;
```

Ladder Diagram



Function Block Diagram



2.1.14.14 MLPmpWriteJerk

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.

Arguments

Input

BlockID	**Description**	ID Name of the PMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
Jerk	**Description**	Jerk Value
	Data type	LREAL
	Range	> 0
	Unit	User unit/sec^3
	Default	Value defined in the setup of a PMP Block within a Pipe Network. It is the “JERK” field in the parameter tab.

Output

Default (.Q)	Description	Returns True if the function block is successfully executing
	Data type	BOOL
	Unit	n/a

Related Functions

"MLPmpAbs" (➜ p. 256)

"MLPmpReadJerk" (➜ p. 264)

"MLPmpRel" (➜ p. 266)

"MLPmpWriteAccel" (➜ p. 270)

"MLPmpWriteFstSpd" (➜ p. 271)

"MLPmpWriteLstSpd" (➜ p. 273)

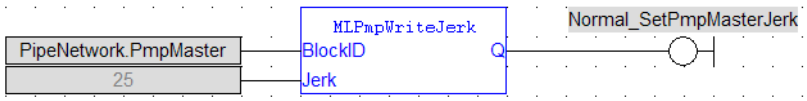
Example

Structured Text

```
MLPmpWriteJerk( PipeNetwork.PmpMaster, 15.0 ) ;
```

Ladder Diagram

Function Block Diagram



2.1.14.15 MLPmpWriteLstSpd

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.

Arguments

Input

BlockID	Description	ID Name of the PMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Last Speed	**Description**	Last Travel Speed Value
	Data type	LREAL
	Range	> 0
	Unit	User unit/sec
	Default	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.

Output

Default (.Q)	**Description**	Returns True if the function block is successfully executing
	Data type	BOOL
	Unit	n/a

Related Functions

"MLPmpAbs" (➜ p. 256)

"MLPmpReadLstSpd" (➜ p. 265)

"MLPmpRel" (➜ p. 266)

"MLPmpWriteAccel" (➜ p. 270)

"MLPmpWriteFstSpd" (➜ p. 271)

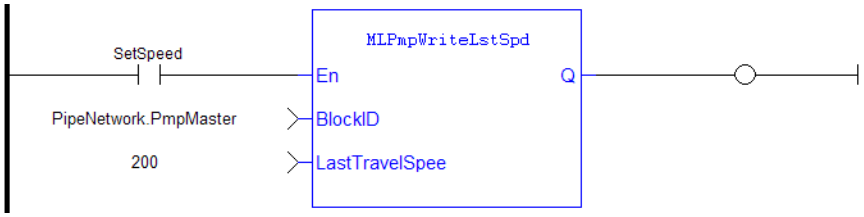
"MLPmpWriteJerk" (➜ p. 272)

Example

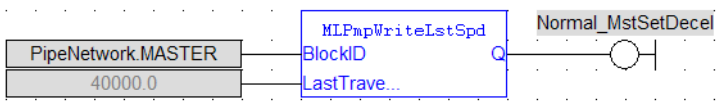
Structured Text

```
MLPmpWriteLstSpd( PipeNetwork.PmpMaster, 650 ) ;
```

Ladder Diagram



Function Block Diagram



2.1.15 Motion Library - Sampler

Name	Description	Return type
"MLSmpConnect" (➜ p. 275)	Connects a sampler block to a pipe network axis or pipe block	BOOL

Name	Description	Return type
"MLSmpConECAT" (➜ p. 276)	Connects a sampler block to the specified CoE object in a PDO	BOOL
"MLSmpConPLCAxis" (➜ p. 279)	Connects a sampler block to a PLCopen axis variable	BOOL
"MLSmpConPNAxis" (➜ p. 281)	Connects a sampler block to a Pipe Network axis variable	BOOL
"MLSmpInit" (➜ p. 278)	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

Description

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

Related Function Blocks

"MLSmpConECAT" (➜ p. 276), "MLSmpInit" (➜ p. 278), "MLSmpConPNAxis" (➜ p. 281), "MLSmpConPLCAxis" (➜ p. 279)

Arguments

Input

BlockID	**Description**	ID Name of the SMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
PipeBlockID	**Description**	ID Name of the Pipe Block the sampler is connected to
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

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

Return Type

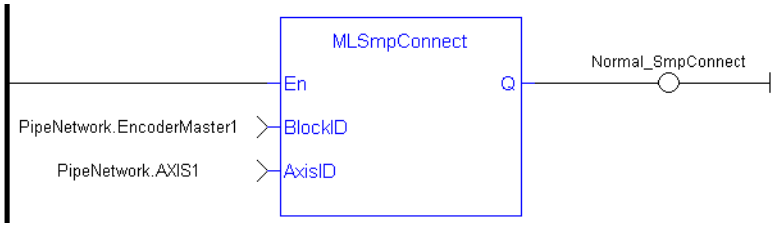
BOOL

Example

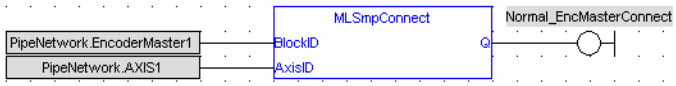
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.15.2 MLSmpConECAT

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.

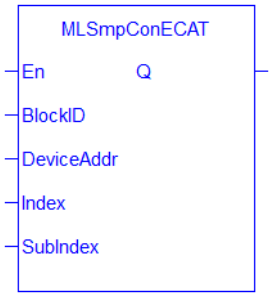


Figure 1-41: MLSmpConECAT function

Related Function Blocks

"MLSmpConnect" (➜ p. 275), "MLSmpInit" (➜ p. 278)

Arguments

Input

BlockID	**Description**	ID Name of the Sampler function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
DeviceAddr	**Description**	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.
	Data type	INT

	Range	—
	Unit	n/a
	Default	—
Index	**Description**	The CoE index of the object to be connected with the Sampler block.
	Data Type	UINT
	Range	—
	Unit	n/a
	Default	—
SubIndex	**Description**	The CoE sub-index of the object to be connected with the Sampler block.
	Data Type	UINT
	Range	—
	Unit	n/a
	Default	—

Output

Default (.Q)	**Description**	Function block is operational
	Data type	BOOL
	Unit	n/a

Return Type

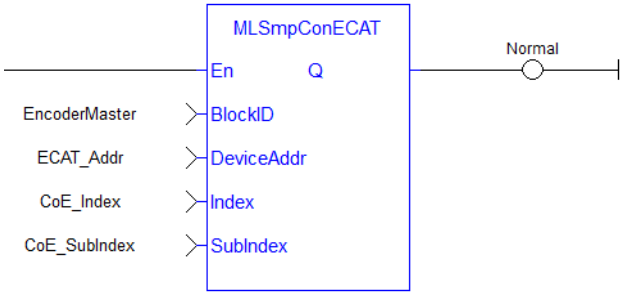
BOOL

Example

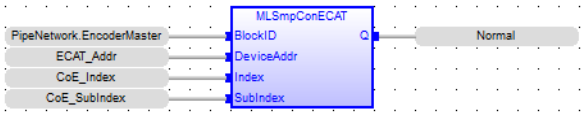
Structured Text

```
MLSmpConECAT( EncoderMaster(100), ECAT_Addr(1001), CoE_Index(5), CoE_
SubIndex(10) );
```

Ladder Diagram



Function Block Diagram



2.1.15.3 MLSmpConnectEx

Description

Connect a sampler to the specified data source.

NOTE

This function was deprecated in KAS v2.9. Please use either "MLSmpConECAT" (➜ p. 276), "MLSmpConPLCAxis" (➜ p. 279), or "MLSmpConPNAxis" (➜ p. 281) instead.

2.1.15.4 MLSmpInit

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.

Related Function Blocks

"MLSmpConnect" (➜ p. 275), "MLSmpConECAT" (➜ p. 276), "MLSmpConPLCAxis" (➜ p. 279), "MLSmpConPNAxis" (➜ p. 281)

Arguments

Input

BlockID	**Description**	ID Name of the SMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
SamplingPeriod	**Description**	period that the device is sampled
	Data type	LREAL
	Range	0.25 to ?
	Unit	millisecond
	Default	1.0
Mode	**Description**	Sampled output can be either position or velocity
	Data type	DINT
	Range	[1 , 2] Position or Speed
	Unit	n/a
	Default	position
InputModuloPosition	**Description**	Period of the input signal. This should be set equal to 2^{32} (4294967296).
	Data type	LREAL

	Range	—
	Unit	User unit
	Default	360.0
OutputModuloPosition	**Description**	Period of the output signal
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	360.0

Output

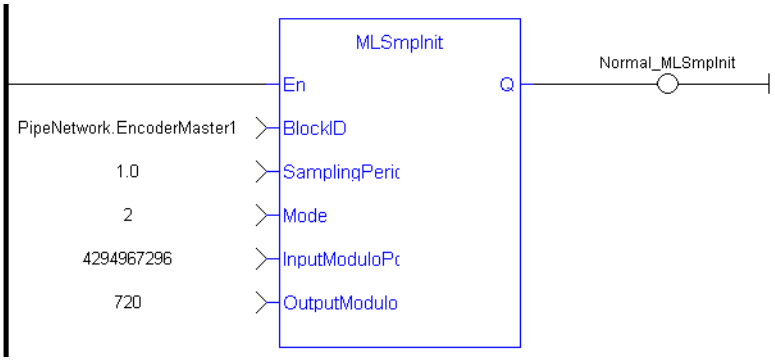
Default (.Q)	**Description**	Smp Block successfully initiated
	Data type	BOOL
	Unit	n/a

Example

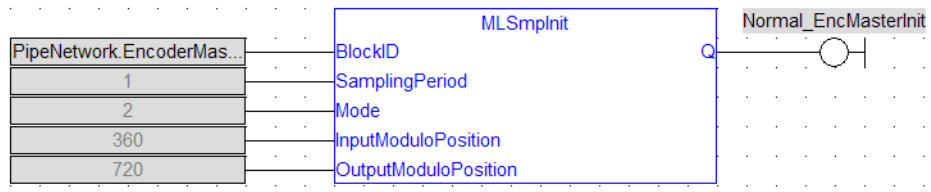
Structured Text

```
      MLSmpInit( PipeNetwork.EncoderMaster1, SamplingPeriod(*LREAL*),
Mode(*DINT*), InputModuloPosition(*LREAL*), OutputModuloPosition(*LREAL*)
) );
```

Ladder Diagram



Function Block Diagram



2.1.15.5 MLSmpConPLCAxis

Description

This function connects a sampler block to a specific variable from a PLCOpen Axis.

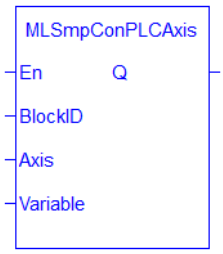


Figure 1-42: MLSmpConPLCAxis function

Related Function Blocks

Arguments

Input

BlockID	**Description**	ID Name of the SMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
AxisID	**Description**	Name of a declared instance of the AXIS_REF library function (for more details, click here....)
	Data Type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Variable	**Description**	Variable to be connected to. You need to use one of the following Internal Defines: • MC_ACTUAL_POSITION • MC_COMMAND_POSITION • MC_NORMAL_COMMAND_POSITION • MC_SUPERIMPOSED_COMMAND_POSITION • MC_PHASE_COMMAND_POSITION
	Data Type	UINT
	Range	n/a (use available macros)
	Unit	n/a
	Default	—

Output

Default (.Q)	**Description**	Function block is operational
	Data type	BOOL
	Unit	n/a

Return Type

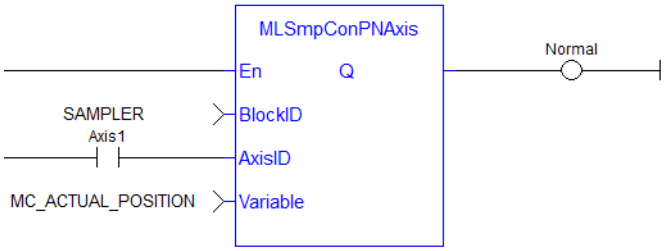
BOOL

Example

Structured Text

```
MLSmpConPLCAxis( PipeNetwork.SAMPLER, Axis1, MC_ACTUAL_POSITION);
```

Ladder Diagram



Function Block Diagram

2.1.15.6 MLSmpConPNAxis

Description

This function connects a sampler block to a specific variable from a PipeNetwork Axis.

Figure 1-43: MLSmpConPNAxis function

Related Function Blocks

Arguments

Input

BlockID	**Description**	ID Name of the SMP function block in the Pipe Network
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
AxisID	**Description**	ID Name of the Axis the sampler is connected to
	Data Type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a

	Default	—
Variable	**Description**	Variable to be connected to. You need to use one of the following Internal Defines : • ML_PIPE_POSITION • ML_REFERENCE_POSITION • ML_GENERATOR_POSITION • ML_ACTUAL_POSITION • ML_FEEDBACK_POSITION • ML_SECOND_FEEDBACK_POSITION • ML_ACTUAL_VELOCITY • ML_ACTUAL_TORQUE • ML_FOLLOWING_ERROR • ML_CURRENT_POSITION
	Data Type	UINT
	Range	n/a (use available macros)
	Unit	n/a
	Default	—

Output

Default (.Q)	**Description**	Function block is operational
	Data type	BOOL
	Unit	n/a

Return Type

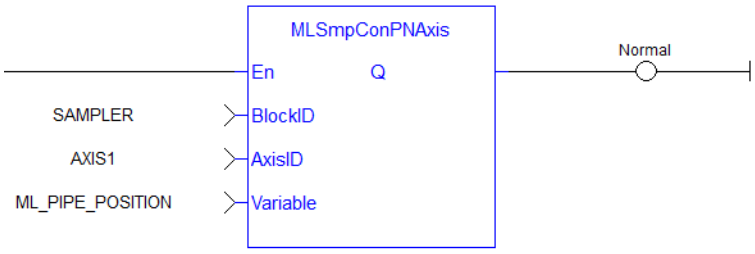
BOOL

Example

Structured Text

```
MLSmpConPNAxis( PipeNetwork.SAMPLER , PipeNetwork.AXIS1, ML_PIPE_POSITION
);
```

Ladder Diagram



Function Block Diagram

2.1.16 Motion Library - Synchronizer

TIP

For usage example about Synchronizer Functions, "" (➜ p. 290)

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

2.1.16.1 MLSyncInit

Description

Initializes a synchronizer Pipe Block. Returns TRUE if the function succeeded.

This FB is automatically created in the compiled code of a Pipe Network.

This function block is part of the MLPN_CREATE_OBJECT to initialize the Pipe Network. It is called at the beginning of an application program with the function call:

```
PipeNetwork(MLPN_CREATE_OBJECTS);
```

Arguments

Input

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

Output

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

Related Functions

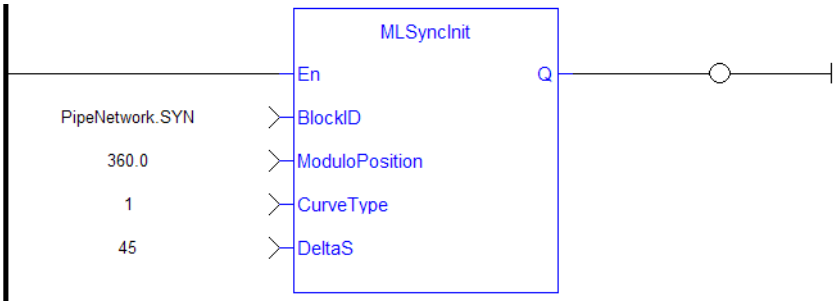
MLSyncWriteDeltaS

Example

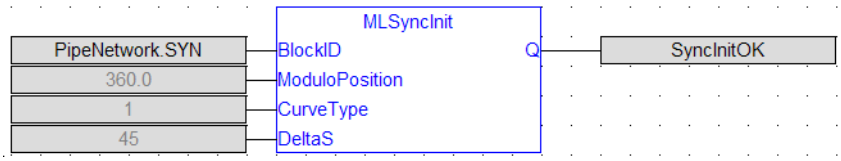
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.16.2 MLSyncReadDeltaS

Description

Gets the output phasing value of a synchronizer block. Output phasing is the distance or the slope the output takes to synchronize with the input when MLSyncStart Block is executed (see "Get Output Phasing after MLSyncStart" (➜ p. 285)). 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. 285)).

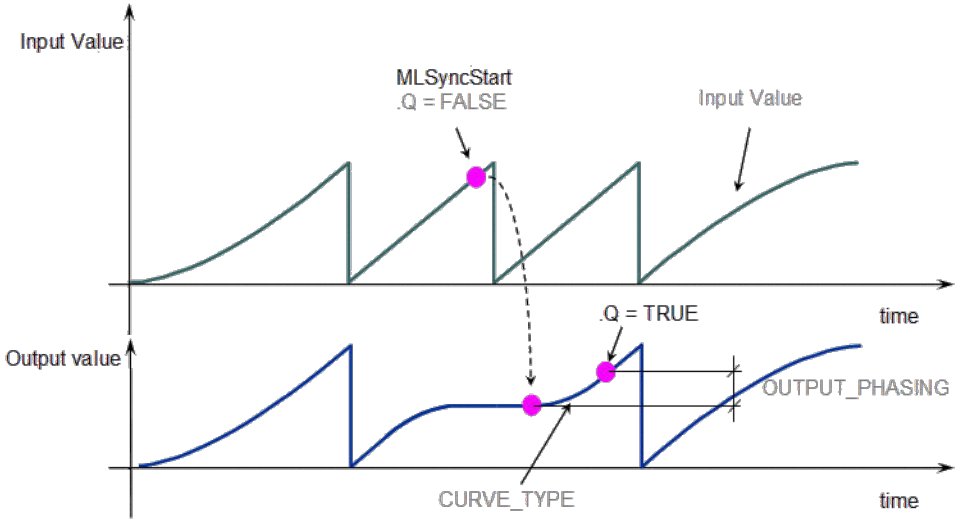


Figure 1-44: Get Output Phasing after MLSyncStart

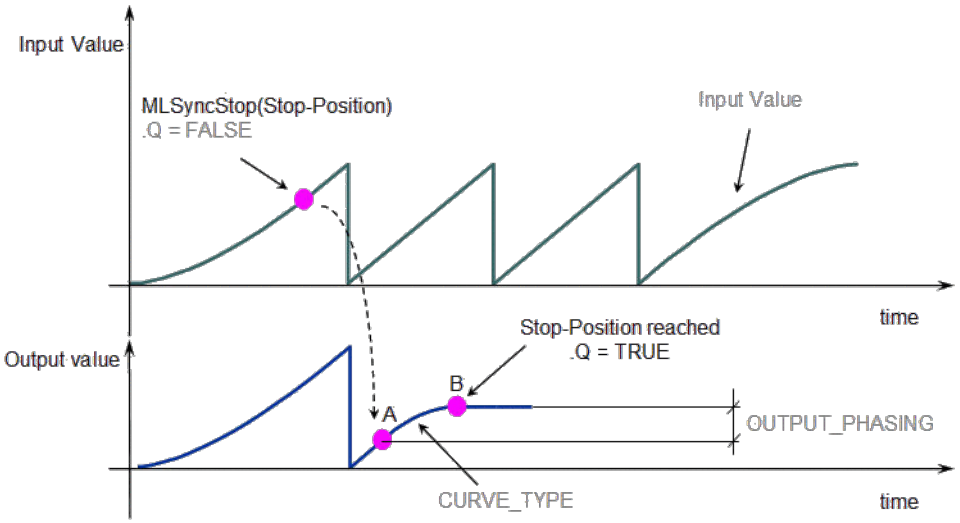


Figure 1-45: Get Output Phasing after MLSyncStop

Arguments

Input

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

Output

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

Related Functions

MLSyncWriteDeltaS

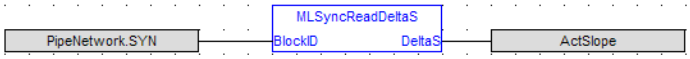
Example

Structured Text

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

Ladder Diagram

Function Block Diagram



2.1.16.3 MLSyncStart

Description

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

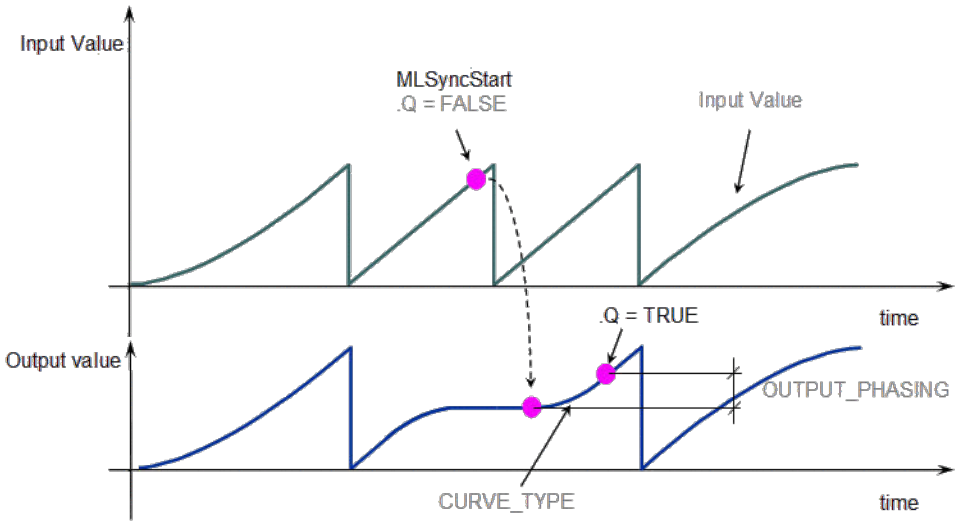


Figure 1-46: MLSyncStart

Arguments

Input

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

	Default	—

Output

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

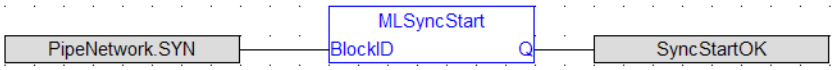
Example

Structured Text

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

Ladder Diagram

Function Block Diagram



2.1.16.4 MLSyncStop

Description

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

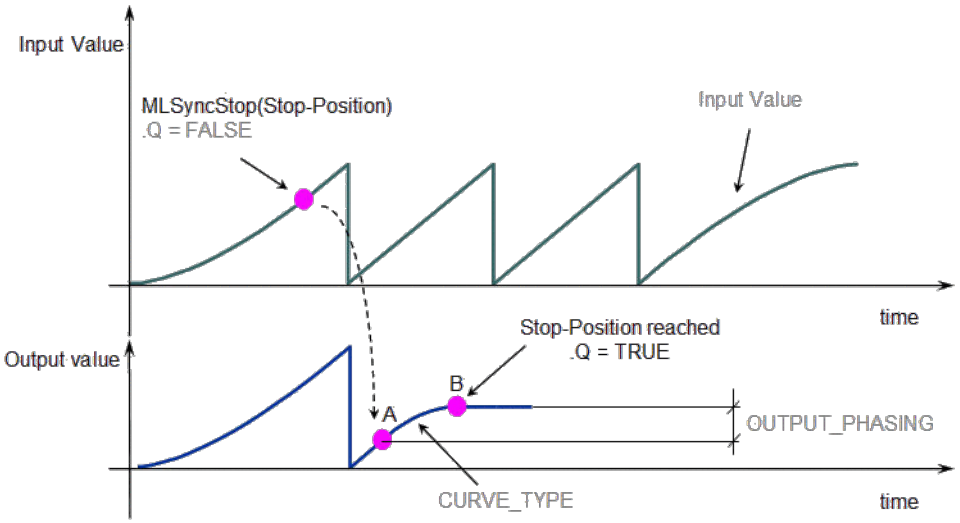


Figure 1-47: MLSyncStop

Arguments

Input

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

Output

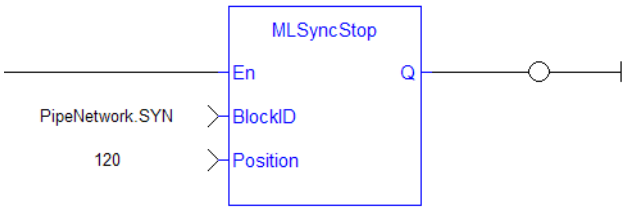
Default (.Q)	Description	Function Block Execute Successfully
	Data type	BOOL
	Unit	n/a

Example

Structured Text

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

Ladder Diagram



Function Block Diagram

2.1.16.5 MLSyncWriteDeltaS

Description

Set the output phasing value of a synchronizer block. Returns TRUE if the function succeeded. Output phasing is the distance or the slope the output takes to synchronize with the input when MLSyncStart Block is executed. It also affects the distance or the slope the output takes to desynchronize with the input and come to a stop when MLSyncStop Block is executed.

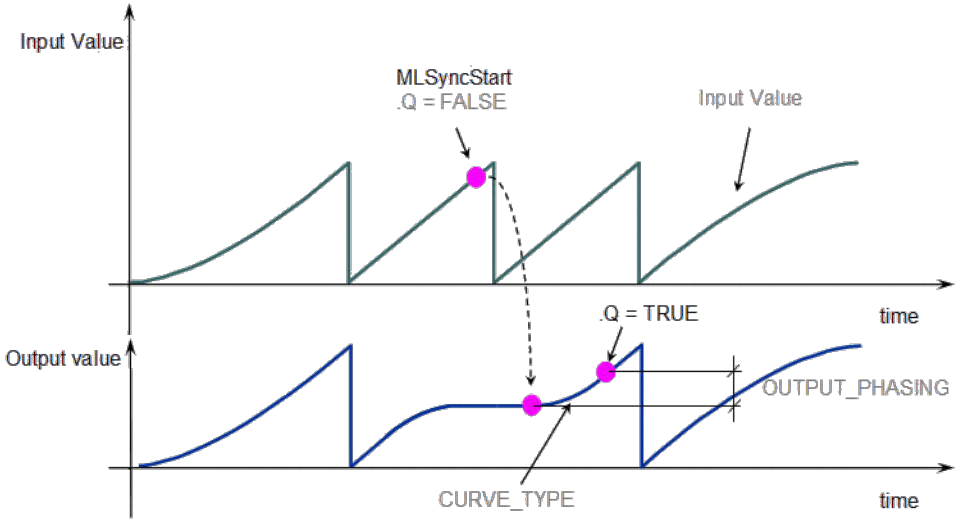


Figure 1-48: Set output phasing after MLSyncStart

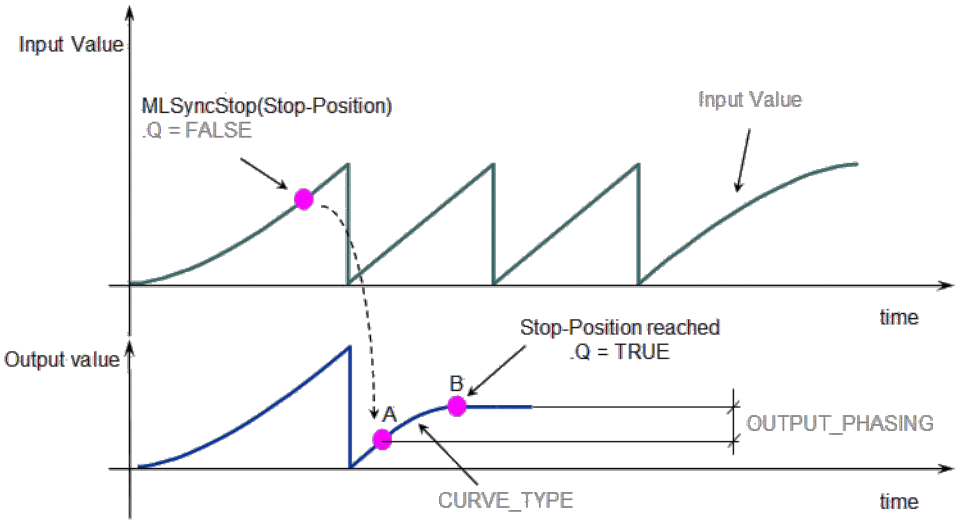


Figure 1-49: Set output phasing after MLSyncStop

Arguments

Input

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

Output

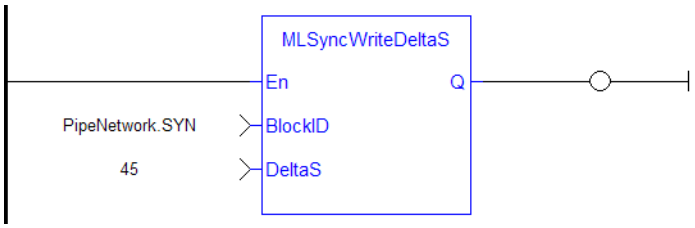
Default (.Q)	Description	Function Block Execute Successfully
	Data type	BOOL
	Unit	n/a

Example

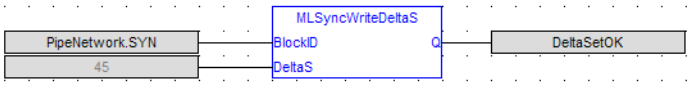
Structured Text

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

Ladder Diagram



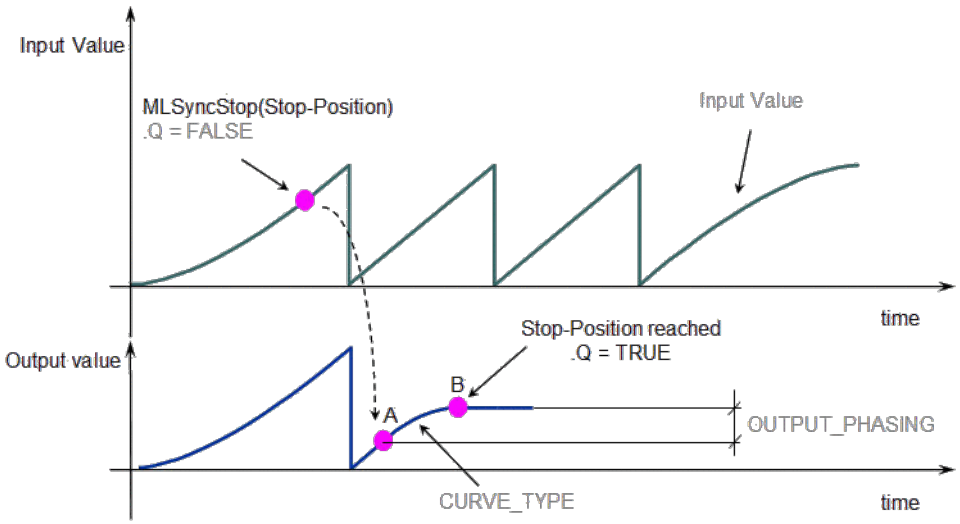
Function Block Diagram



2.1.16.6 Usage example of Synchronizer Functions

When you call the **MLSyncStop** function, the output value is adapted according to the specified Stop-Position (point B).

The OUTPUT_PHASING parameter is used to define point A, where the flow follows a curve in order to smooth the output value.



When you call the **MLSyncStart** function, the output value is adapted to catch up with the input value.

The OUTPUT_PHASING parameter is also used to define a curve in order to smooth the output value.

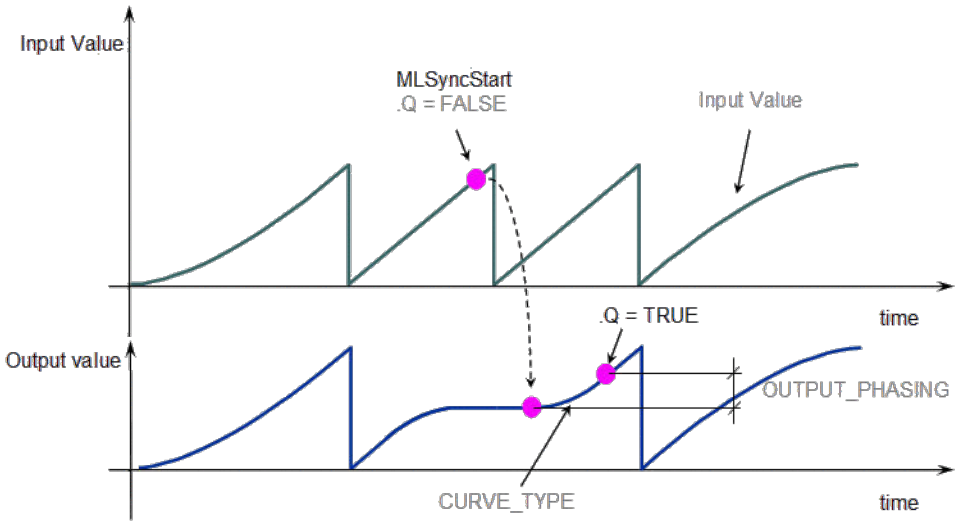


Figure 1-50: Synchronizer Functions Usage

2.1.17 Motion Library - Trigger

TIP

For usage example about Trigger Functions, "" (➜ p. 302)

Name	Description	Return type
MLTrigClearFlag	Clears the flag of an initiated Trigger block	BOOL
MLTrigInit	Initializes a Trigger object	BOOL
MLTrigIsTrigged	Checks if the selected block has been triggered	BOOL
MLTrigReadDelay	Returns the time that the trigger block uses to compensate the delay of the sensor that captures the triggering signal	None
MLTrigReadPos	Returns the position of the block at the moment when it was triggered	None
MLTrigReadTime	Returns the time of the moment where the block was triggered in milliseconds	None
"MLTrigSetEdge" (➜ p. 300)	Sets the edge configuration for a Trigger object	BOOL
MLTrigWriteDelay	Sets the time that the trigger block uses to compensate for the delay introduced by the sensor that captures the triggering signal	BOOL

2.1.17.1 MLTrigClearFlag

Description

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

IMPORTANT

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

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLAxisRstFastIn

MLTrigIsTrigged

MLTrigReadPos

MLTrigReadTime

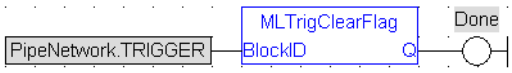
Example

Structured Text

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

Ladder Diagram

Function Block Diagram



2.1.17.2 MLTrigInit

Description

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

The Trigger object monitors a selected Fast Input and captures the time of a rising or falling edge event. With the time and pipe position information the Trigger object extrapolates the axis position when the Fast Input event occurred.

Parameters to enter include the name of the Pipe Block, the Axis where the Fast Input is located, the number of the desired Fast Input, and whether to trigger on the rising or falling edge of the input.

NOTE

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

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLTrigIsTrigged

MLTrigReadPos

MLTrigClearFlag

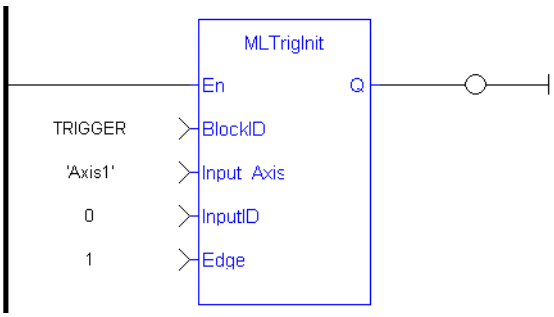
MLAxisRstFastIn

Example

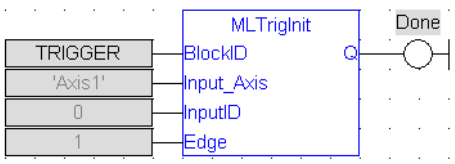
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.17.3 MLTrigIsTrigged

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.

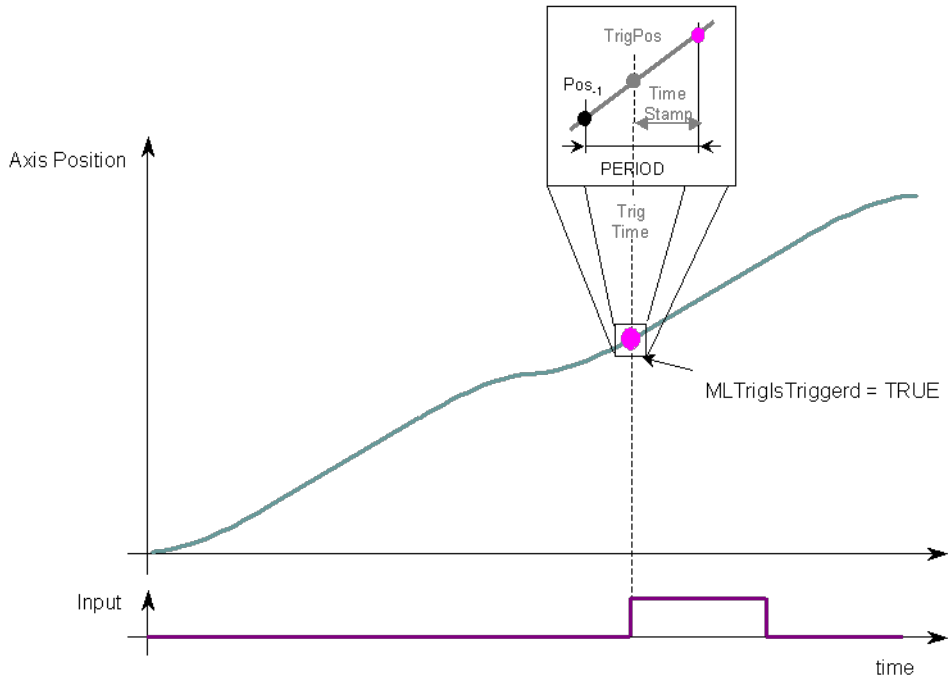


Figure 1-51: MLTrigIsTrigged

NOTE

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

Arguments

Input

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

Output

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

Return Type

BOOL

Related Functions

MLTrigReadPos

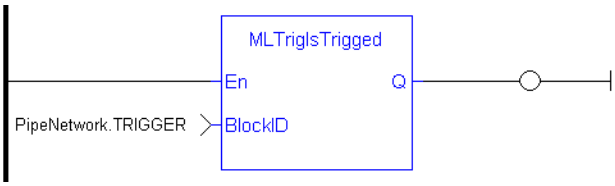
MLTrigReadTime

Example

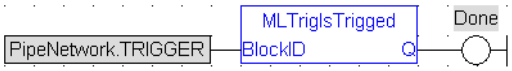
```
//Check if a Trigger Block has been triggered, then save position
```

```
IF MLTrigIsTrigged( PipeNetwork.TRIGGER ) THEN
Trig_Position := MLTrigReadPos( PipeNetwork.TRIGGER );
END_IF
```

Ladder Diagram



Function Block Diagram



2.1.17.4 MLTrigReadDelay

Description

Electronic sensors are not able to respond immediately to a signal. Sensors usually require a certain amount of time to process a change of state in their input signal. This function returns the delay that has been programmed in a trigger block by the MLTrigWriteDelay function to compensate for this reaction time required by the sensor.

Input

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

Output

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

Related Functions

MLTrigWriteDelay

2.1.17.5 MLTrigReadPos

Description

Returns the position of the block at the moment when it was triggered by the Trigger Block's selected Fast Input. This value is only valid when TrigIsTrigged() returns TRUE. The Trigger block extrapolates the output value based on the timestamp of the Fast Input event to provide an accurate position even if the event occurs in the middle of a program cycle.

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

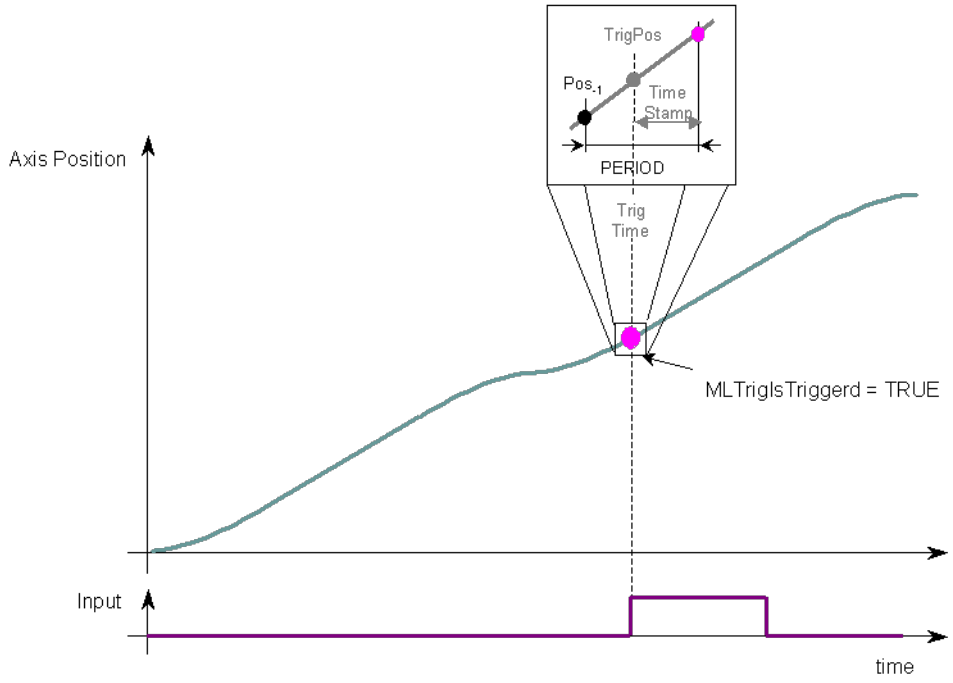


Figure 1-52: MLTrigReadPos

Arguments

Input

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

Output

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

Related Functions

MLTrigIsTrigged

MLTrigReadTime

MLTrigClearFlag

MLAxisRstFastIn

Previous Function Name

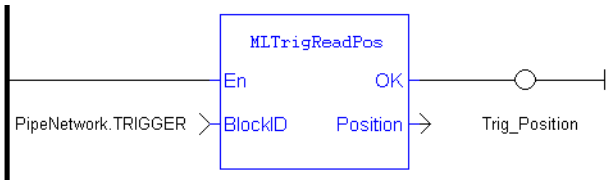
MLTrigGetPos

Example

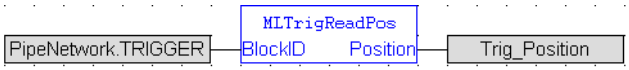
Structured Text

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

Ladder Diagram



Function Block Diagram



2.1.17.6 MLTrigReadTime

Description

Returns the time of the moment where the block was triggered in milliseconds. This value is only valid when TrigIsTrigged() returns TRUE. The output is computed from the timestamp of a Fast Input time event

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

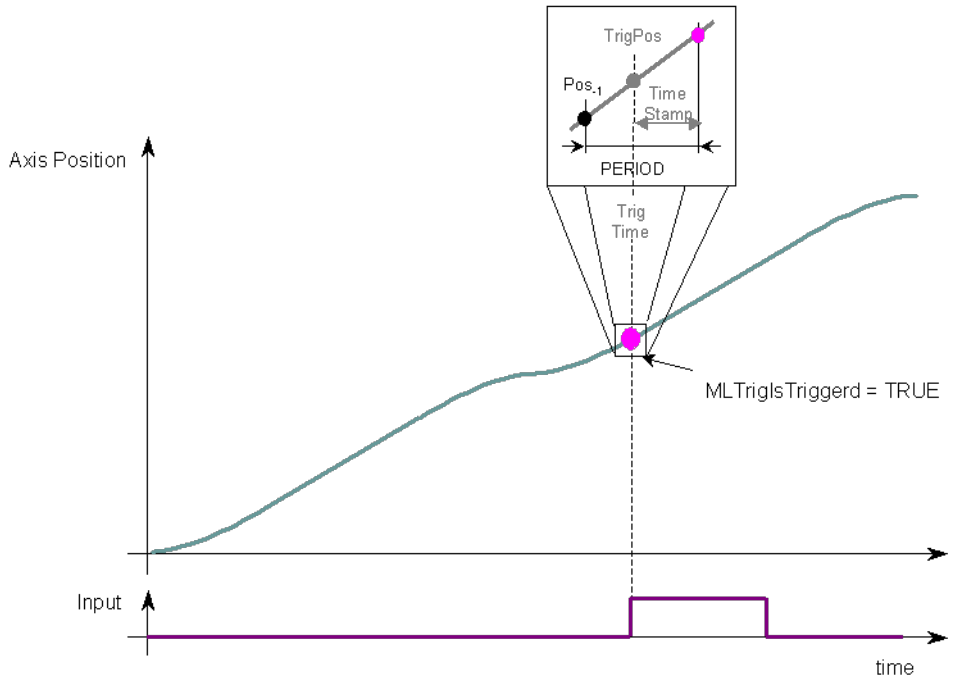


Figure 1-53: MLTrigReadTime

Arguments

Input

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

Output

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

Related Functions

MLTrigIsTrigged

MLTrigReadPos

MLTrigClearFlag

MLAxisRstFastIn

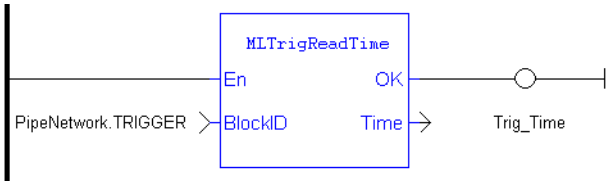
Previous Function Name

MLTrigGetTime

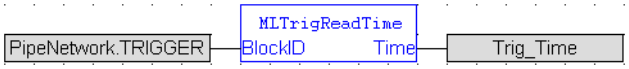
Example

```
//Save time when Fast Input event occurs
Trig_Time := MLTrigReadTime( PipeNetwork.TRIGGER );
```

Ladder Diagram



Function Block Diagram



2.1.17.7 MLTrigSetEdge

Description

Sets the edge configuration (rising, falling, etc.) for a trigger block. This block should be called prior to calling "MLAxisCfgFastIn" (➜ p. 116). Also the value at the Edge input must match the value at MLAxisCfgFastIn's Mode input.

Arguments

Input

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

Output

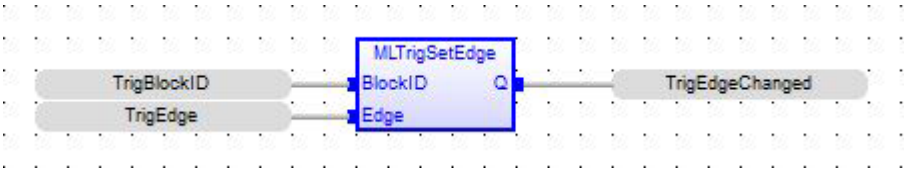
Q	**Description**	True if block executed successfully False if execution is not successful
	Data type	BOOL
	Unit	n/a

Return Type

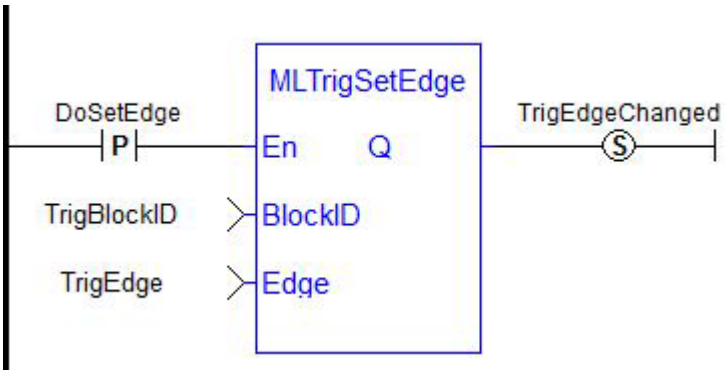
BOOL

Examples

Function Block Diagram



Ladder Diagram



Structured Text

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

2.1.17.8 MLTrigWriteDelay

Description

Electronic sensors are not able to respond immediately to a signal. Sensors usually require a certain amount of time to process a change of state in their input signal. This function allows the trigger block to calculate the exact moment at which a signal was triggered by letting you specify the delay introduced by the sensor.

Input

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

Output

Default (.Q)	**Description**	Returns TRUE if the delay is successfully set
	Data type	BOOL
	Unit	n/a

Return Type

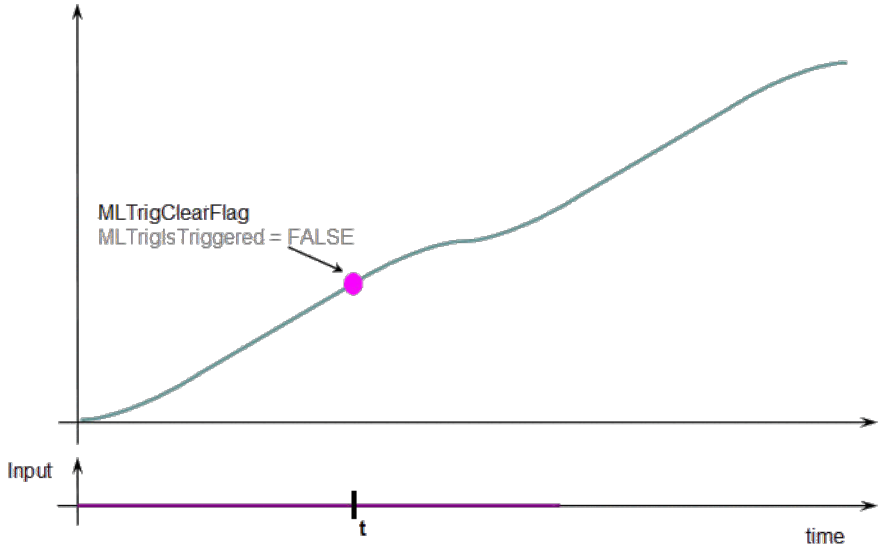
BOOL

Related Functions

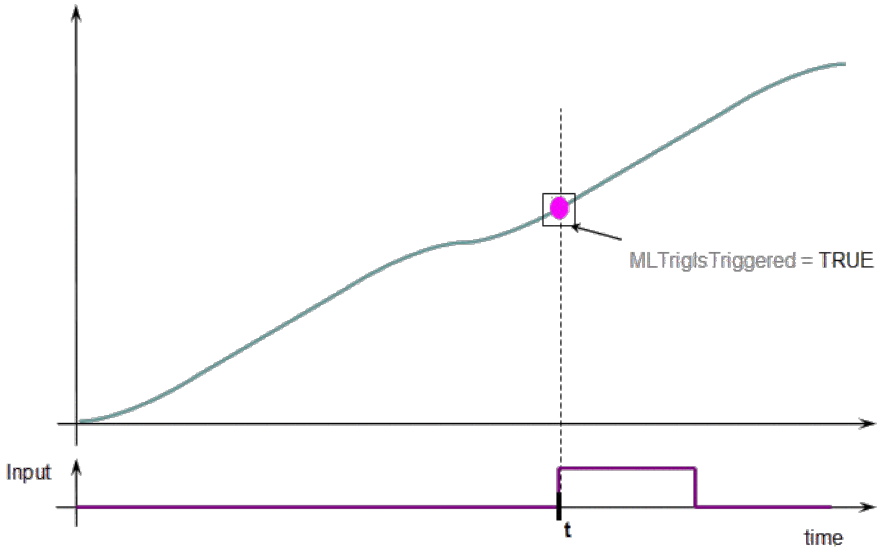
MLTrigReadDelay

2.1.17.9 Usage example of Trigger Functions

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



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



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

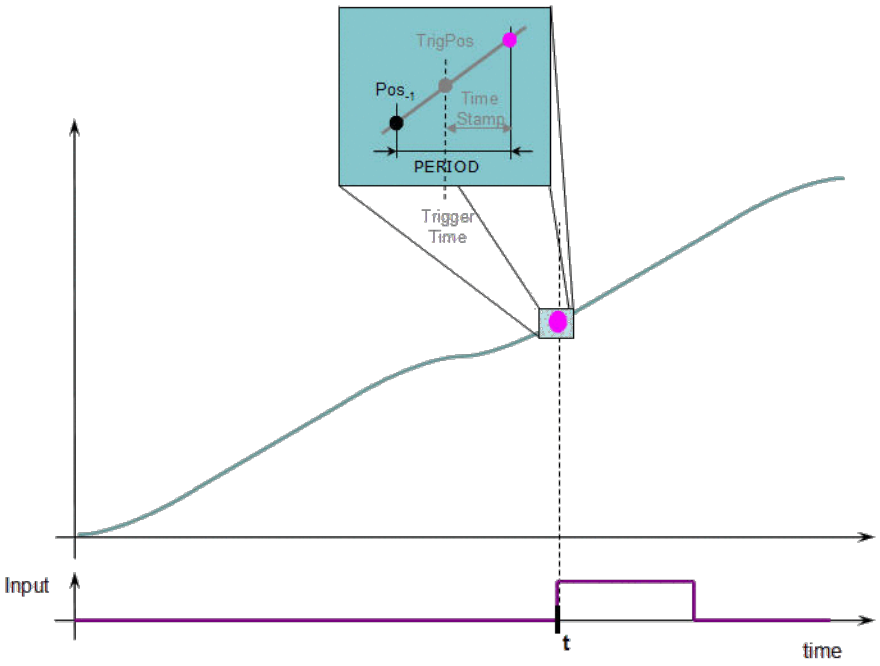


Figure 1-54: Trigger Functions Usage

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
MC_AbortTrigger	Abort MC_TouchProbe
" MC_AddSuperAxis" (➜ p. 408)	Add an axis to the axis's list of assigned, superimposed axes.
MC_CamIn	Performs a slave axis move based on the Cam Table
MC_CamOut	Disengages the slave axis from a MC_CamIn move
"MC_CamResumePos" (➜ p. 367)	Returns the slave axis position for resuming an MC_CamIn move
"MC_CamStartPos" (➜ p. 369)	Returns the slave axis position for starting an MC_CamIn move
MC_CamTblSelect	Defined to read and initialize the specified profile
MC_ClearFaults	Clear Drive Faults
MC_CreateAxis	Creates a PLCopen Axis
MC_EStop	Performs a Emergency stop
"MC_ErrorDescription" (➜ p. 411)	Converts the PLCopen error IDs into message strings.
MC_GearIn	Performs a slave axis move based on the ratio
MC_GearInPos	Performs a slave axis move based on the ratio
MC_GearOut	Disengages the slave axis from a MC_GearIn or MC_GearInPos move
MC_Halt	Decelerates an axis to zero velocity

Name	Description
MC_InitAxis	Initializes a PLCopen Servo Axis' data
MC_MachRegist	Runs Mark-to-Machine registration
MC_MarkRegist	Runs Mark-to-Mark registration
MC_MoveAbsolute	Performs a single-axis move to a specified endpoint position
MC_MoveAdditive	Performs a single-axis move for a specified distance from the end-point of the previous move
"MC_MoveRelative" (➜ p. 346)	Performs a single-axis move for a specified distance
MC_MoveSuperimp	Performs a single-axis move which is superimposed upon the active move
MC_MoveVelocity	Performs a single-axis non-ending move at a specified velocity
MC_Phasing	Performs a master position phase shift for the slave axis
MC_Power	Requests to enable the drive and close the loop, or disable the drive and open the loop
MC_ReadActPos	Reads the actual position of the axis
MC_ReadActVel	Reads the actual velocity of the axis
MC_ReadAxisErr	Returns the error status of the specified axis
MC_ReadBoolPar	Returns the value of the specified Boolean axis parameter
MC_ReadParam	Returns the value of the specified axis parameter
MC_ReadStatus	Returns the state of the specified axis
MC_Reference	Defines the position at the reference location for PLCopen Axis
" MC_RemSuperAxis" (➜ p. 409)	Remove an axis from the axis's list of assigned, superimposed axes.
MC_ResetError	Resets the errors of the specified axis
MC_SetOverride	Writes velocity and acceleration override factors
MC_SetPosition	Deprecated by "MC_SetPos" (➜ p. 394)
"MC_SetPos" (➜ p. 394)	Sets a new axis position
MC_Stop	Aborts the active move, removes the next move from the queue, performs a controlled stop, and switches the axis to Stopping state
MC_StopRegist	Turns off registration for the specified axis
MC_SyncSlaves	Specifies synchronized slaves
MC_TouchProbe	Arm a Fast Input and capture an axis position
MC_WriteBoolPar	Writes the specified axis Boolean parameter
MC_WriteParam	Writes the specified axis parameter

2.2.1 Control Functions

This set of functions provide general controls to drives and axes.

2.2.1.1 MC_ClearFaults (Function)

Description

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

NOTE

The condition causing the drive fault has to be corrected before calling this function. If the fault condition still exists when this function is called, this function sends a request to the drive but the drive faults remain.

This function does **not** reset axis errors. MC_ResetError is required to reset axis errors.

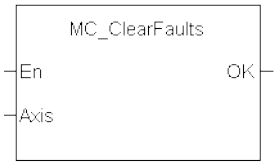


Figure 1-55: MC_ClearFaults

Arguments

Input

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

Output

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

Usage

Upon the positive transition of the EN input, this function requests a Fault Reset of the Drive for the Axis defined in the axis input of this function.

Related Functions

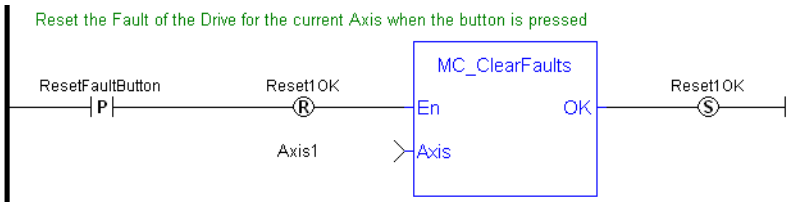
MC_ResetError

Example

Structured Text

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

Ladder Diagram



2.2.1.2 MC_CreateAxis

Description

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

IMPORTANT

MC_CreateAxis must be called between "MLMotionInit" (➜ p. 425) and "MLMotionStart" (➜ p. 426).

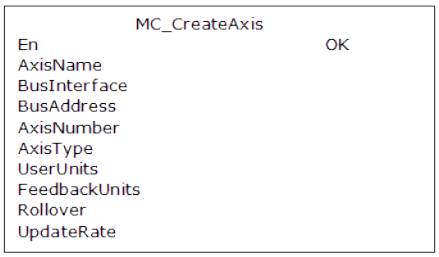


Figure 1-56: MC_CreateAxis

Arguments

Input

En	**Description**	Requests to create a PLCopen axis
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxisName	**Description**	Axis name
	Data type	STRING
	Range	—
	Unit	n/a
	Default	—
BusInterface	**Description**	Bus interface identifier: “EtherCATDriver” = EtherCAT interface “MSBusDriver” = KAS Simulator interface
	Data type	STRING
	Range	—
	Unit	n/a
	Default	—

BusAddress	**Description**	Address of the drive on the bus
	Data type	DINT
	Range	bus dependent
	Unit	n/a
	Default	—
AxisNumber	**Description**	Axis number
	Data type	UINT
	Range	[1,256]
	Unit	n/a
	Default	—
AxisType	**Description**	Axis type: 0 (MC_AXIS_TYPE_SERVO) denotes servo 1 (MC_AXIS_TYPE_DIGITIZING) denotes digitizing 2 (MC_AXIS_TYPE_VIRTUAL_SERVO) denotes virtual servo
	Data type	USINT
	Range	[0,1]
	Unit	n/a
	Default	—
UserUnits	**Description**	User unit portion of the user unit/feedback unit ratio
	Data type	UDINT
	Range	[1,4294967296]
	Unit	User unit
	Default	—
FeedbackUnits	**Description**	Feedback unit portion of the user unit/feedback unit ratio
	Data type	UDINT
	Range	[1,4294967296]
	Unit	feedback units. **Note:** *The FeedbackUnits input must be a power of 2.* If input FeedbackUnits is not a power of two, the axis will not be created, and the OK output will be FALSE.
	Default	—
Rollover	**Description**	Rollover position (0 = no rollover)
	Data type	UDINT
	Range	[0, 4294967296]
	Unit	User unit
	Default	—
UpdateRate	**Description**	Servo update rate (0, 1, and 2 are reserved for future enhancements) 3 = 125 µsec 4 = 250 µsec 5 = 500 µsec 6 = 1 msec 7 = 2 msec 8 = 4 msec 9 = 8 msec
	Data type	UINT
	Range	[3,9]

	Unit	n/a
	Default	—

Output

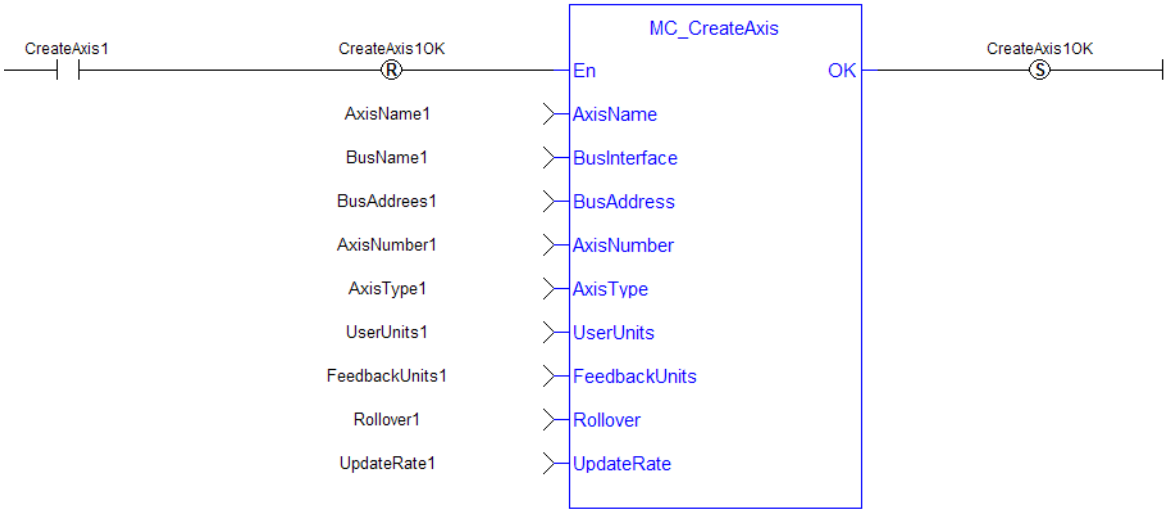
OK	**Description**	Indicates the axis has been created
	Data type	BOOL

Example

Structured Text

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

Ladder Diagram



2.2.1.3 MC_EStop

Description

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

Figure 1-57: MC_EStop

Arguments

Input

En	Description	A positive transition of this input causes an E-stop on the specified axis
	Data type	BOOL

	Range	0, 1
	Unit	n/a
	Default	—
Axis	Description	Axis identifier
	Data type	AXIS_REF
	Range	1-256 The AXIS_NUM element of the AXIS_REF structure must be in the range [1-256]
	Unit	n/a
	Default	—

Output

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

Usage

Call MC_EStop to generate an emergency stop for an axis.

Call MC_ResetError to reset the emergency stop.

Related Functions

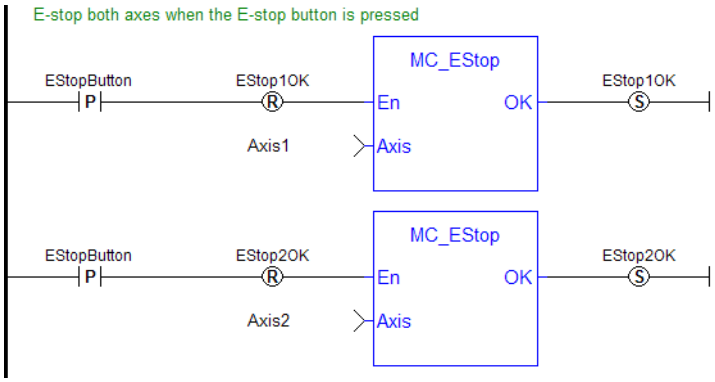
MC_ResetError

Example

Structured Text

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

Ladder Diagram



2.2.1.4 MC_InitAxis (Function)

Description

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

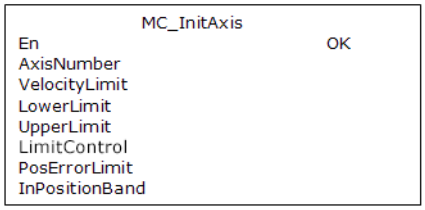


Figure 1-58: MC_InitAxis

Arguments

Input

En	Description	Request to initialize a PLCopen servo axis
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxisNumber	Description	Servo axis number
	Data type	UINT
	Range	[1,256]
	Unit	none
	Default	—
VelocityLimit	Description	Velocity limit
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
LowerLimit	Description	Lower position limit
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
UpperLimit	Description	Upper position limit
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
LimitControl	Description	Establishes how position limits are applied 0 = apply position limits 1 = ignore position limits 2 = ignore limits until referenced
	Data type	UINT
	Range	[0,2]

	Unit	n/a
	Default	—
PosErrorLimit	Description	Position error limit – when the Position Error (command position – actual position) exceeds this value, an E-stop is generated
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
InPositionBand	Description	In-position bandwidth – when the axis actual position is within this distance from its programmed endpoint, the axis is considered “in position”
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

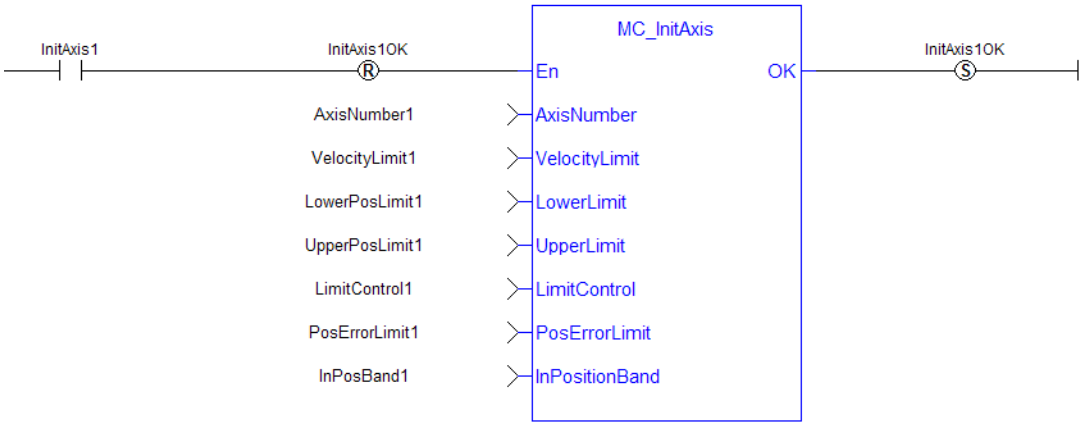
OK	Description	Indicates the initialization is complete
	Data type	BOOL

Example

Structured Text

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

Ladder Diagram



2.2.1.5 MC_Power

Description

This function block requests to enable the drive and close the position loop, or disable the drive and open the position loop. The Status output indicates the state of the position loop. If the position loop is open, the axis command position is set to the actual position of the axis and tracks the actual position.

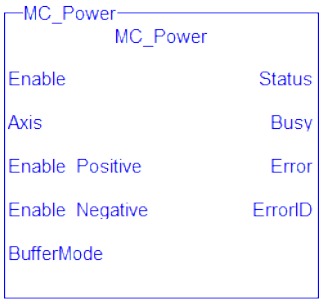


Figure 1-59: MC_Power

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.

Arguments

Input

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

BufferMode	Description	Unused
	Data type	SINT
	Range	[0]
	Unit	n/a
	Default	—

Output

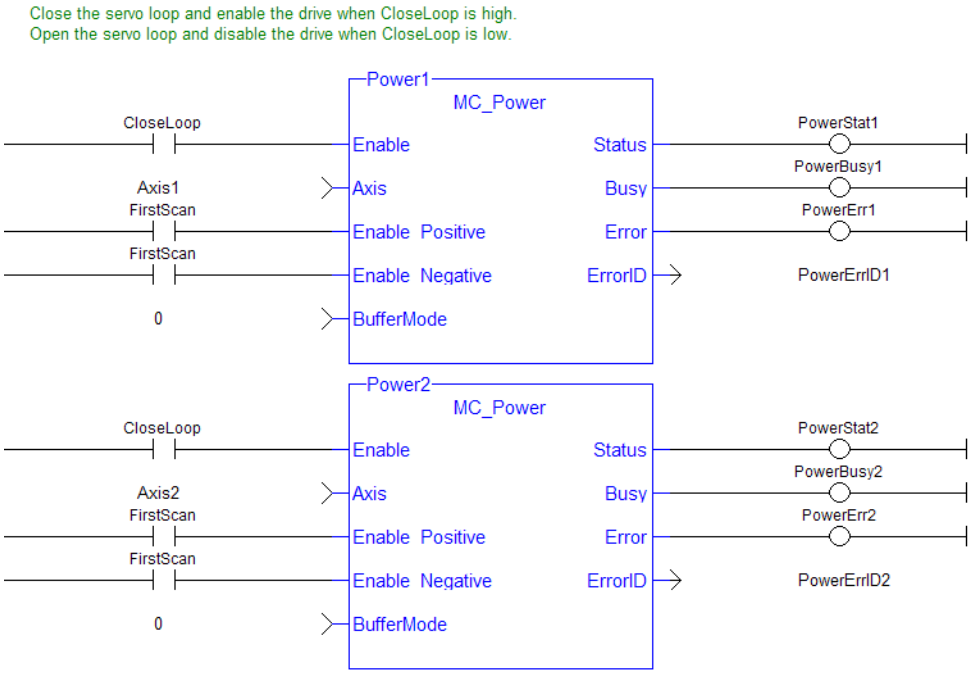
Status	Description	Indicates the enabled/disabled state of the drive
	Data type	BOOL
Busy	Description	for future enhancement – always false
	Data type	BOOL
Error	Description	Indicates an invalid input was specified
	Data type	BOOL
ErrorID	Description	Indicates the error if Error output is set to TRUE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

Example

Structured Text

```
(* MC_Power ST example *)
Inst_MC_Power( CloseLoopReq, Axis1, TRUE, TRUE, 0 );
//Inst_MC_Power is an instance of MC_Power function block
DriveIsOn := Inst_MC_Power.Status; //store the Status output into a
user defined variable
```

Ladder Diagram



2.2.1.6 MC_ErrorDescription

This function converts the PLCopen error IDs into message strings which can be used for display or logging.

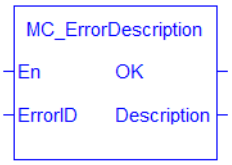


Figure 1-60: MC_ErrorDescription Function Block

Arguments

Inputs

En	**Description**	If True, then this function will convert the Error Id into a string message
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
ErrorID	**Description**	Error ID generated from a PLCopen Function Block. See PLCopen Function Block ErrorID Output for output details.
	Data type	INT
	Range	0,69
	Unit	n/a
	Default	—

Outputs

OK	**Description**	If True, then the command completed successfully.
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Description	**Description**	String error description
	Data type	STRING
	Range	—
	Unit	n/a
	Default	—

Examples

Structured Text

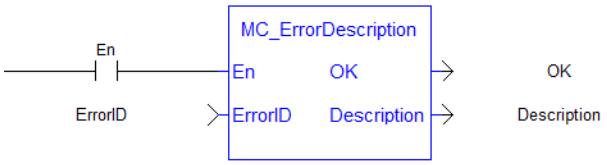
```
Description:= MC_ErrorDescription(ErrorID);
```

IL

Not applicable

Function Block

Ladder Diagram



2.2.1.7 MC_ResetError

Description

The function MC_ResetError resets the errors of a specified axis.

This function performs in sequence the following tasks:

- It sends a request to the drive to clear any drive faults that exists
- Then it resets the axis errors

NOTE

The condition causing the axis error has to be corrected before calling this function. The axis error still remains until the error condition exists when this function is called.

See also transition 15 in the status machine of the CANopen protocol.

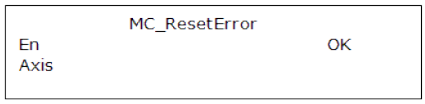


Figure 1-61: MC_ResetError

Arguments

Input

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

Output

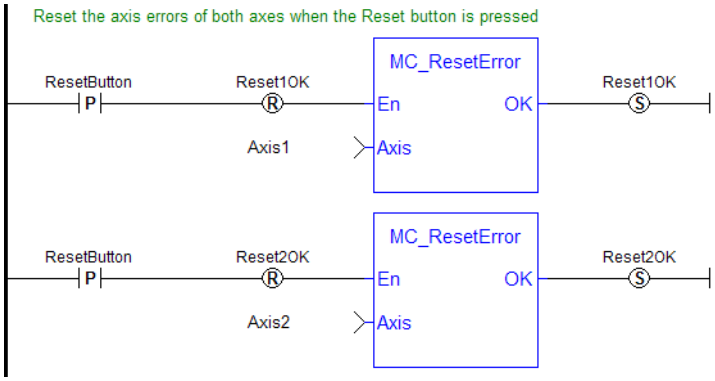
OK	**Description**	Indicates the function has completed successfully
	Data type	BOOL

Example

Structured Text

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

Ladder Diagram



2.2.1.8 MC_Stop

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.

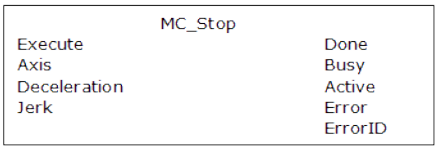


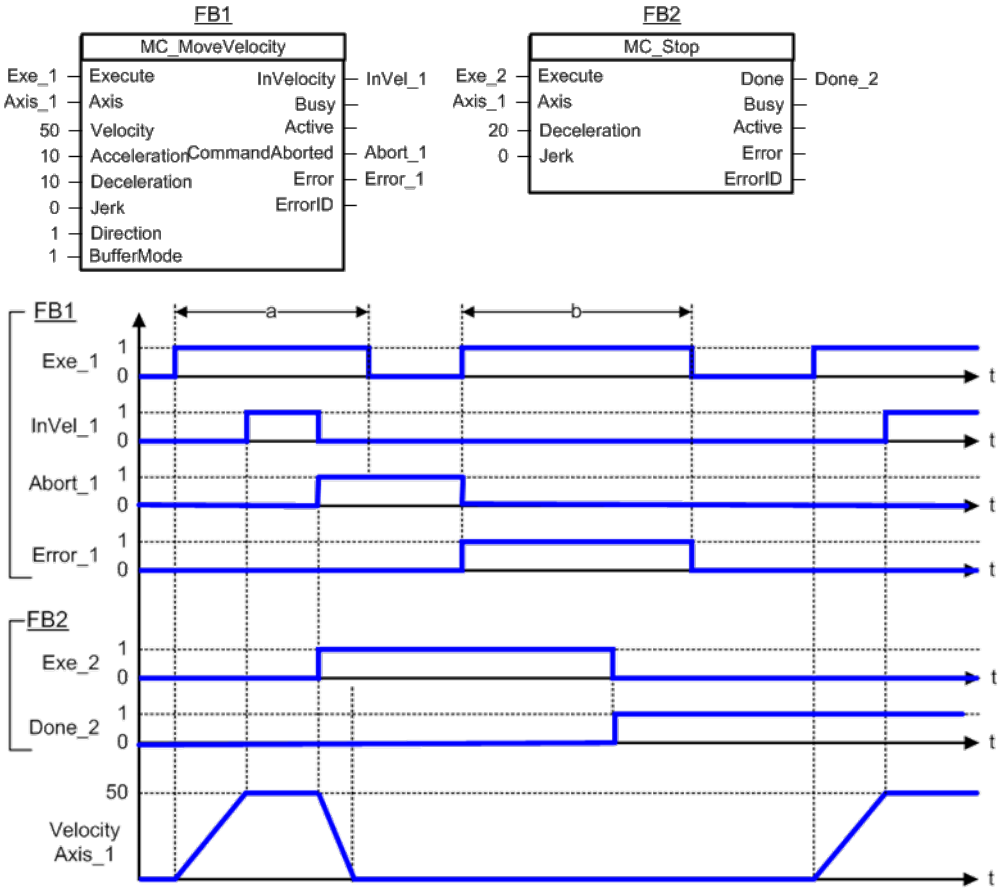
Figure 1-62: MC_Stop

Time Diagram

The example below shows the behavior of the combination of a MC_Stop FB with a MC_MoveVelocity FB.

- A rotating axis is ramped down with FB2 MC_Stop
- The axis rejects motion commands as long as MC_Stop parameter “Execute” = TRUE

FB1 MC_MoveVelocity reports an error indicating the busy MC_Stop command.



Arguments

Input

Execute

Description	Requests to stop the axis. It can be held high to prevent any other moves from being queued
Data type	BOOL
Range	0, 1
Unit	n/a

	Default	—
Axis	**Description**	Name of a declared instance of the AXIS_REF library function.
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Deceleration	**Description**	Trapezoidal: Deceleration rate S-curve: Maximum deceleration
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Jerk	**Description**	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec^3
	Default	—

Output

Done	**Description**	Indicates the axis has reached zero velocity AND the Execute input is low
	Data type	BOOL
Busy	**Description**	High from the time the Execute input goes high until the axis reaches zero velocity AND the Execute input is low
	Data type	BOOL
Active	**Description**	High from the time the MC_Stop move becomes the active move, until the axis reaches zero velocity AND the Execute input is low
	Data type	BOOL
Error	**Description**	Indicates an invalid input was specified
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE
	Data type	INT

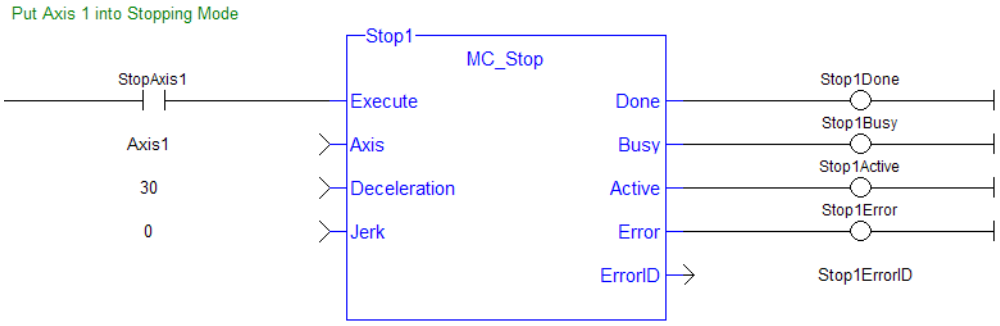
Example

Structured Text

```
(* MC_Stop ST example *)

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

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

```
StopActive := Inst_MC_Stop.Active;      //store the Active output into a
user defined variable

StopError := Inst_MC_Stop.Error;        //store the Error output into a
user defined variable
```

Ladder Diagram



2.2.2 I/O Functions

This set of functions provides I/O control over TouchProbe functions.

2.2.2.1 MC_AbortTrigger (Function Block)

Description

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

Arguments

Input

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

TriggerInput	Description	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: INT TriggerInput.InputID 0 = Capture Engine 0 1 = Capture Engine 1 Range is [0,1] For information on configuring the capture engines, refer to AKD Capture Engine Configuration. INT TriggerInput.Direction 1 = rising edge 2 = falling edge Range is [1,2] INT TriggerInput.TrigID is the axis number of the input. 0 indicates that the trigger axis is to be the same as Axis.AXIS_NUM. Range is [0,256]
	Data type	TRIGGER_REF
	Range	See Description above
	Unit	n/a
	Default	—

Output

Done	Description	Function block has completed
	Data type	BOOL
Busy	Description	Indicates the function block is currently executing
	Data type	BOOL
Error	Description	Indicates the function block did not complete due to an error. The ErrorID output indicates the type of error when this output is high
	Data type	BOOL
ErrorID	Description	When the Error output is high, this output indicates the type of error. When the Error output is low, this output is undefined
	Data type	INT

Usage

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

Related Functions

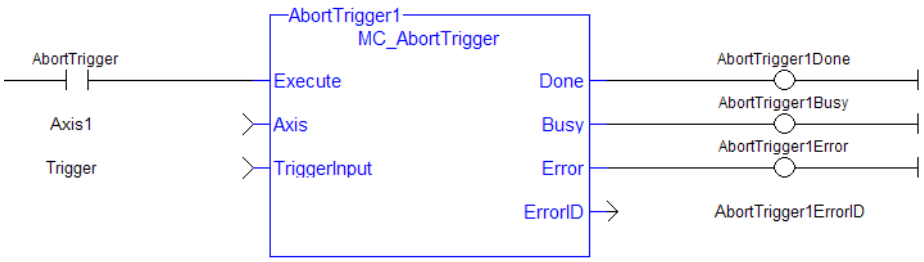
MC_TouchProbe

Example

Structured Text

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

Ladder Diagram



2.2.2.2 MC_TouchProbe

Description

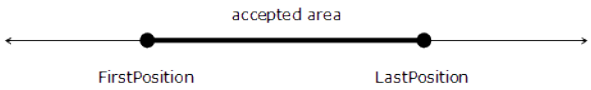
This function block arms a Fast Input and returns the latched position when the Fast Input event occurs. This function block causes no motion.

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.

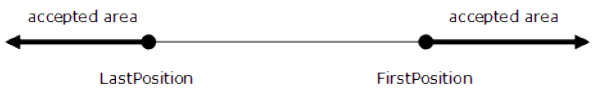
If First Position <= LastPosition, the window in which a Fast Input is accepted is:

FastInputPosition >= FirstPosition AND FastInputPosition <= LastPosition.



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

FastInputPosition >= FirstPosition OR FastInputPosition <= LastPosition.



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

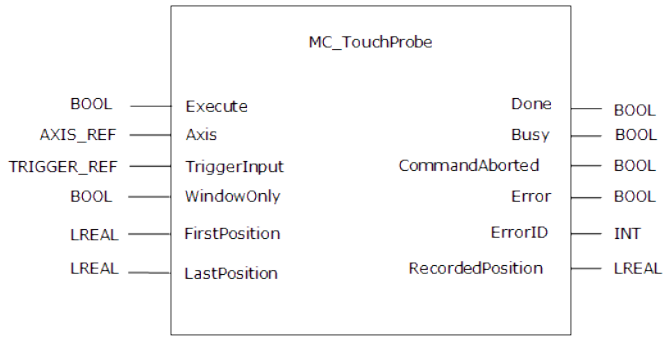


Figure 1-63: MC_TouchProbe

Arguments

Input

Execute

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

Axis

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

TriggerInput

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

WindowOnly	**Description**	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
	Data type	BOOL
	Range	—
	Unit	n/a
	Default	—
FirstPosition	**Description**	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
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
LastPosition	**Description**	See the function block Description above for an explanation of how this input and the FirstPosition input define the window. This input is only applicable when the WindowOnly input is high. If the WindowOnly input is low, this input is ignored
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

Output

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

Usage

This function block can be used to:

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

Limitations

- Both high speed inputs cannot be used at the same time.

Related Functions

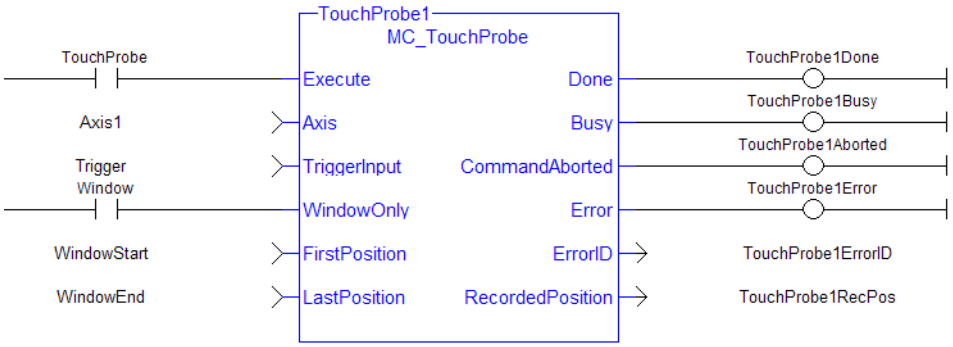
MC_AbortTrigger

Example

Structured Text

```
(* MC_TouchProbe ST example *)
TriggerInputRef.InputID := 1; //configure InputID
TriggerInputRef.Direction := 1; //configure Direction
TriggerInputRef.TrigID := 0; //configure TrigID
Inst_MC_TouchProbe( ArmProbe, Axis1, TriggerInputRef, FALSE,0.0, 0.0 );
//Inst_MC_TouchProbe is an instance of MC_TouchProbe function block
ProbeIsDone := Inst_MC_TouchProbe.Done; //store Done output into a user
defined variable
ProbeValue := Inst_MC_TouchProbe.RecordedPosition; //store Recor-
dedPosition output into a user defined variable
```

Ladder Diagram



2.2.3 Information Functions

This set of functions provides feedback and allows you to writer parameters.

2.2.3.1 MC_ReadActPos

Description

The MC_ReadActPos function block reads the actual position of the axis.

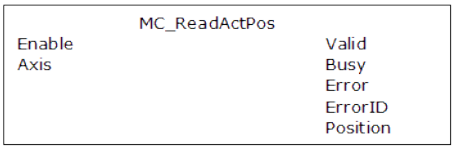


Figure 1-64: MC_ReadActPos

Arguments

Input

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

Output

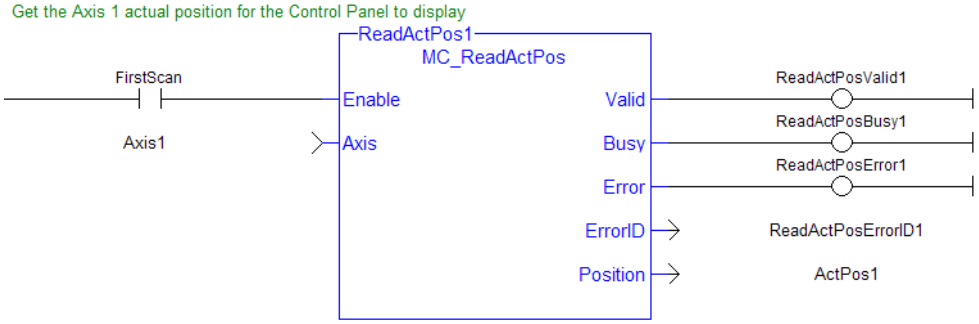
Valid	**Description**	Indicates the value at the Position output is available
	Data type	BOOL
Busy	**Description**	Indicates this function block is executing
	Data type	BOOL
Error	**Description**	Indicates an invalid input
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is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

Example

Structured Text

```
(* MC_ReadActPos ST example *)
Inst_MC_ReadActPos( TRUE, Axis1 );
//Inst_MC_ReadActPos is an instance of MC_ReadActPos function block

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

Ladder Diagram



2.2.3.2 MC_ReadActVel

Description

The MC_ReadActVel function block reads the actual velocity of the axis.

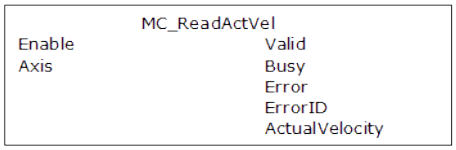


Figure 1-65: MC_ReadActVel

Arguments

Input

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

Output

Valid	**Description**	Indicates the value at the ActualVelocity output is available
	Data type	BOOL
Busy	**Description**	Indicates this function block is executing
	Data type	BOOL
Error	**Description**	Indicates an invalid input
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE.
	Data type	INT

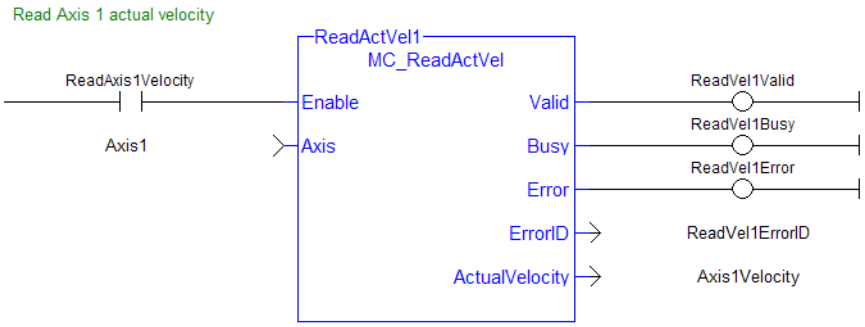
ActualVelocity	**Description**	Actual velocity of the axis. Please note that oscillations may be seen due to this being an instant velocity, not an average velocity.
	Unit	User unit/sec
	Data type	LREAL

Example

Structured Text

```
* MC_ReadActVel ST example *);
Inst_MC_ReadActVel( TRUE, Axis1 ); //Inst_MC_ReadActVel is an instance of
MC_ReadActVel function block

ActualVel := Inst_MC_ReadActVel.ActualVelocity; // store ActualVelocity
output into a user defined variable
```

Ladder Diagram



2.2.3.3 MC_ReadAxisErr

Description

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

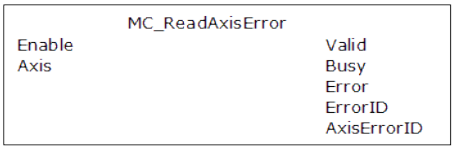


Figure 1-66: MC_ReadAxisErr

Arguments

Input

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

	Range	[1,256]
	Unit	n/a
	Default	—

Output

Valid	**Description**	Indicates the AxisErrorID output is valid
	Data type	BOOL
Busy	**Description**	Indicates this function block is executing
	Data type	BOOL
Error	**Description**	Indicates an invalid input
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE
	Data type	INT
AxisErrorID	**Description**	Indicates the error status of the axis. Each bit indicates a specific error. Both emergency-stop (E-stop) and controlled-stop (C-stop) errors are indicated. The table below defines the bits of this output.
	Data type	INT

Hexadecimal	Decimal	Description
0000H	0	No Error
0001H	1	User-set E-stop via MC_EStop, E-stop
0002H	2	Loss of Feedback, E-stop
0004H	4	Drive Fault, E-stop
0008H	8	Drive Communication Failure, E-stop
0400H	1024	Synchronization Error, C-stop

NOTE

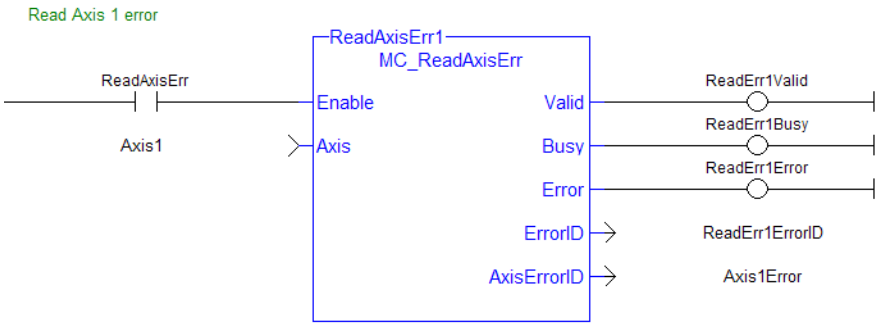
Multiple errors can be active at the same time. For example, if a User-set E-stop and an Excess Position Error E-stop are both active, the value would be 00000011H (17 decimal).

Example

Structured Text

```
(* MC_ReadAxisErr ST example *)
Inst_MC_ReadAxisErr( TRUE, Axis1 );
//Inst_MC_ReadAxisErr is an instance of MC_ReadAxisErr function block
AxisErrorBits := Inst_MC_ReadAxisErr.AxisErrorID; //AxisErrorID contains
the error bits
```

Ladder Diagram



2.2.3.4 MC_ReadBoolPar (Function Block)

Description

MC_ReadBoolPar returns the value of the specified Boolean axis parameter.

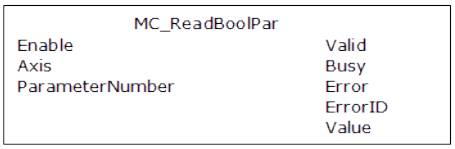


Figure 1-67: MC_ReadBoolPar

Arguments

Input

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

Output

Valid	**Description**	Indicates the Value output is valid
	Data type	BOOL
Busy	**Description**	Indicates this function block is executing
	Data type	BOOL
Error	**Description**	Indicates an invalid input

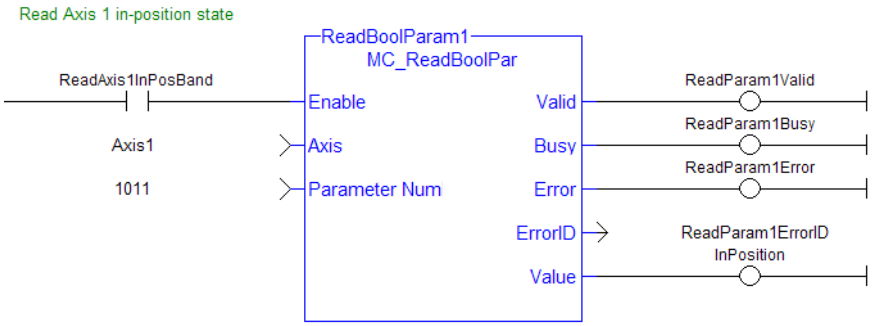
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE
	Data type	INT
Value	**Description**	State of the Boolean parameter
	Data type	BOOL

Example

Structured Text

```
(* MC_ReadBoolPar ST example *)
Inst_MC_ReadBoolPar( EnableRead, Axis1, 3 );
//Inst_MC_ReadBoolPar is an instance of MC_ReadBoolPar function block
BoolParm := Inst_MC_ReadBoolPar.Value; //store the Value output into a
user defined variable
```

Ladder Diagram



2.2.3.5 MC_ReadParam (Function Block)

Description

MC_ReadParam returns the value of the specified axis parameter.

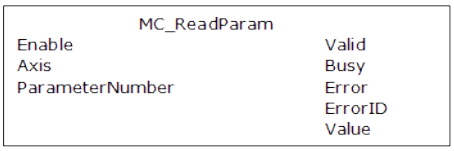


Figure 1-68: MC_ReadParam

Arguments

Input

Enable	**Description**	Requests to read the axis parameter
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	**Description**	Name of a declared instance of the AXIS_REF library function.)
	Data type	AXIS_REF

	Range	[1,256]
	Unit	n/a
	Default	—
ParameterNumber	**Description**	Parameter number, see table in Axis Parameters
	Data type	INT
	Range	—
	Unit	n/a
	Default	—

Output

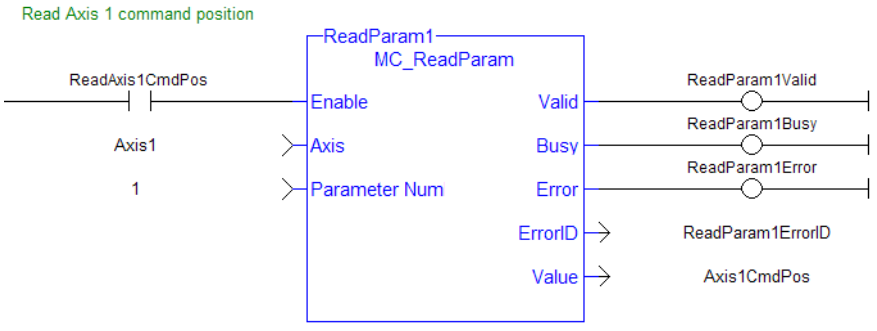
Valid	**Description**	Indicates the Value output is valid
	Data type	BOOL
Busy	**Description**	Indicates this function block is executing
	Data type	BOOL
Error	**Description**	Indicates an invalid input
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE
	Data type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

Example

Structured Text

```
(* MC_ReadParam ST example *)
ParameterNumber := 3;          //configure the parameter to read
Inst_MC_ReadParam( EnableRead, Axis1, ParameterNumber );
//Inst_MC_ReadParam is an instance of MC_ReadParam function block
ParmVal := Inst_MC_ReadParam.Value; //store the Value output into a user
defined variable
```

Ladder Diagram



2.2.3.6 MC_ReadStatus

Description

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

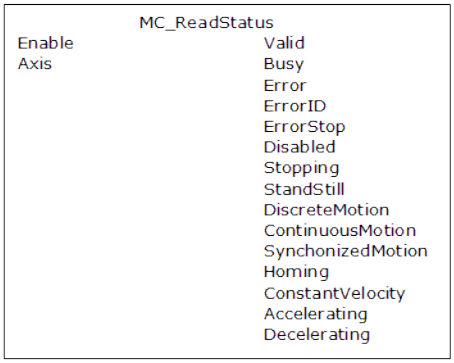


Figure 1-69: MC_ReadStatus

Arguments

Input

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

Output

Valid	**Description**	Indicates the outputs are valid
	Data type	BOOL
Busy	**Description**	Indicates this function block is executing
	Data type	BOOL
Error	**Description**	Indicates an invalid input
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE
	Data type	INT
ErrorStop	**Description**	Indicates Error Stop state – E-stop or C-stop
	Data type	BOOL
Disabled	**Description**	Indicates Disabled state – open loop and drive is disabled
	Data type	BOOL
Stopping	**Description**	Indicates Stopping state – MC_Stop command
	Data type	BOOL
StandStill	**Description**	Indicates Stand Still state – no move, closed loop, drive enabled
	Data type	BOOL

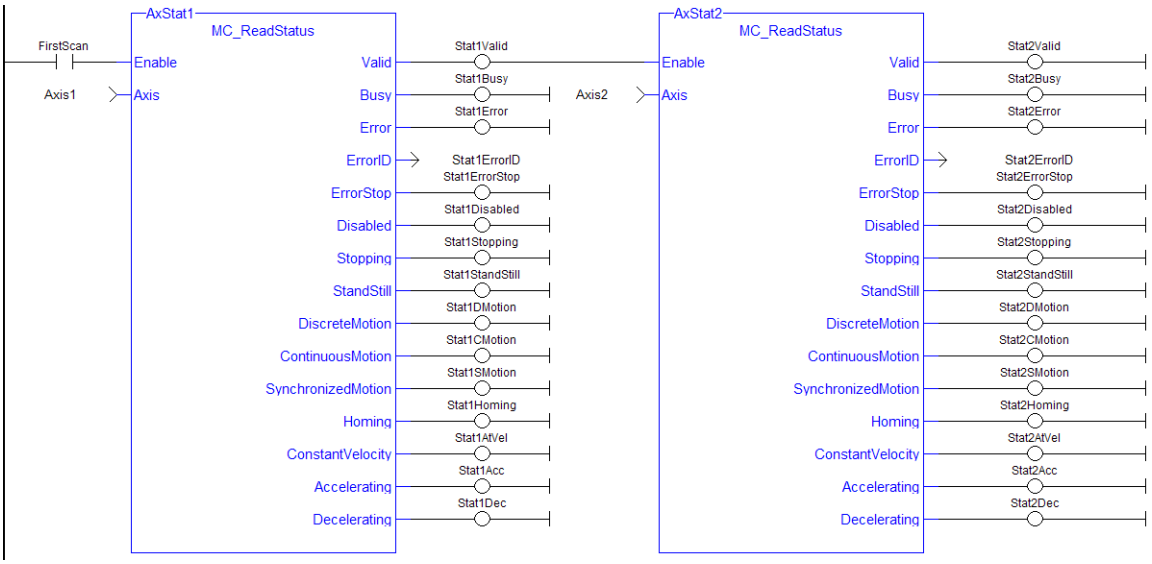
DiscreteMotion	**Description**	Indicates Discrete Motion state – programmed endpoint move is active
	Data type	BOOL
ContinuousMotion	**Description**	Indicates Continuous Motion state – unending, single-axis move is active
	Data type	BOOL
SynchronizedMotion	**Description**	Indicates Synchronized Motion state – slave move is active
	Data type	BOOL
Homing	**Description**	Indicates Homing state – a homing cycle is currently executing
	Data type	BOOL
ConstantVelocity	**Description**	Indicates the axis is moving at a constant velocity
	Data type	BOOL
Accelerating	**Description**	Indicates the axis is accelerating
	Data type	BOOL
Decelerating	**Description**	Indicates the axis is decelerating
	Data type	BOOL

Example

Structured Text

```
(* MC_ReadStatus ST example *)
Inst_MC_ReadStatus( EnableRead, Axis1 );
//Inst_MC_ReadStatus is an instance of MC_ReadStatus function block
AxisStopping := Inst_MC_ReadStatus.Stopping; // store Stopping output to
a user defined variable
AxisAccelerating := Inst_MC_ReadStatus.Accelerating; // store Accel-
erating output to a user defined variable
```

Ladder Diagram



2.2.3.7 MC_WriteBoolPar (Function Block)

Description

MC_WriteBoolPar writes the specified axis Boolean parameter.

Arguments

Input

Execute	**Description**	Requests to write a Boolean axis parameter
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	**Description**	Name of a declared instance of the AXIS_REF library function.
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
ParameterNumber	**Description**	Parameter number, see table in Axis Parameters
	Data type	INT
	Range	—
	Unit	n/a
	Default	—
Value	**Description**	State to write
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—

Output

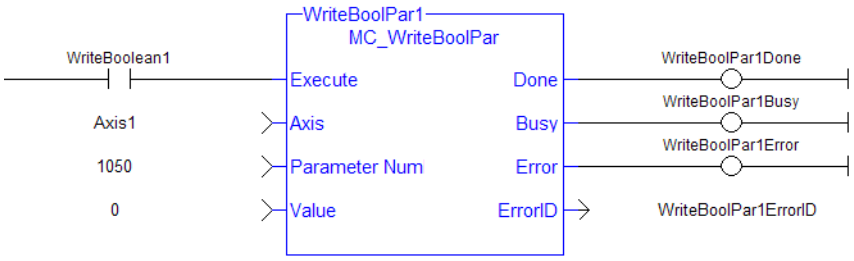
Done	**Description**	Indicates the Boolean parameter has been written
	Data type	BOOL
Busy	**Description**	Indicates this function block is executing
	Data type	BOOL
Error	**Description**	Indicates an invalid input
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to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

Example

Structured Text

```
(* MC_WriteBoolPar ST example *)
WriteBool := TRUE;          //value to write to the boolean parameter #1
Inst_MC_WriteBoolPar( WriteReq, Axis1, 1, WriteBool ); //Inst_MC_
WriteBoolPar is an instance of MC_WriteBoolPar
```

Ladder Diagram



NOTE

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

2.2.3.8 MC_WriteParam (Function Block)

Description

MC_WriteParam writes the specified axis parameter.

Arguments

Input

Execute	**Description**	Requests to write the axis parameter
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	**Description**	Name of a declared instance of the AXIS_REF library function.
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
ParameterNumber	**Description**	Parameter number, see table in Axis Parameters
	Data type	INT
	Range	—
	Unit	n/a
	Default	—
Value	**Description**	Value to write
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—

Output

Done	**Description**	Indicates the parameter has been written
	Data type	BOOL
Busy	**Description**	Indicates this function block is executing

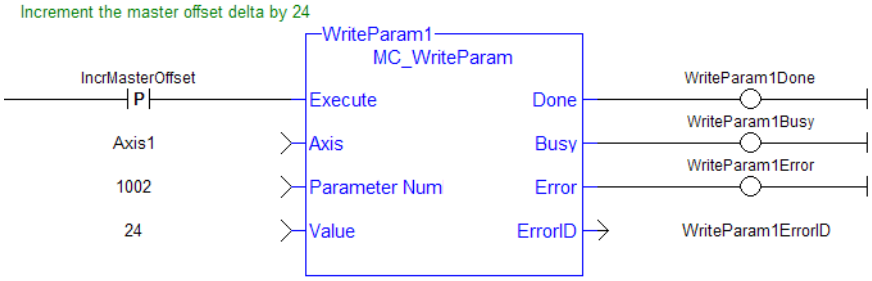
	Data type	BOOL
Error	**Description**	Indicates an invalid input
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE
	Data type	INT

Example

Structured Text

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

Ladder Diagram



2.2.4 PLCOpenMotion Functions

This set of functions provides control over an axis.

2.2.4.1 MC_Halt (Function Block)

Description

This function block decelerates an axis to zero velocity. It is a queued single-axis move. The move is complete when the axis reaches zero velocity. It is typically used with Abort at the BufferMode input to terminate a move. To execute a stop that cannot be aborted, see MC_Stop.

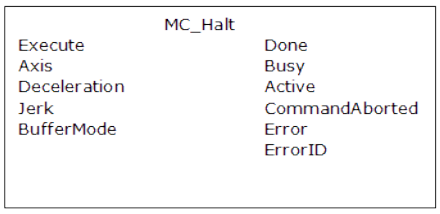


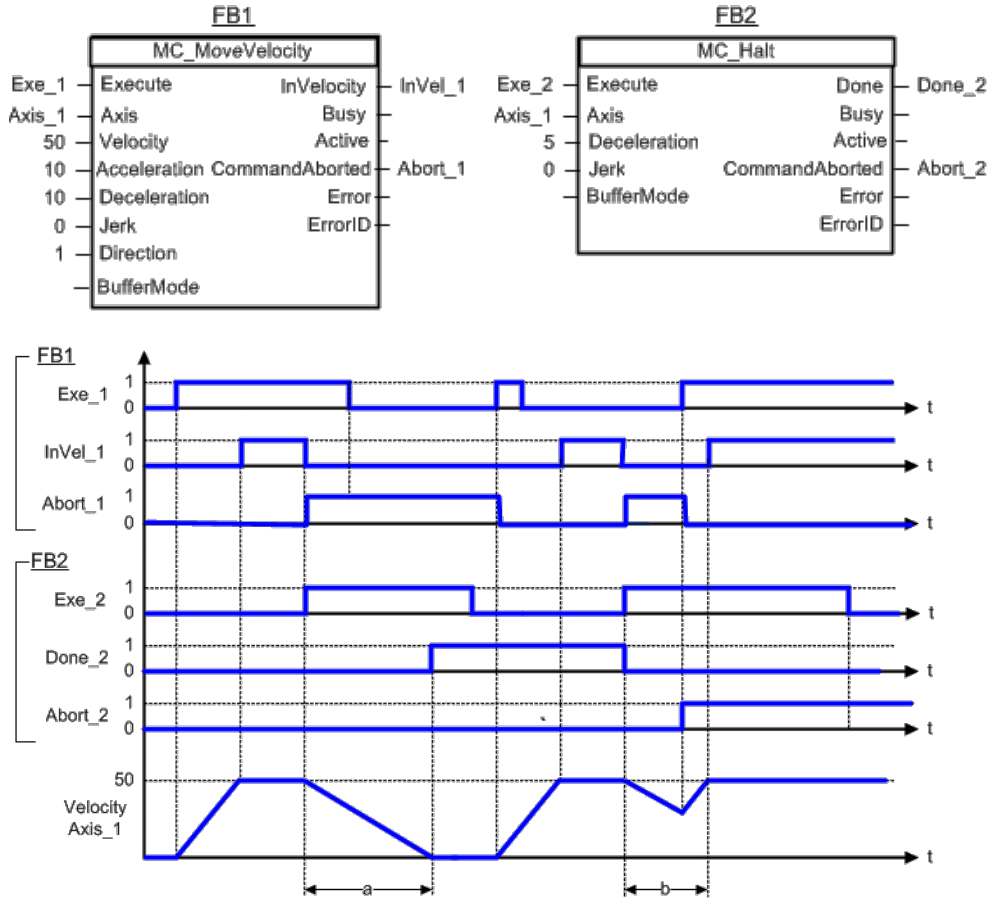
Figure 1-70: MC_Halt

Time Diagram

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

- A rotating axis is ramped down with FB2 MC_Halt
- Another motion command overrides the MC_Halt command

MC_Halt allows this, in contrast to MC_Stop. The axis can accelerate again without reaching standstill.



Arguments

Input

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

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

Output

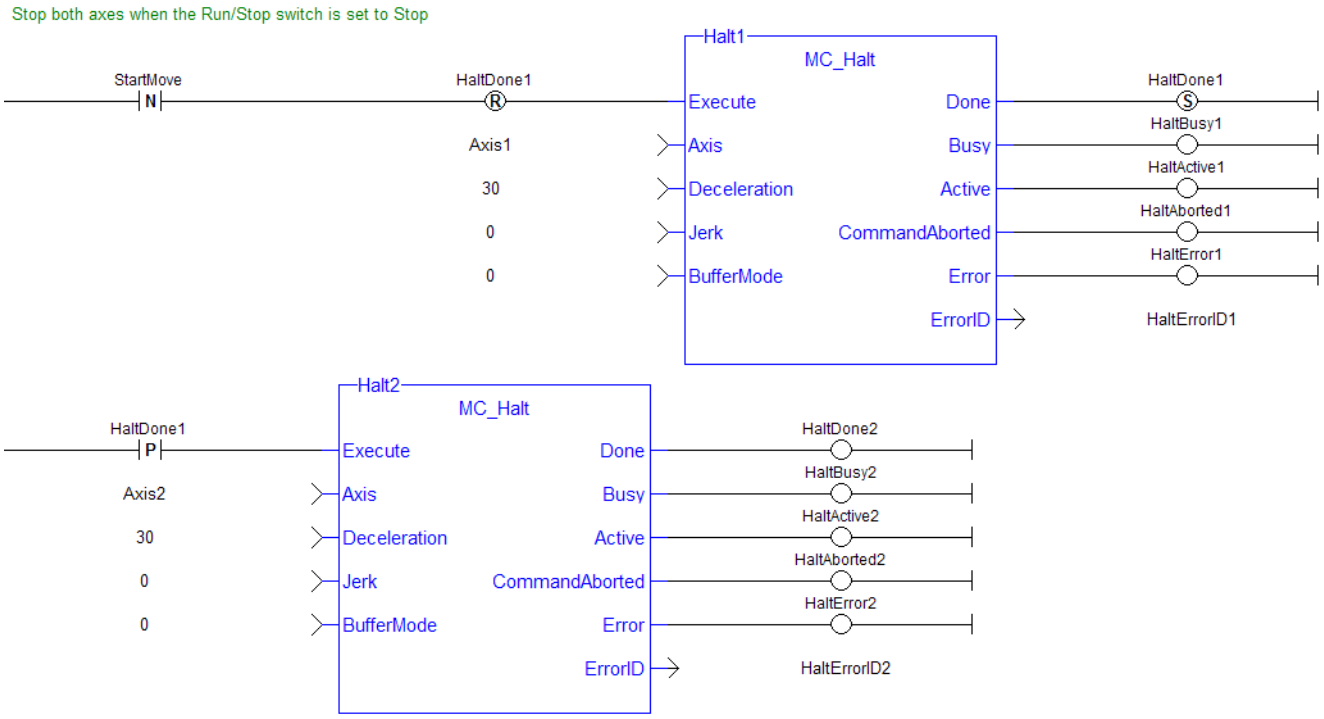
Done	Description	Indicates the move completed successfully. The Command Position has reached the endpoint.
	Data type	BOOL
Busy	Description	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
Active	Description	Indicates this move is the active move
	Data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

Example

Structured Text

```
(* MC_Halt ST example *)
Inst_MC_Halt( HaltReq, Axis1,100.0, 100.0, 0 );
//Inst_MC_Halt is an instance of MC_halt function block
HaltComplete := Inst_MC_Halt.Done; //store Done output into user
defined variable
```

Ladder Diagram



2.2.4.2 MC_MoveAbsolute

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.

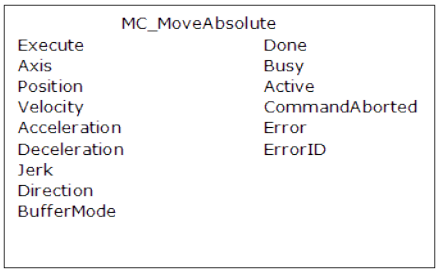
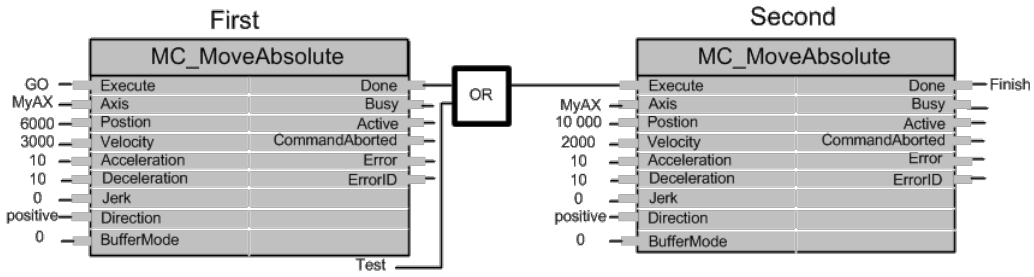


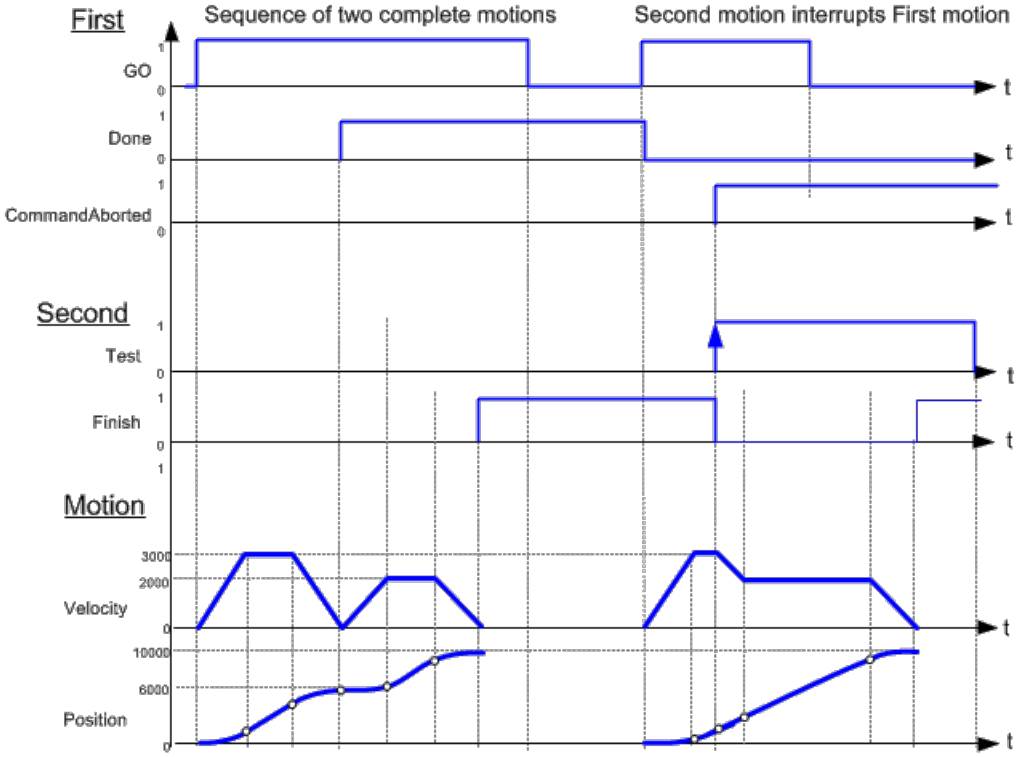
Figure 1-71: MC_MoveAbsolute

Time Diagram

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

- The left part of timing diagram illustrates the case if the Second Function Block is called **after** the First one. If First reaches the commanded position of 6000 (and the velocity is 0) then the output Done causes the Second FB to move to the position 10000
- The right part of the timing diagram illustrates the case if the Second move Function Block starts the execution **while** the First FB is still executing. In this case the First motion is interrupted and aborted by the Test signal during the constant velocity of the First FB. The Second FB moves directly to the position 10000 although the position of 6000 is not yet reached





Arguments

Input

Execute	**Description**	Requests to queue the move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	**Description**	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—

Position	**Description**	Endpoint position. If Rollover Position is nonzero, this value must be in the range 0 <= Position < Rollover Position When not in Rollover mode, the input accepts a 64-bit floating point value. When converted to feedback units, the range is $[-2^{51},2^{51}-1]$ feedback units.
	Data type	LREAL
	Range	[see Description]
	Unit	User unit
	Default	—
Velocity	**Description**	Velocity setpoint
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	**Description**	Trapezoidal: Acceleration rate S-curve: Maximum acceleration If Acceleration is not valid, ErrorID is set to 21 Selection of Acceleration and Jerk Parameters for Function Blocks
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Deceleration	**Description**	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Jerk	**Description**	Trapezoidal: 0 S-curve: Constant jerk If Jerk is not valid, ErrorID is set to 21 Selection of Acceleration and Jerk Parameters for Function Blocks
	Data type	LREAL
	Range	—
	Unit	User unit/sec^3
	Default	—

Direction

Description When Rollover Position is zero, a value of 0 must be specified. When Rollover Position is nonzero, a value of 1, 2, 3, or 4 must be specified.

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

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

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

Data type SINT

Range [0,4]

Unit n/a

Default —

BufferMode

Description
0 = abort
1 = buffer
2 = blend to active
3 = blend to next
4 = blend to low velocity
5 = blend to high velocity

Data type SINT

Range [0,5]

Unit n/a

Default —

Output

Done

Description Indicates the move completed successfully.
The Command Position has reached the endpoint.

Data type BOOL

Busy

Description High from the moment the Execute input is one-shot to the time the move is ended

Data type BOOL

Active

Description Indicates this move is the active move

Data type BOOL

CommandAborted

Description Indicates the move was aborted

Data type BOOL

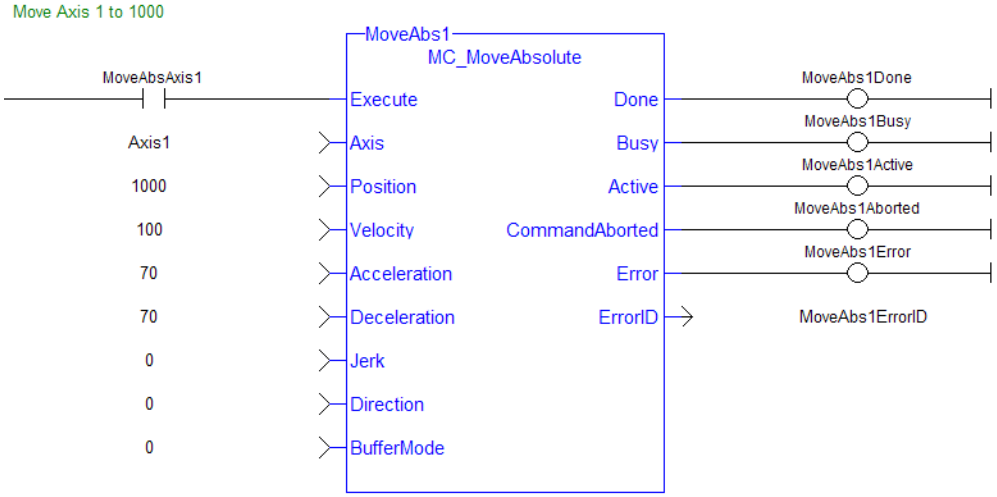
Error	**Description**	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE
	Data type	INT

Example

Structured Text

```
(* MC_MoveAbsolute ST example *)
Inst_MC_MoveAbsolute( MovAbsReq, Axis1, 1234.567, 100.0, 100.0, 100.0, 0,
0, 0 );  //instance of MC_MoveAbsolute
MovAbsDone := Inst_MC_MoveAbsolute.Done;  //store done output into user
defined variable
MovAbsBusy := Inst_MC_MoveAbsolute.Busy;
MovAbsActive := Inst_MC_MoveAbsolute.Active;
MovAbsAborted := Inst_MC_MoveAbsolute.CommandAborted;
MovAbsError := Inst_MC_MoveAbsolute.Error;
MovAbsErrID := Inst_MC_MoveAbsolute.ErrorID;
```

Ladder Diagram



2.2.4.3 MC_MoveAdditive

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.

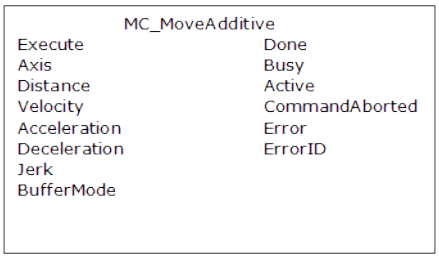
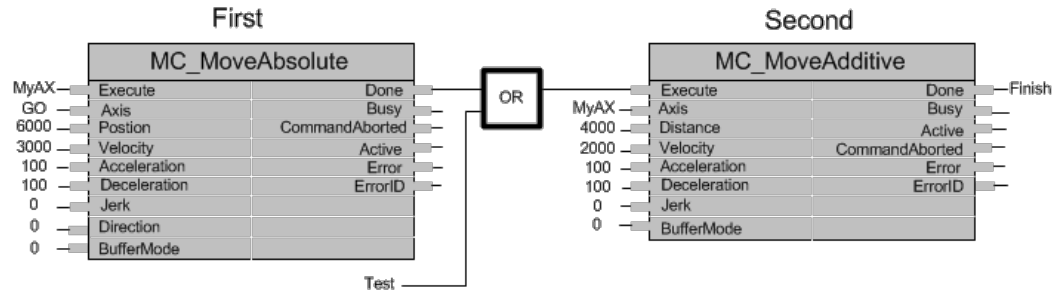


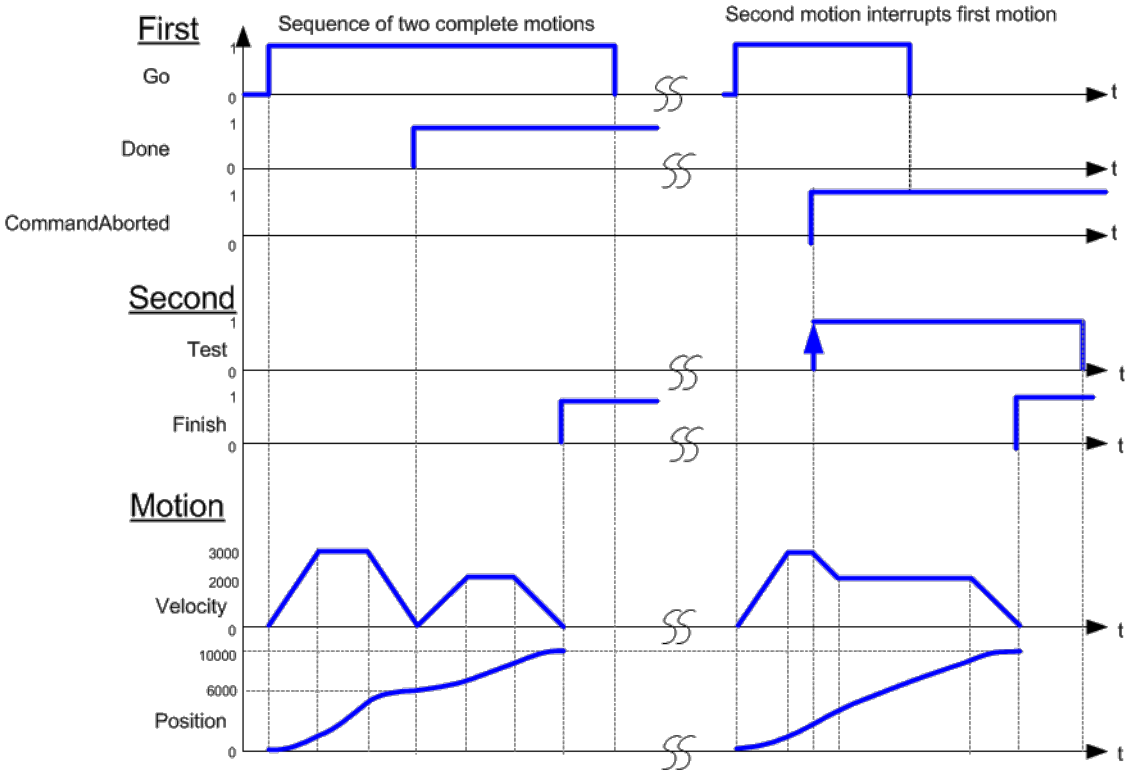
Figure 1-72: MC_MoveAdditive

Time Diagram

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

- The left part of timing diagram illustrates the case if the Second Function Block is called **after** the First one. If First reaches the commanded distance 6000 (and the velocity is 0) then the output **Done** causes the Second FB to move to the distance 10000
- The right part of the timing diagram illustrates the case if the Second move Function Blocks starts the execution **while** the First FB is still executing. In this case the First motion is interrupted and aborted by the Test signal during the constant velocity of the First FB. The Second FB **adds on the previous commanded position** of 6000 the distance 4000 and moves the axis to the resulting position of 10000





Arguments

Input

Execute	**Description**	Requests to queue the move
	Data type	BOOL
	Range	0, 1

	Unit	n/a
	Default	—
Axis	**Description**	Name of a declared instance of the AXIS_REF library function.)
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Distance	**Description**	Distance to add to the endpoint of the previous move
	Data type	REAL
	Range	—
	Unit	User unit
	Default	—
Velocity	**Description**	Velocity setpoint
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	**Description**	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Deceleration	**Description**	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Jerk	**Description**	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec^3
	Default	—
BufferMode	**Description**	0 = abort 1 = buffer 2 = blend to active 3 = blend to next 4 = blend to low velocity 5 = blend to high velocity
	Data type	SINT
	Range	[0,5]
	Unit	n/a
	Default	—

Output

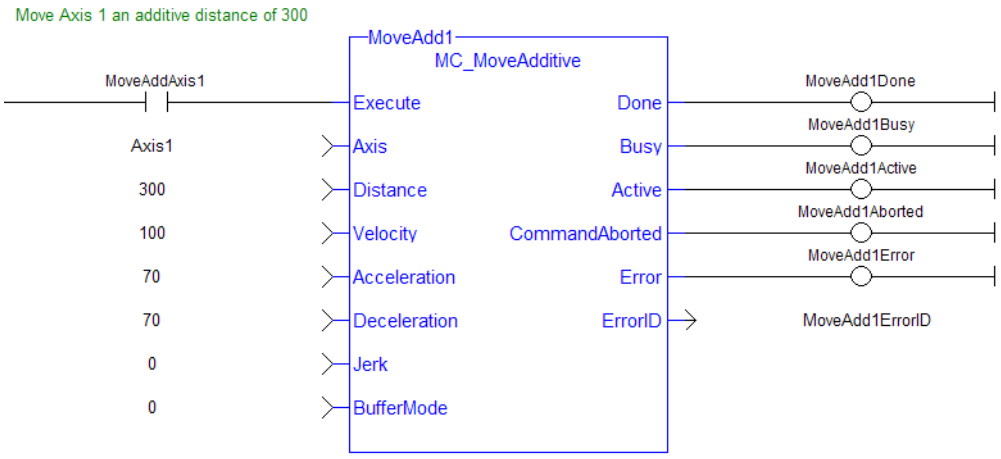
Done	**Description**	Indicates the move completed successfully. The Command Position has reached the endpoint.
	Data type	BOOL
Busy	**Description**	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
Active	**Description**	Indicates this move is the active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

Example

Structured Text

```
(* MC_MoveAdditive ST example *)
Inst_MC_MoveAdditive( 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
```

Ladder Diagram



2.2.4.4 MC_MoveRelative

Description

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

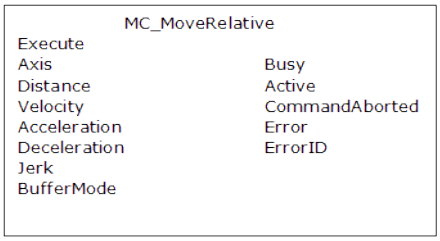
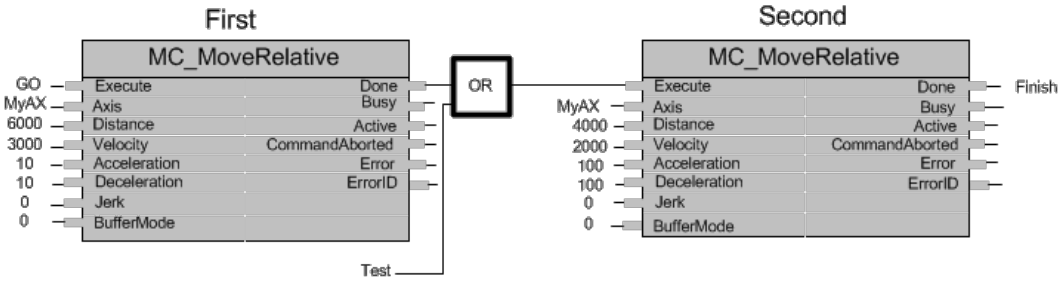


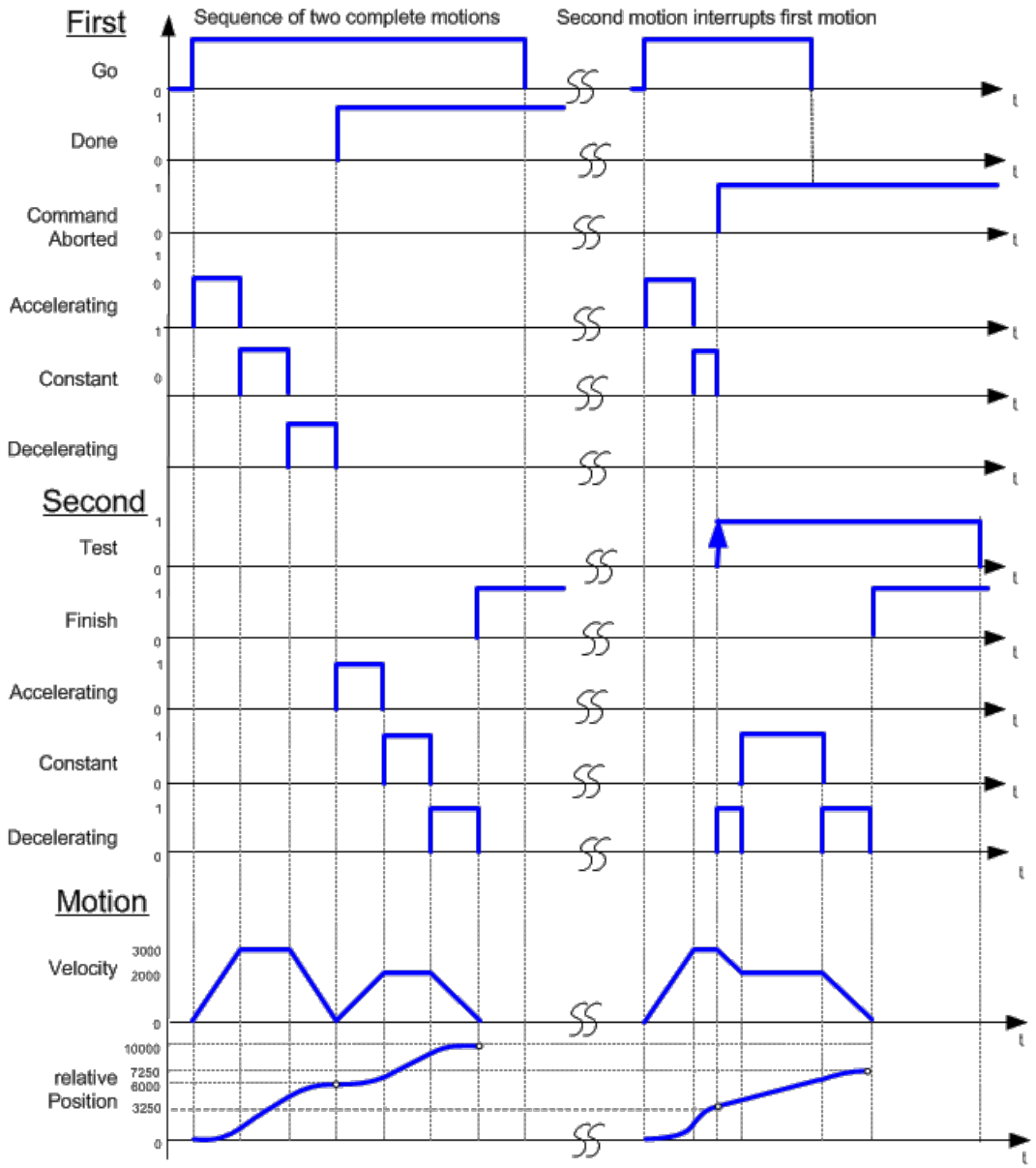
Figure 1-73: MC_MoveRelative

Time Diagram

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

- The left part of timing diagram illustrates the case if the Second Function Block is called **after** the First one. If First reaches the commanded distance 6000 (and the velocity is 0) then the output **Done** causes the Second FB to move to the distance 10000
- The right part of the timing diagram illustrates the case if the Second move Function Blocks starts the execution **while** the First FB is still executing. In this case the First motion is interrupted and aborted by the Test signal during the constant velocity of the First FB. The Second FB **adds on the actual position** of 3250 the distance 4000 and moves the axis to the resulting position of 7250





Arguments

Input

Execute	**Description**	Requests to queue the move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	**Description**	Name of a declared instance of the AXIS_REF library function.
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—

Distance	**Description**	Distance
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
Velocity	**Description**	Velocity setpoint
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	**Description**	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Deceleration	**Description**	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Jerk	**Description**	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec^3
	Default	—
BufferMode	**Description**	0 = abort 1 = buffer 2 = blend to active 3 = blend to next 4 = blend to low velocity 5 = blend to high velocity
	Data type	SINT
	Range	[0,5]
	Unit	n/a
	Default	—

Output

Done	**Description**	Indicates the move completed successfully. The Command Position has reached the endpoint.
	Data type	BOOL
Busy	**Description**	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL

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

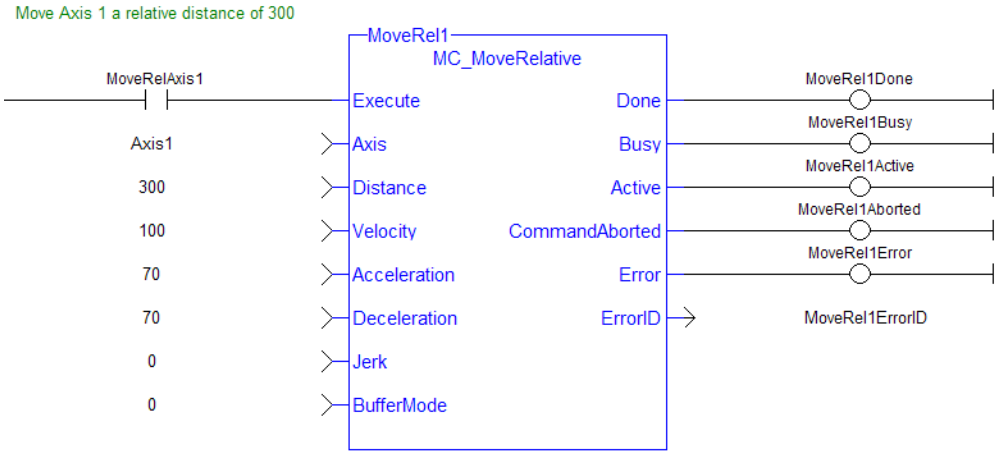
Example

Structured Text

```
(* MC_MoveRelative ST example *)
Inst_MC_MoveRelative( MovRelReq, Axis1, 10.0, 200.0,150.0, 150.0, 0,0 );
MovRelDone := Inst_MC_MoveRelative.Done;  //store Done output into user
defined variable
```

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

Ladder Diagram



2.2.4.5 MC_MoveSuperimp

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. 346) 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.

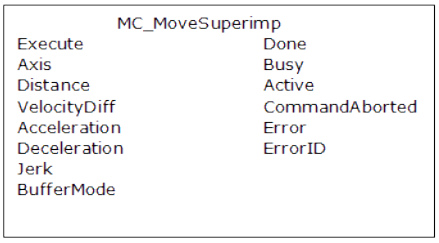
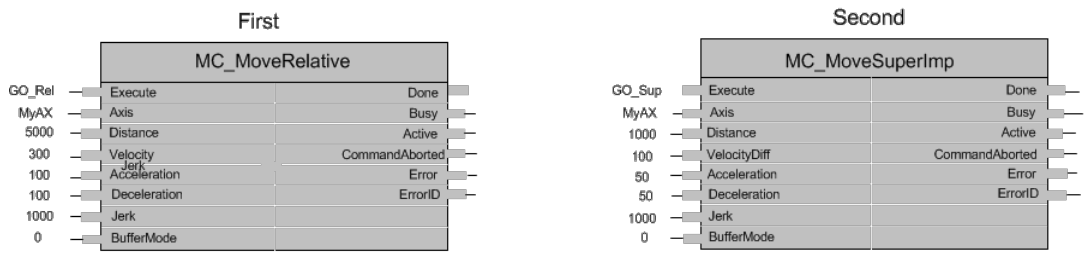
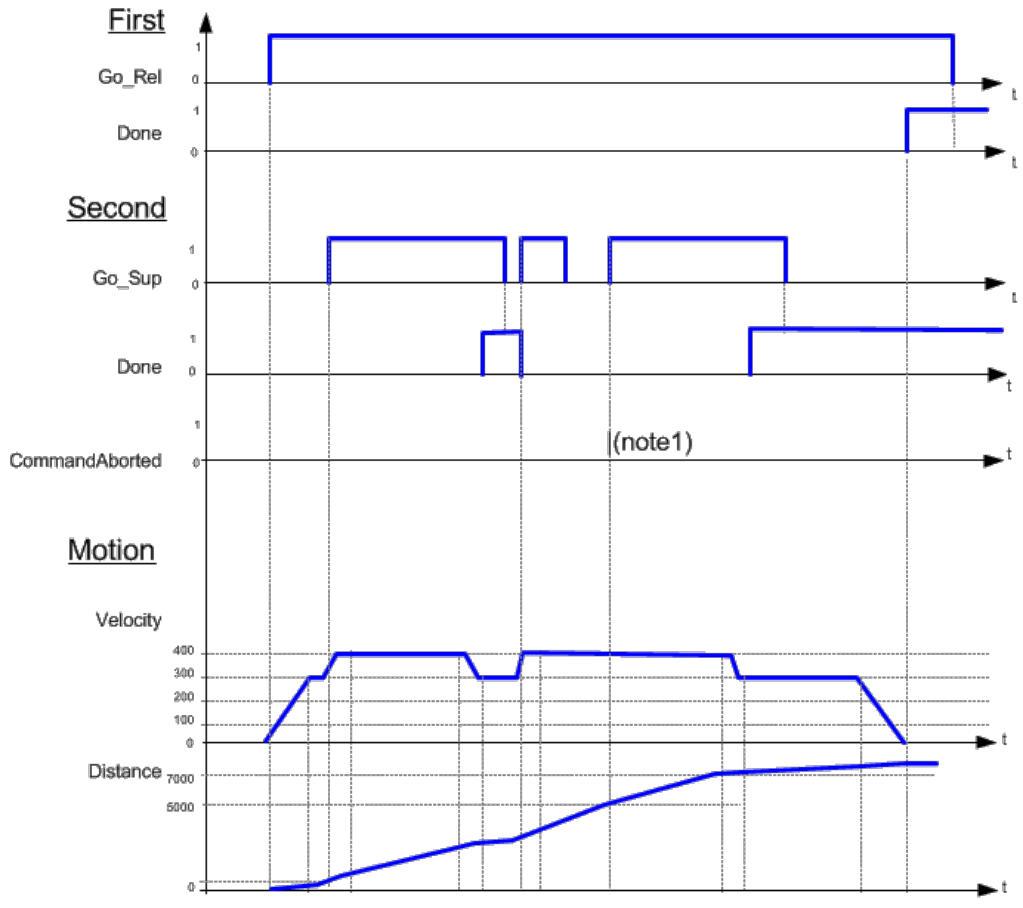


Figure 1-74: MC_MoveSuperimp

Time Diagram





NOTE

1. The CommandAborted is not visible here, because the new command works on the same instance
2. The end position is between 7000 and 8000, depending on the timing of the aborting of the second command set for the MC_MoveSuperimposed

Arguments

Input

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

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

Output

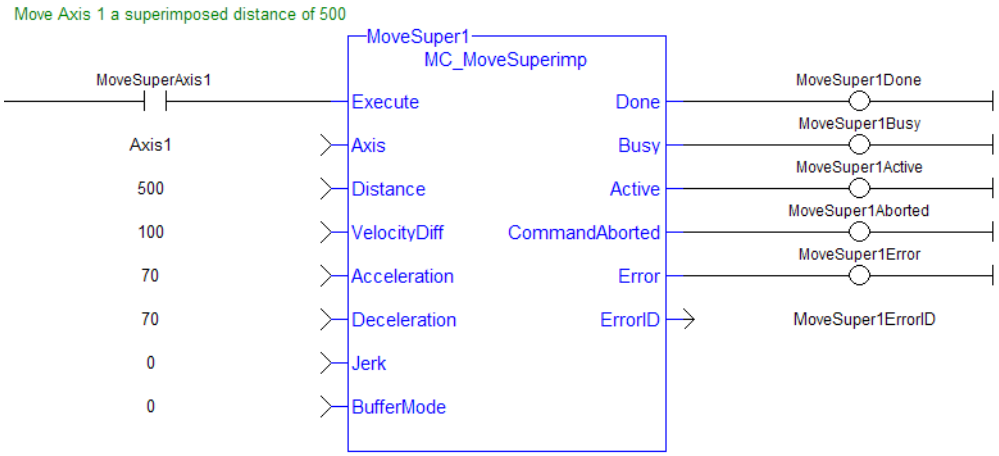
Done	**Description**	Indicates the move completed successfully. The Command Position has reached the endpoint.
	Data type	BOOL
Busy	**Description**	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
Active	**Description**	Indicates this move is the active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

Example

Structured Text

```
(* MC_MoveSuperimp ST example *)
Inst_MC_MoveSuperimp( MovSupReq, Axis1, 123.555, 10.0, 100.0, 100.0, 0, 0
);
MovSupDone := Inst_MC_MoveSuperimp.Done;  //store Done output into user
defined variable
```

Ladder Diagram



2.2.4.6 MC_MoveVelocity

Description

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

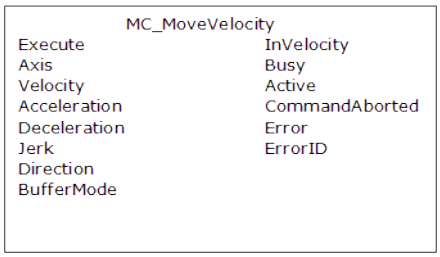


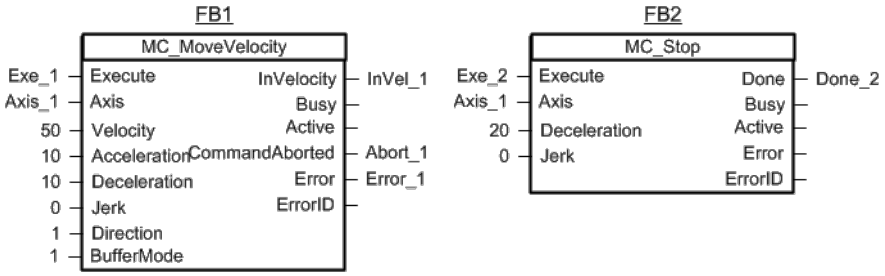
Figure 1-75: MC_MoveVelocity

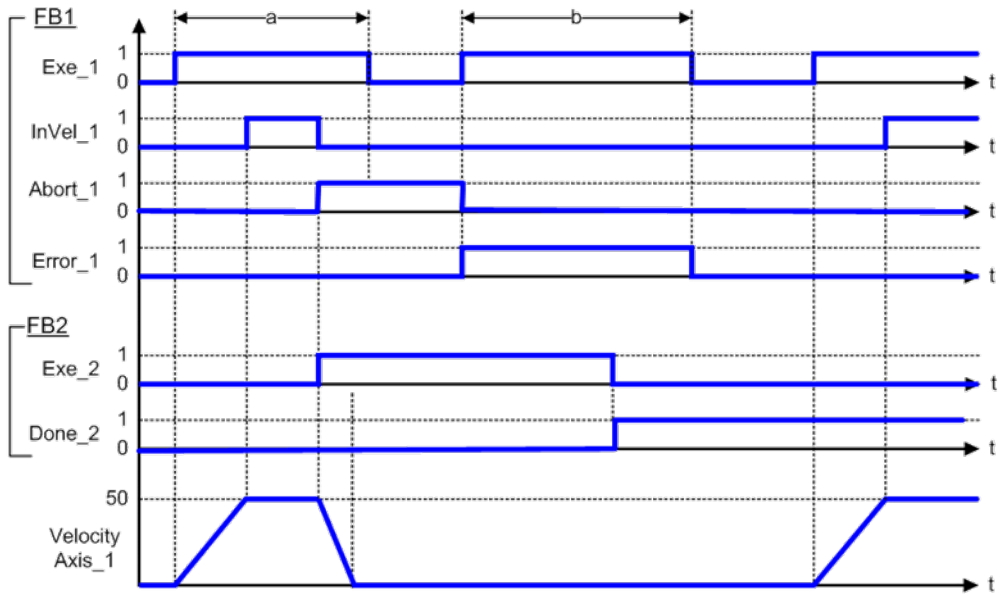
Time Diagram

The example below shows the behavior of the combination of a MC_Stop FB with a MC_MoveVelocity FB.

- A rotating axis is ramped down with FB2 MC_Stop
- The axis rejects motion commands as long as MC_Stop parameter “Execute” = TRUE

FB1 MC_MoveVelocity reports an error indicating the busy MC_Stop command.





Arguments

Input

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

	Default	—
Jerk	**Description**	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec^3
	Default	—
Direction	**Description**	0 = positive direction 1 = negative direction
	Data type	SINT
	Range	[0,1]
	Unit	n/a
	Default	—
BufferMode	**Description**	0 = abort 1 = buffer 2 = blend to active 3 = blend to next 4 = blend to low velocity 5 = blend to high velocity
	Data type	SINT
	Range	[0,5]
	Unit	n/a
	Default	—

Output

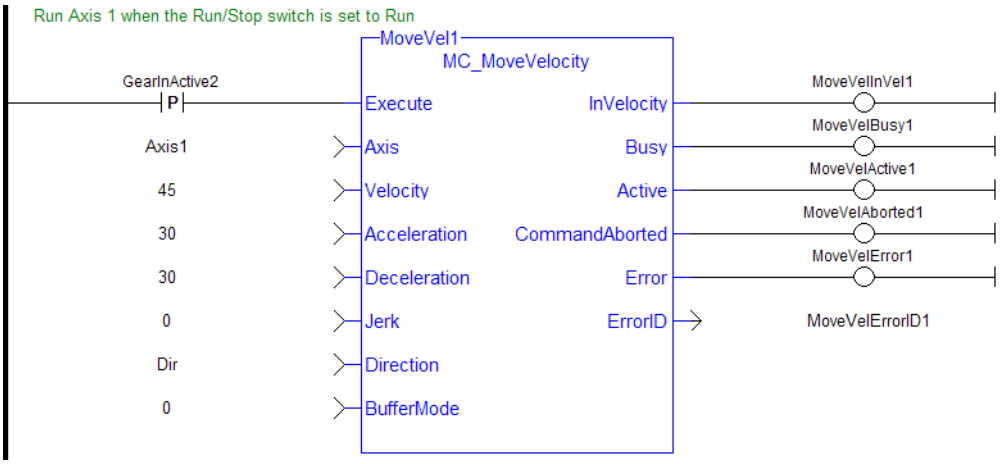
InVelocity	**Description**	Indicates the command velocity has reached the programmed velocity
	Data type	BOOL
Busy	**Description**	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
Active	**Description**	Indicates this move is the active move
	Data type	BOOL
CommandAborted	**Description**	Indicates the move was aborted
	Data type	BOOL
Error	**Description**	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE
	Data type	INT

Example

Structured Text

```
(* MC_MoveVelocity ST example *)
Inst_MC_MoveVelocity( MovVelReq , Axis1, 200.0, 100.0,100.0, 0, 0, 0 );
```

Ladder Diagram



2.2.4.7 MC_SetOverride

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. 339), "MC_MoveAdditive" (➜ p. 343), "MC_MoveRelative" (➜ p. 346), "MC_MoveSuperimp" (➜ p. 350), or "MC_MoveVelocity" (➜ p. 354) function block) to determine the command velocity:

```
command velocity = programmed velocity * VelFactor
```

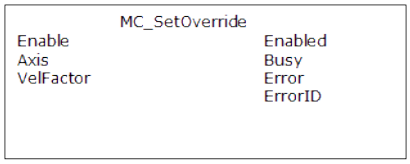


Figure 1-76: MC_SetOverride

Arguments

Input

Enable	**Description**	Request to write the override factors
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	**Description**	Name of a declared instance of the AXIS_REF library function.
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
VelFactor	**Description**	Velocity override factor
	Data type	REAL

	Range	[0.0, 2.0]
	Unit	n/a
	Default	—

Output

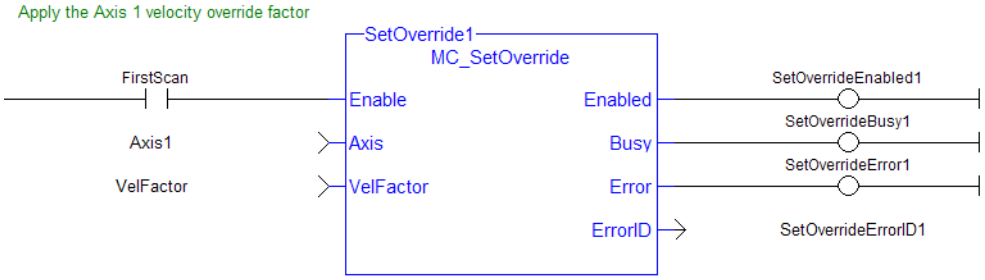
Enabled	**Description**	Indicates the override values have been written
	Data type	BOOL
Busy	**Description**	Indicates the function block is executing
	Data type	BOOL
Error	**Description**	Indicates an invalid input is specified
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE
	Data type	INT

Example

Structured Text

```
(* MC_SetOverride ST example *)
VelFactor := 1.25 ;   //set the velocity factor to 1.25 (125%)
Inst_MC_SetOverride( TRUE , Axis1, VelFactor ); // Inst_MC_Setoverride is
an instance of MC_SetOverride
```

Ladder Diagram



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

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. 369) and "MC_CamResumePos" (➜ p. 367) for information on positioning the slave axis prior to calling MC_CamIn.

Figure 1-77: MC_CamIn

Arguments

Input

Execute	**Description**	Requests to queue the CamIn move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Master	**Description**	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	1 - 256
	Unit	n/a
Slave	**Description**	AXIS_REF.AXIS_NUM is the slave axis number
	Data type	AXIS_REF
	Range	1-256
	Unit	n/a
MasterOffset	**Description**	Profile shift along the master axis. This input is not used if the StartMode input is set to 1 for Resume Mode.
	Data type	LREAL
	Range	—
	Unit	User unit
SlaveOffset	**Description**	Profile shift along the slave axis. This input is not used if the StartMode input is set to 1 for Resume Mode.
	Data type	LREAL
	Range	—
	Unit	User unit
MasterScaling	**Description**	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.

	Data type	LREAL
	Range	—
	Unit	User unit
SlaveScaling	**Description**	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.
	Data type	LREAL
	Range	—
	Unit	User unit
Startmode	**Description**	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.
	Data type	INT
	Range	[0,1]
	Unit	n/a
CamTableID	**Description**	ID number of the profile to be used with MC_CamIn
	Data type	INT
	Range	—
	Unit	n/a
BufferMode	**Description**	Buffer mode for CamIn block.
	Data type	SINT
	Range	0-5
	Unit	n/a

Output

InSync	**Description**	Indicates the slave axis is in sync with the profile
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Busy	**Description**	Indicates this function block is executing
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Active	**Description**	Indicates this move is the Active move
	Data type	BOOL

	Range	0, 1
	Unit	n/a
CommandAborted	**Description**	Indicates the move was aborted
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Error	**Description**	Indicates an invalid input, or the move was terminated due to an error
	Data type	BOOL
	Range	0, 1
	Unit	n/a
ErrorID	**Description**	Indicates the error if the Error output is high.
	Data type	INT
	Range	—
	Unit	n/a
EndOfProfile	**Description**	Indicates the end of profile has been reached. If the profile is periodic this output is set to ON for one ladder scan. If the profile is not periodic, the output remains ON while outside the range of the profile.
	Data type	BOOL
	Range	0, 1
	Unit	n/a

Usage

The slave axis immediately locks on to the Cam Table profile.

The **Master Offset** is used to shift the profile along the master axis.

The **Master Scaling** defines the range of the profile along the master axis.

The **Slave Offset** is used to shift the profile along the Slave axis.

The **Slave Scaling** defines the range of the profile along the slave axis.

If the profile is periodic, when the end of profile reached, the profile continues at the start of the profile. The EndOfProfile output is ON for 1 ladder scan.

If the profile is not periodic, when the end of profile is reached, the slave axis stops and remains at the end of the profile until the master axis returns to within the profile range as defined by MasterScaling. The EndOfProfile output remains ON anytime the master axis is outside of the profile range.

Adjustments computation is done as follows:

When cam is first started, offsets are adjusted if necessary

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

At run-time

- Master position for profile = master position - master offset
- Use master position for profile table to obtain slave profile position
- Slave commanded position = slave profile position + slave offset

Related Functions

"MC_CamResumePos" (➜ p. 367)

"MC_CamStartPos" (➜ p. 369)

MC_CamTblSelect

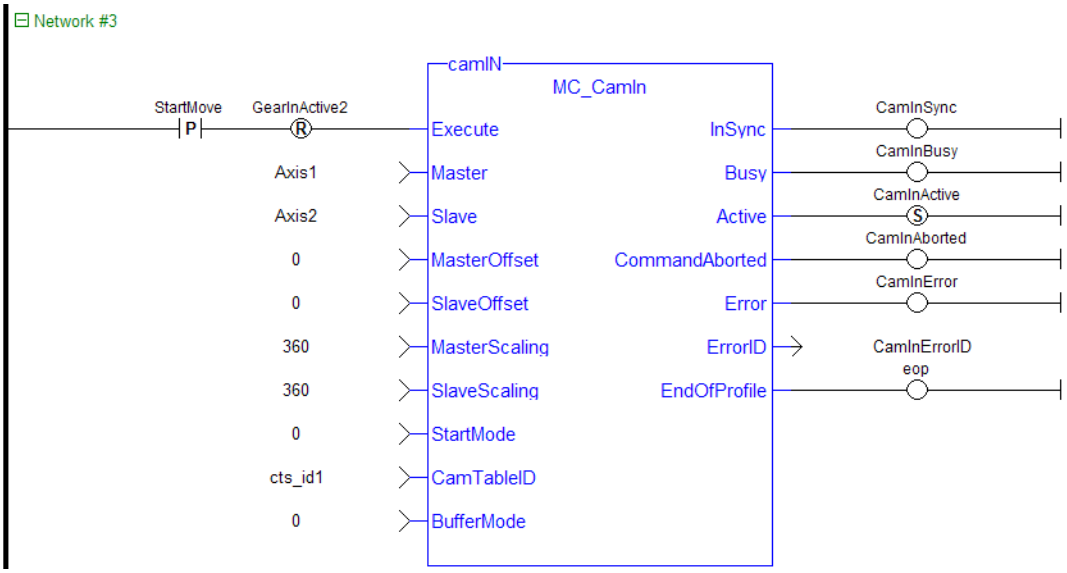
MC_CamOut

Examples

Structured Text

```
(* MC_CamIn ST example *)  //Inst_MC_CamIn is an instance of MC_CamIn
Inst_MC_CamIn( CamStartBool, Axis1, Axis2, 0.0, 0.0, 360.0, 360.0, 0,
CamTableID, 0 );
```

Ladder Diagram



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

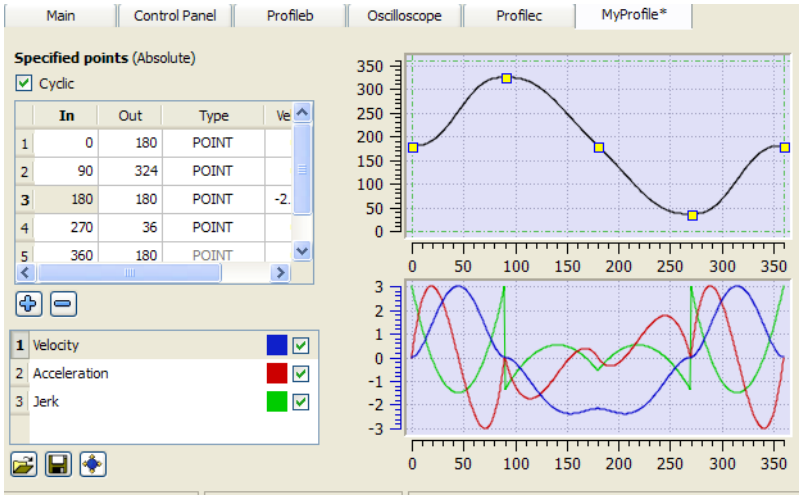


Figure 1-78: MC_CamIn examples

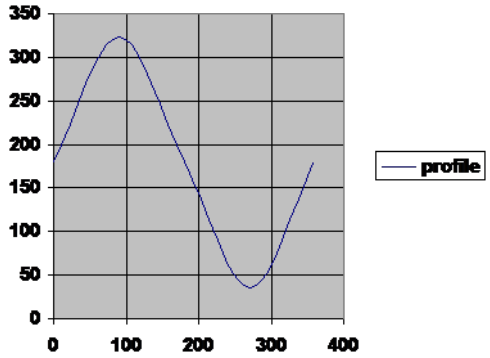
Example 1

Profile	MyProfile

Cycle	NO
MasterAbsolute	YES
SlaveAbsolute	YES
MasterOffset	0.0
SlaveOffset	0.0
MasterScaling	360.0
SlaveScaling	360.0
Initial Master position	0.0
Initial Slave position	180.0

After `MC_CamTblSelect` and `MC_CamIn` are programmed with the above parameters, the slave axis is locked on to the profile. Since both have zero offsets, the profile is not shifted in either axis. The initial condition of the master axis at position 0, yields a slave command position of 180.0. As the master axis moves positive, the slave position follows the profile. When the master position is at 90.0, the slave is commanded to 324.0 (see curve below where in = 90, out = 324). The slave follows the profile as the master axis moves until the master axis reaches a position of 360.0. At this time the slave is commanded to 180.0.

If the master were to continue to move past 360.0 the slave commanded position would remain at 180.0 since the Cyclic input is false. If the master moves negative and its position returns to less than 360.0, then the slave follows the profile again.



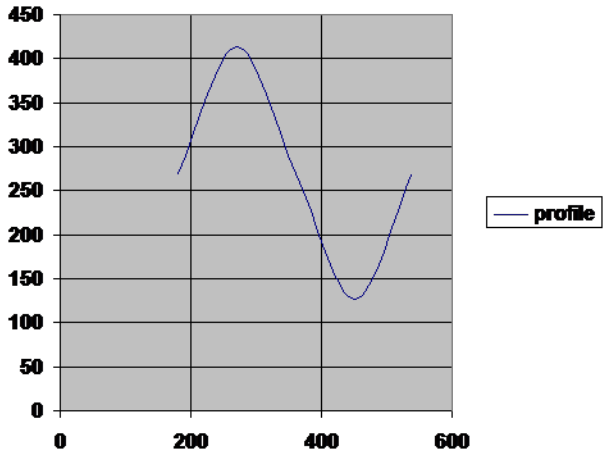
Example 2

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

After `MC_CamTblSelect` and `MC_CamIn` are programmed with the above parameters, the slave axis is locked on to the profile. Since the both axes have zero offsets, the profile is not shifted in either axis. Neither the *MasterAbsolute* nor *SlaveAbsolute* input is on, so the profile is relative to the axes initial positions. Specifically, the initial condition of the master axis at position 180 would represent a master profile position of 0

(180-180). This yields a slave command position of 270 (180 + 90). As the master axis moves positive, the slave position follows the profile. When the master position is at 270, the slave is commanded to 414.0 (324 + 90). The slave follows the profile as the master axis moves until the master axis reaches a position of 540. At this time the slave is commanded to 270.0 (180 + 90).

If the master continues to move past 540.0, the slave commanded position follows the profile from the beginning since the Cyclic input is TRUE. When the master reaches a position of 630, the slave is commanded to a position of 414.0 (324 + 90).



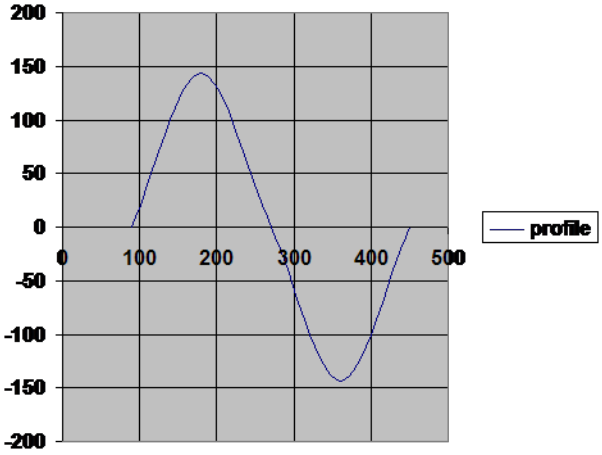
Example 3

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

After `MC_CamTblSelect` and `MC_CamIn` are programmed with the above parameters, the slave axis is locked on to the profile. Since the both axes have offsets, the profile is shifted along both axes. Specifically the master axis is shifted 90, and the slave axis is shifted -180. Initially the master axis position of 180 yields a master position for the profile calculation of 90 (master position 180 - Master offset 90), which yields a slave command position of 144 (slave profile command 324 + slave offset (-180)). As the master axis moves positive, the slave position follows the profile. When the master axis position is at 270, the master position for profile calculation is 180 (270 - 90). This yields a slave command position of 0 (180 + (-180)).

The slave follows the profile as the master axis moves until the master axis reaches a position of 450. The master axis position of 450 yields a master position for profile calculation of 360 (450 - 90). The slave command position is 0 (180 + (-180)).

When the master reaches a position of 450, the slave commanded position remains at 0 since the Cyclic input is false.



2.2.5.2 MC_CamOut

Description

This function block:

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

Like a MC_MoveVelocity move, the control continues to command the axis to move at this velocity until this MC_CamOut move is aborted. If this function block is called and the active move is not a MC_CamIn move, this function block returns an error and the active move is not aborted.

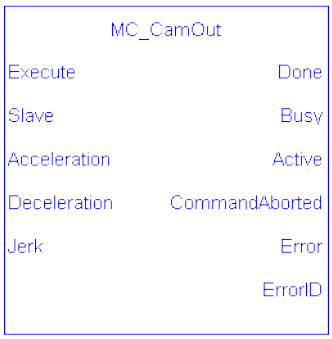


Figure 1-79: MC_CamOut

Arguments

Input

Execute	**Description**	Requests to queue the CamOut move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Slave	**Description**	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	1 – 256
	Unit	n/a

	Default	—
Acceleration	**Description**	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Deceleration	**Description**	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Jerk	**Description**	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec^3
	Default	—

Output

Done	**Description**	Indicates the axis is disengaged from its master
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Busy	**Description**	Indicates this function block is executing
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Active	**Description**	Indicates this move is the Active move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
CommandAborted	**Description**	Indicates the move was aborted
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Error	**Description**	Indicates an invalid input was specified or no MC_CamIn move was active
	Data type	BOOL
	Range	0, 1
	Unit	n/a
ErrorID	**Description**	Indicates the error if the Error output is high
	Data type	INT

Range	—
Unit	n/a

Usage

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

Related Functions

MC_CamIn

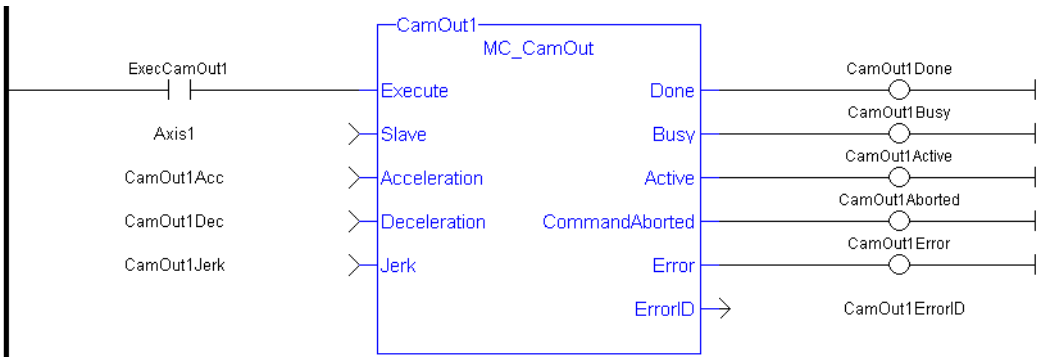
MC_CamTblSelect

Example

Structured Text

```
(* MC_CamOut ST example *)
Inst_MC_CamOut(ExecCamOut1,Axis1,CamOut1Acc,CamOut1Dec,CamOut1Jerk);
//Inst_MC_CamOut is an instance of MC_CamOut
```

Ladder Diagram



See also MC_CamIn for examples.

2.2.5.3 MC_CamResumePos

Description

This function block returns the slave axis position for the most recently executed "MC_CamIn" (➜ p. 358) 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.

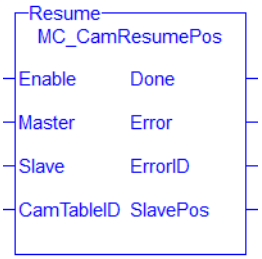


Figure 1-80: MC_CamStartPos

Related Functions

"MC_CamIn" (➜ p. 358)

"MC_CamStartPos" (➜ p. 369)

Arguments

Inputs

Enable	**Description**	Enables execution of the function block
	Data Type	BOOL
	Range	n/a
	Units	n/a
	Default	—
Master	**Description**	Master axis. This must be the same as the Master Axis specified for the most recently executed MC_CamIn function block.
	Data Type	AXIS_REF
	Range	[1,256] for .AXIS_NUM
	Units	n/a
	Default	—
Slave	**Description**	Slave axis. This must be the same as the Slave Axis specified for the most recently executed MC_CamIn function block.
	Data Type	AXIS_REF
	Range	[1,256] for .AXIS_NUM
	Units	n/a
	Default	—
CamTableID	**Description**	Profile ID number. This value was generated by "MC_CamTblSelect" (➜ p. 372). This must be the same as the CamTableID specified for the most recently executed MC_CamIn function block.
	Data Type	INT
	Range	[0,255]
	Units	n/a
	Default	—

Outputs

Done	**Description**	TRUE = the function block has successfully calculated the slave position. The slave position is available at the SlavePos output.

	Data Type	BOOL
	Units	n/a
Error	**Description**	TRUE = an invalid input was specified or an error occurred in the calculations. The value at the SlavePos output is undefined.
	Data Type	BOOL
	Units	n/a
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See table in PLCopen Function Block ErrorID Output
	Data Type	INT
	Units	n/a
SlavePos	**Description**	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.
	Data Type	LREAL
	Units	User Units

Examples

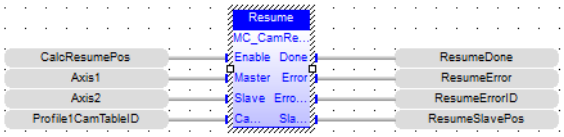
ST

```
Inst_MC_CamStartPos( TRUE, Axis1, Axis2, Profile1CamTableID);
```

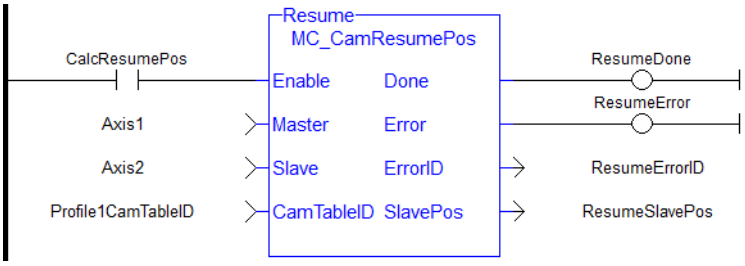
IL

```
CAL Inst_MC_CamResumePos( TRUE, Axis1, Axis2, Profile1CamTable ID)
```

FBD



FFLD



2.2.5.4 MC_CamStartPos

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. 358) move with StartMode = 0 (Start mode).

The typical application of MC_CamStartPos is to aid in positioning a slave axis to its starting position for a MC_CamIn move with a slave absolute profile. See Positining an Axis Before Starting Camming for complete instructions.

Figure 1-81: MC_CamStartPos

Arguments

Inputs

Enable	**Description**	Enables execution of the function block
	Data Type	BOOL
	Range	n/a
	Units	n/a
	Default	—
Master	**Description**	Master axis
	Data Type	AXIS_REF
	Range	[1,256] for .AXIS_NUM
	Units	n/a
	Default	—
Slave	**Description**	Slave axis
	Data Type	AXIS_REF
	Range	[1,256] for .AXIS_NUM
	Units	n/a
	Default	—
MasterOffset	**Description**	Master axis offset
	Data Type	LREAL
	Range	—
	Units	User Units
	Default	—
SlaveOffset	**Description**	Slave axis offset
	Data Type	LREAL
	Range	—

	Units	User Units
	Default	—
MasterScaling	**Description**	Master axis scale factor. Scaling must be a positive value that is greater than 0.
	Data Type	LREAL
	Range	—
	Units	User Units
	Default	—
SlaveScaling	**Description**	Slave axis scale factor. Scaling must be a positive value that is greater than 0.
	Data Type	LREAL
	Range	—
	Units	User Units
	Default	—
CamTableID	**Description**	Profile ID number. This number was generated by "MC_CamTblSelect" (➜ p. 372).
	Data Type	INT
	Range	[0,255]
	Units	n/a
	Default	—

Outputs

Done	**Description**	TRUE = the function block has successfully calculated the slave position. The slave position is available at the SlavePos output.
	Data Type	BOOL
	Units	n/a
Error	**Description**	TRUE = an invalid input was specified or an error occurred in the calculations. The value at the SlavePos output is undefined.
	Data Type	BOOL
	Units	n/a
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See table in PLCopen Function Block ErrorID Output
	Data Type	INT
	Units	n/a
SlavePos	**Description**	If the Done output is TRUE, this output returns the position for the slave axis given the profile and the current master axis position.
	Data Type	LREAL
	Units	User Units

Examples

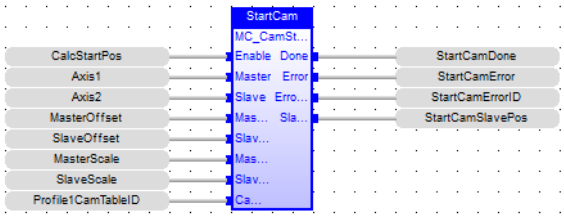
ST

```
Inst_MC_CamStartPos( TRUE, Axis1, Axis2, MasterOffset, SlaveOffset,
MasterScale, SlaveScale, Profile1CamTableID);
```

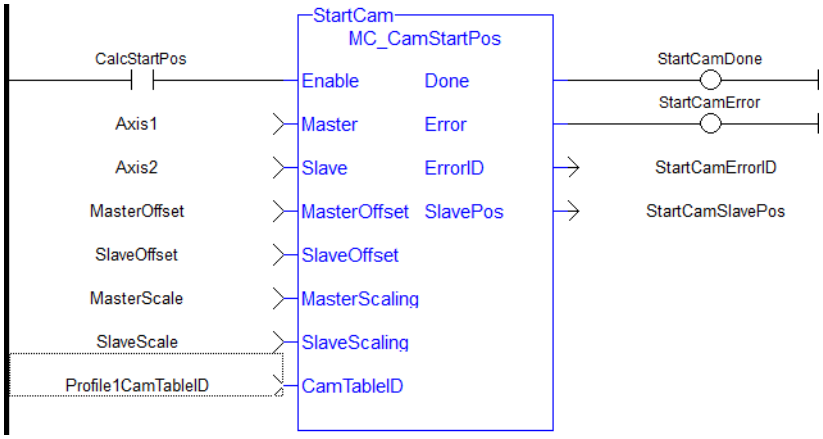
IL

```
CAL Inst_MC_CamStartPos( TRUE, Axis1, Axis2, MasterOffset, SlaveOffset,
MasterScale, SlaveScale, Profile1CamTable ID)
```

FBD



FFLD



2.2.5.5 MC_CamTblSelect

Description

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

Arguments

Input

Execute	**Description**	Requests to queue the slave gear ratio move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
CamTable	**Description**	Profile name as defined in the CAM Profile Properties dialog
	Data type	STRING

	Range	—
	Unit	n/a
	Default	—
Periodic	**Description**	Selects if the profile is periodic (see also Usage section)
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
MasterAbsolute	**Description**	Selects if master profile is absolute or relative (see also Usage section)
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
SlaveAbsolute	**Description**	Selects if Slave profile is absolute or relative (see also Usage section)
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—

Output

Done	**Description**	Indicates the function block has completed successfully
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Busy	**Description**	Indicates this function block is executing
	Data type	BOOL
	Range	0, 1
	Unit	n/a
Error	**Description**	Indicates an invalid input was specified
	Data type	BOOL
	Range	0, 1
	Unit	n/a
ErrorID	**Description**	Indicates the error if the Error output is high
	Data type	INT
	Range	—
	Unit	n/a
CamTableID	**Description**	Indicates the ID number of the profile to be used with MC_CamIn
	Data type	INT
	Range	0 - 255
	Unit	n/a

Usage

- Each positive transition of the **Enable** input will create a unique Cam ID and store the profile information in a table. The number of unique Cam IDs is limited to 256. If the application attempts to create more than 256 Cam IDs, the **Error** output will be true and the **ErrorID** output will be `22` (`Too Many Profiles`). It is only necessary to call MC_CamTblSelect once for each Profile/Periodic/MasterAbsolute/SlaveAbsolute configuration to be used.
- The **Periodic** input selects if the profile is to repeat each cycle. If the profile is not periodic and the master axis moves beyond the profile range, the slave stops at the end of the profile.

NOTE

If the master axis moves back into the profile range, the slave resumes following the profile.

- If the **MasterAbsolute** input is ON, the profile is in reference to the Master axis position.
  If the MasterAbsolute input is OFF, the profile is in reference to the Master axis position at the time the MC_CamIn function block is executed.
- Similarly, the **SlaveAbsolute** input selects if the slave positions are in reference to the Slave axis position or the Slave axis position at the time the MC_CamIn function block is executed.

TIP

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

Related Functions

MC_CamIn

MC_CamOut

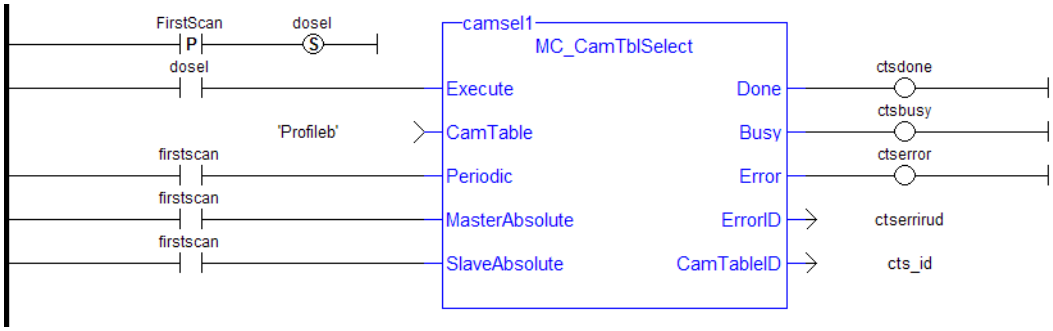
Example

Structured Text

```
(* MC_CamTblSelect ST example *) //call this function block every scan
until "Done"
Inst_MC_CamTblSelect(DoSelect, 'Profileb', TRUE, TRUE, TRUE ); //Inst_MC_
CamTblSelect is instance of MC_CamTblSelect
CamSelDone := Inst_MC_CamTblSelect.Done;  //store Done output to user
defined variable
IF CamSelDone = TRUE THEN//when function block is "done" store
CamTableID := Inst_MC_CamTblSelect.CamTableID;  //CamTableID in user
defined variable
END_IF;
```

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

Ladder Diagram



See also MC_CamIn for examples.

2.2.5.6 MC_GearIn

Description

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

```
SlaveCommandPosition = MasterActualPosition * RatioNumerator / RatioDe-
nominator
```

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.

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.

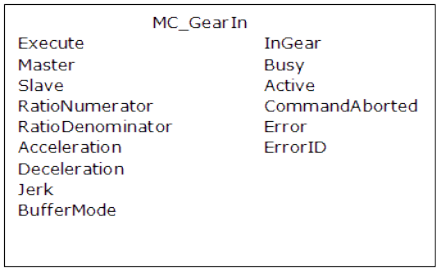
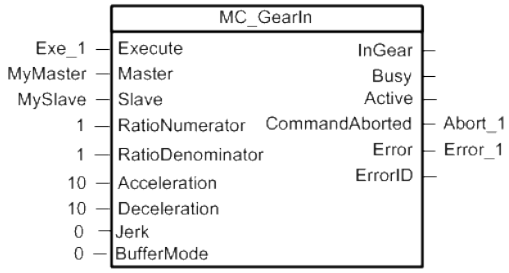
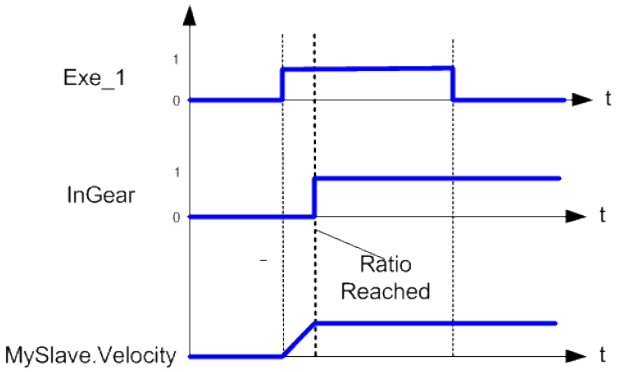


Figure 1-82: MC_GearIn

Time Diagram





Arguments

Input

Execute	**Description**	Requests to queue the slave gear ratio move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Master	**Description**	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Slave	**Description**	AXIS_REF.AXIS_NUM is the slave axis number
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
RatioNumerator	**Description**	Numerator of master/slave ratio
	Data type	DINT
	Range	[-2147483648, 2147483647]
	Unit	n/a
	Default	—
RatioDenominator	**Description**	Denominator of master/slave ratio
	Data type	DINT
	Range	[-2147483648, 2147483647]
	Unit	n/a
	Default	—
Acceleration	**Description**	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—

Deceleration	**Description**	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Jerk	**Description**	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec^3
	Default	—
BufferMode	**Description**	0 = abort 1 = buffer
	Data type	SINT
	Range	[0,1]
	Unit	n/a
	Default	—
Slave	**Description**	AXIS_REF.AXIS_NUM is the slave axis number
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
RatioDenominator	**Description**	Denominator of master/slave ratio
	Data type	DINT
	Range	[-2147483648, 2147483647]
	Unit	n/a
	Default	—

Output

InGear	**Description**	Indicates the slave axis is locked on to the master axis
	Data type	BOOL
Busy	**Description**	High from the moment the Execute input goes high until the time the move is ended
	Data type	BOOL
Active	**Description**	Indicates this move is the Active move
	Data type	BOOL
CommandAborted	**Description**	Indicates the move was aborted
	Data type	BOOL
Error	**Description**	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE
	Data type	INT
Active	**Description**	Indicates this move is the Active move

Data type BOOL

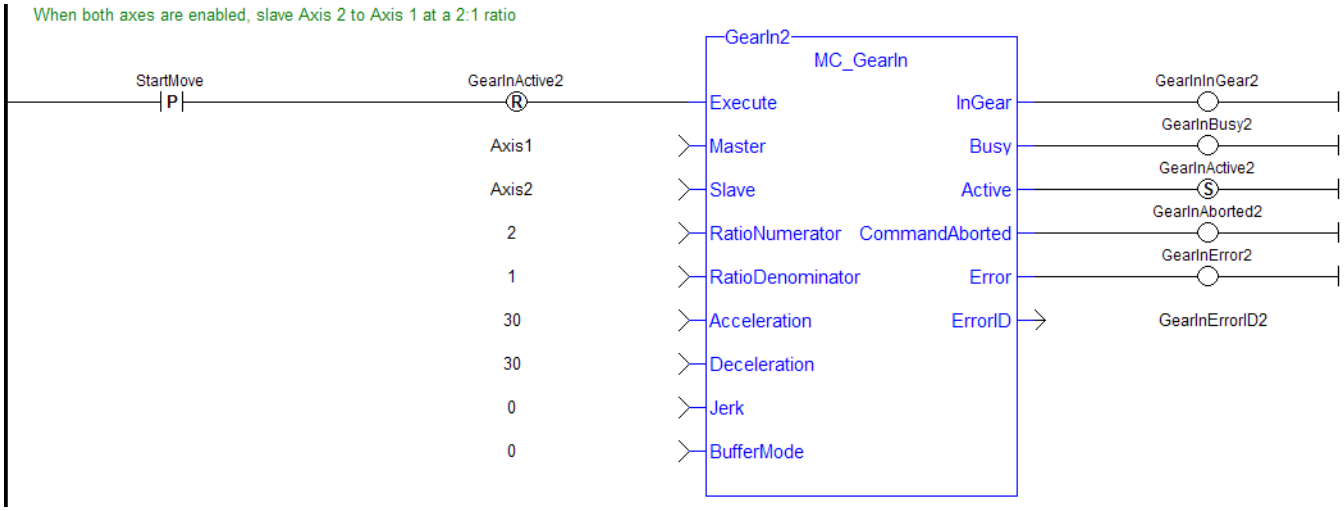
Example

Structured Text

```
(* MC_GearIn ST example *)
  Inst_MC_GearIn( GearInReq, Axis1, Axis2, 2, 1, 150.0, 150.0, 0, 0 );
//Inst_MC_GearIn is an instance of MC_GearIn
```

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

Ladder Diagram



2.2.5.7 MC_GearInPos

Description

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

```
SlaveCommandPosition = MasterActualPosition * RatioNumerator / RatioDe-
nominator
```

This function block also allows the application to specify sync positions for the master and slave axes. It is the point in which the master and slave axes become engaged in synchronous motion. When the master axis reaches the MasterStartDistance from the MasterSyncPosition, the slave axis begins to accelerate to the target velocity determined by the master axis velocity and the ratio. The slave axis arrives at the target velocity and the SlaveSyncPosition at the same time the master axis arrives at the MasterSyncPosition. At that time, the slave is locked on to the master and follows the master at the ratio specified. The slave axis continues to follow the master axis until this move is aborted.

Time to Reach the Target Velocity

While following the master, gearing functions can generate large accelerations. If the gearing function is aborted while the axis is currently accelerating, and the aborting function block has small non-zero Jerk or small acceleration values, it can take a long time to reach the target velocity, or position of the aborting function block. If the Jerk and/or acceleration of the aborting function cannot be increased to suitable values, it may be desirable to:

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

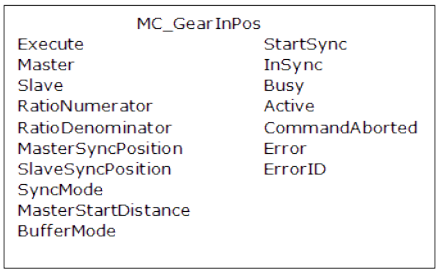
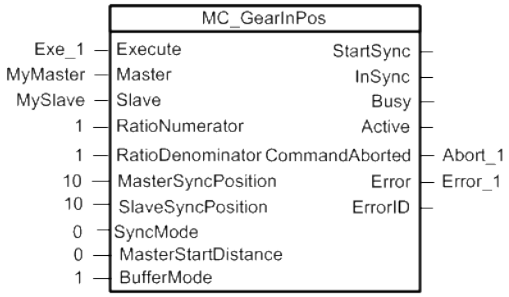
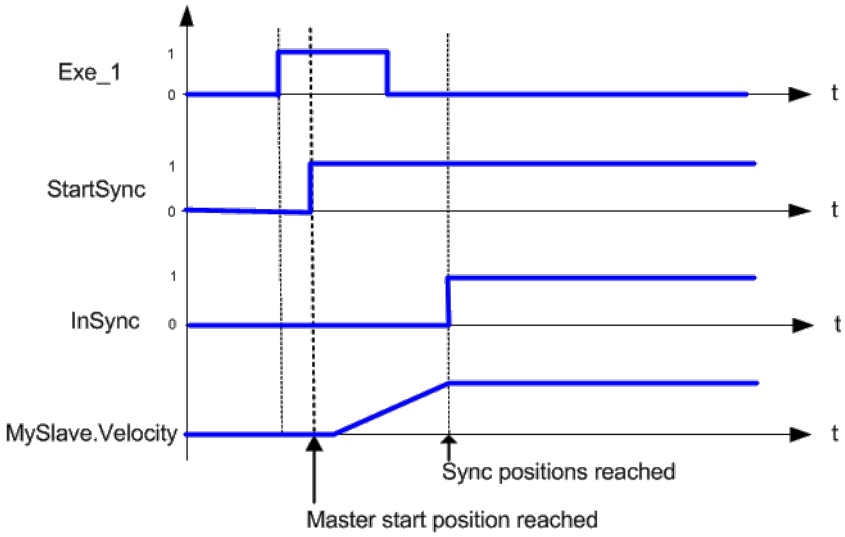


Figure 1-83: MC_GearInPos

Time Diagram





Arguments

Input

Execute	**Description**	Requests to queue the slave gear ratio move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Master	**Description**	Name of a declared instance of the AXIS_REF library function

	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Slave	**Description**	AXIS_REF.AXIS_NUM is the slave axis number
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
RatioNumerator	**Description**	Numerator of master/slave ratio
	Data type	DINT
	Range	[-2147483648, 2147483647]
	Unit	n/a
	Default	—
RatioDenominator	**Description**	Denominator of master/slave ratio
	Data type	DINT
	Range	[-2147483648, 2147483647]
	Unit	n/a
	Default	—
MasterSyncPosition	**Description**	Master axis sync position
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—
SlaveSyncPosition	**Description**	Slave axis sync position
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—

SyncMode	**Description**	SyncMode determines the allowed conditions for synchronization: **0 = Normal synchronization.** Prior to executing the MC_GearInPos function block, the *Master* axis position must be before the *MasterSyncPosition* by a distance greater than the *MasterStartDistance*. The *Slave* axis position must be before the *SlaveSyncPosition*. 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. *MasterStartDistance* must be positive and greater than the distance the master axis is currently moving per axis update. If the master start distance and the slave axis distance from the *SlaveSyncPosition* are sufficiently large enough, the slave axis will ramp to the sync position. If not sufficiently large enough, acceleration of the slave axis may be excessive. **1 = Immediate synchronization allowed.** This mode is only allowed if both the 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 *MasterStartDistance* may be 0. If the MasterStartDistance is zero, the Slave axis will synchronize with the master the instant the master axis crosses the MasterSyncPosition. If either the master or slave axis are beyond their respective sync start positions, the slave axis will immediately synchronize to the master axis. If the master start distance and the slave axis distance from the *SlaveSyncPosition* 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. Excessive slave acceleration may also occur if the master axis velocity is large or the master and slave axes have disproportionally 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.
	Data type	INT
	Range	0-1
	Unit	n/a
	Default	—
MasterStartDistance	**Description**	When the master axis reaches this distance before MasterSyncPosition, the slave axis begins its lock-on process
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
BufferMode	**Description**	1 = buffer
	Data type	SINT
	Range	[1]
	Unit	n/a
	Default	—

Output

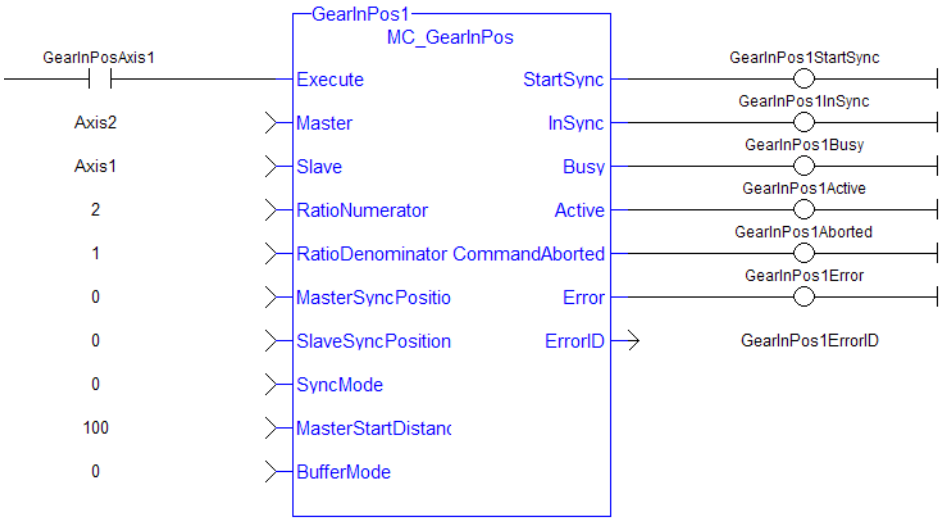
StartSync	**Description**	Indicates that the master axis has reached the Master-StartDistance from the MasterSyncPosition and the lock-on process has begun
	Data type	BOOL
InSync	**Description**	Indicated the slave axis is locked on to the master axis
	Data type	BOOL
Busy	**Description**	High from the moment the Execute input goes high until the time the move is ended
	Data type	BOOL
Active	**Description**	Indicates this move is the Active move
	Data type	BOOL
CommandAborted	**Description**	Indicates the move was aborted. If the abort arises because the inputs cause inconsistent motion, then this FB: • performs no motion • sets an error flag • set the ErrorID to 13
	Data type	BOOL
Error	**Description**	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE
	Data type	INT

Example

Structured Text

```
(* MC_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
```

Ladder Diagram



2.2.5.8 MC_GearOut

Description

This function block:

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

Like a MC_MoveVelocity move, the control continues to command the axis to move at this velocity until this MC_GearOut move is aborted. The Acceleration, Deceleration and Jerk input parameters are applied if this command velocity is modified by the MC_SetOverride function block. If this function block is called and the active move is not a MC_GearIn or MC_GearInPos move, this function block returns an error and the active move is not aborted.

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.

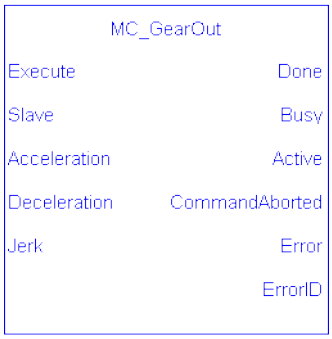


Figure 1-84: MC_GearOut

Arguments

Input

Execute	**Description**	Requests to disengage the slave axis from a MC_GearIn or MC_GearInPos move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Slave	**Description**	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Acceleration	**Description**	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Deceleration	**Description**	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Jerk	**Description**	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec^3
	Default	—

Output

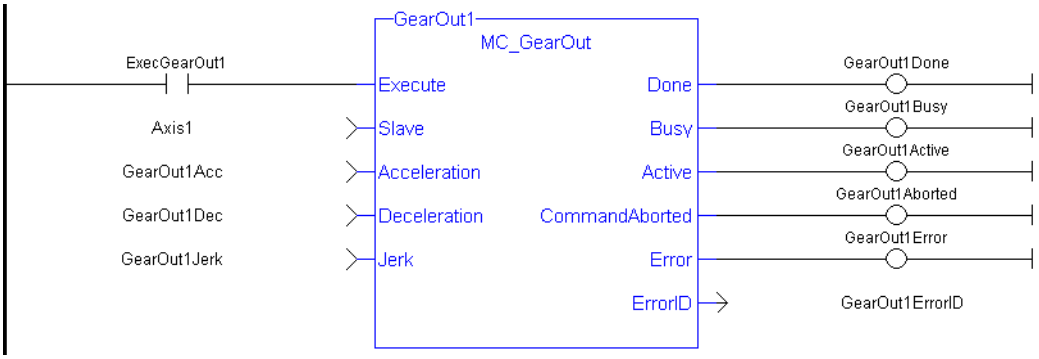
Done	**Description**	Indicates the axis is disengaged from its master
	Data type	BOOL
Busy	**Description**	Indicates the function is executing
	Data type	BOOL
Active	**Description**	Indicates this move is the Active move
	Data type	BOOL
CommandAborted	**Description**	Indicates the move was aborted
	Data type	BOOL
Error	**Description**	Indicates an invalid input was specified or no MC_GearIn or MC_GearInPos move is active
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE
	Data type	INT

Example

Structured Text

```
(* MC_GearOut ST example *)
Inst_MC_GearOut(ExecGearOut1,Axis1,GearOut1Acc,GearOut1Dec,GearOut1Jerk);
//Inst_MC_GearOut is instance of MC_GearOut
```

Ladder Diagram



2.2.5.9 MC_Phasing

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. 346) 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.

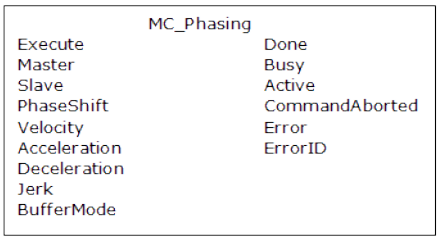


Figure 1-85: MC_Phasing

Arguments

Input

Execute

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

Master	**Description**	Name of a declared instance of the AXIS_REF library function.)
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
Slave	**Description**	AXIS_REF AXIS_NUM is the slave axis number
	Data type	AXIS_REF
	Range	[1,256]
	Unit	n/a
	Default	—
PhaseShift	**Description**	Amount of phase shift
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
Velocity	**Description**	Velocity setpoint
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	**Description**	Trapezoidal: Acceleration rate S-curve: Maximum acceleration
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Deceleration	**Description**	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Jerk	**Description**	Trapezoidal: 0 S-curve: Constant jerk
	Data type	LREAL
	Range	—
	Unit	User unit/sec^3
	Default	—
BufferMode	**Description**	0. abort 1. buffer 2. blend to acrive 3. blend to next 4. blend to low velocity 5. blend to high velocity
	Data type	SINT
	Range	[0,5]
	Unit	n/a

	Default	—

Output

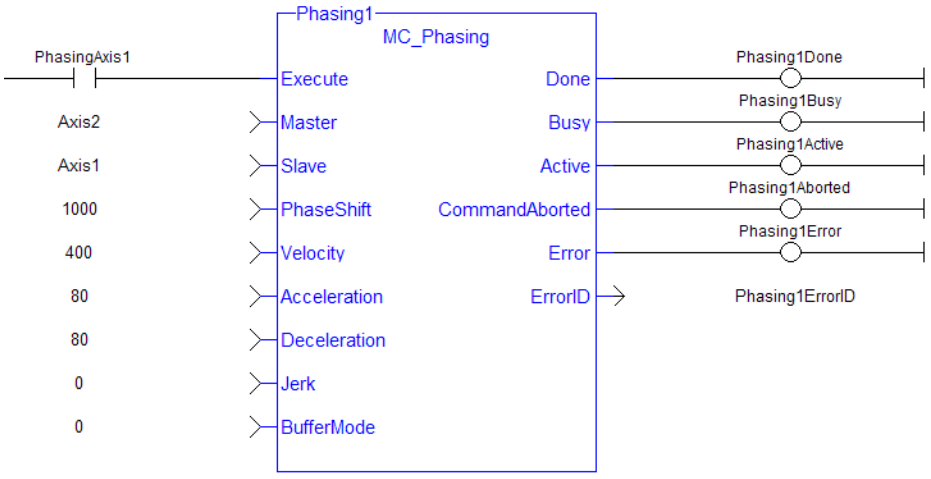
Done	**Description**	Indicates the phase shift has been completely applied
	Data type	BOOL
Busy	**Description**	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
Active	**Description**	Indicates this phase shift is the active phase shift
	Data type	BOOL
CommandAborted	**Description**	Indicates the move was aborted
	Data type	BOOL
Error	**Description**	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE
	Data type	INT

Example

Structured Text

```
(* MC_Phasing ST example *) //Inst_MC_Phasing is an instance of MC_Phasing function block
Inst_MC_Phasing(PhasingAxis1, Axis2, Axis1, 1000.0,100.0, 200.0, 200.0, 0, 0 );
```

Ladder Diagram



2.2.5.10 MC_SyncSlaves

Description

This function block allows the application to specify what slave axes are to be synchronized and which master they follow. After this function block is executed successfully, all the slave axes specified at the SlaveList

input start their slave moves (i.e. MC_GearIn, MC_CamIn, etc.) on the same servo interrupt for a synchronized slave start. When a slave move is commanded for one of the slave axes listed, the slave move is queued but the motion is held off until all of the listed slaves have queued their slave moves.

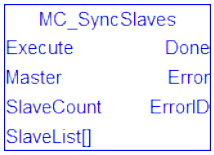


Figure 1-86: MC_SyncSlaves

Arguments

Input

Execute	Description	A positive transition of this input causes the function block to execute
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Master	Description	Master axis identifier
	Data type	AXIS_REF
	Range	1 - 256
	Unit	n/a
	Default	—
SlaveCount	Description	The number of slave axes listed in the SlaveList array input that are to be synchronized. This number must not be greater than the declared size of the SlaveList array. If this number is 0, the list of synchronized slaves for the specified Master axis is cleared.
	Data type	AXIS_REF
	Range	1-256 The AXIS_NUM element of the AXIS_REF structure must be in the range [1-256]
	Unit	n/a
	Default	—
SlaveList	Description	The list of slave axes that are to be synchronized. Each element of this array contains a unique axis number. The axis number must not be the same as the Master axis number.
	Data type	UINT
	Range	1-32
	Unit	n/a
	Default	—

Output

Done	Description	Indicates the synchronized slave assignments were completed without error
	Data type	BOOL
Error	Description	Indicates an invalid input was specified
	Data type	BOOL

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

Usage

Call MC_SyncSlaves to specify the slave axes to synchronize.

Call each slave move (e.g. MC_GearIn) for each slave axis. The motion is held off until all the slave moves have been queued.

After all the slave moves have been queued, the interpolation for all the slave axes begin on the same servo interrupt, providing a synchronized start.

The master axis can be in motion prior to this sequence, or the master can be commanded after all the slave moves are queued.

Related Functions

MC_GearIn

MC_GearInPos

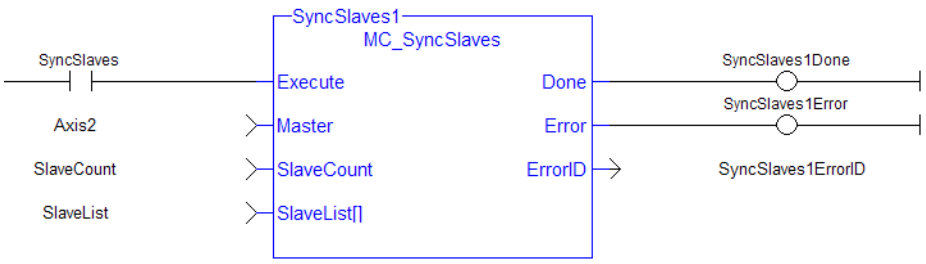
MC_CamIn

Example

Structured Text

```
(* MC_SyncSlaves ST example *)
// Inst_MC_SyncSlaves is an instance of MC_SyncSlaves function block
Inst_MC_SyncSlaves( SyncSlaves, Axis1, SlaveCount, SlaveList );
```

Ladder Diagram



2.2.6 Reference Functions

This set of functions provides commands for reference points.

2.2.6.1 MC_Reference

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.

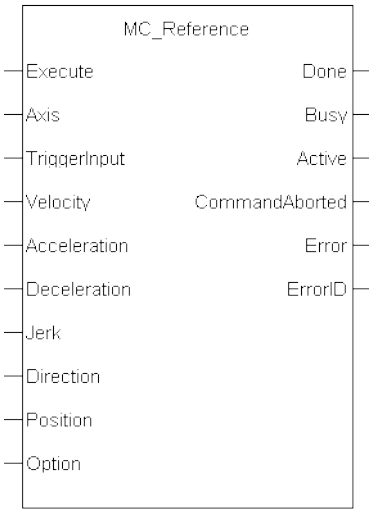


Figure 1-87: MC_Reference

Arguments

Input

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

Velocity	Description	Commanded velocity for the reference move
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	Description	Commanded acceleration for the reference move
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Deceleration	Description	Commanded deceleration for the reference move
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Jerk	Description	Commanded jerk for the reference move (if zero, then trapezoidal acc/dec is used)
	Data type	LREAL
	Range	—
	Unit	User unit/sec^3
	Default	—
Direction	Description	Commanded Direction of the reference
	Data type	SINT
	Range	[0,1]
	Unit	n/a
	Default	—
Position	Description	Position of the axis at the reference location
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
Option	Description	Option identifier for Resolvers/Modulo reference. 0 = Use latched position for reference 1 = use resolver position of nearest null for reference 2 pole resolver 2 = use resolver position of nearest null for reference 4 pole resolver 3 = use resolver position of nearest null for reference 6 pole resolver 4 = use resolver position of nearest null for reference 8 pole resolver 5 = use resolver position of nearest null for reference 10 pole resolver … 15 = use resolver position of nearest null for reference 30 pole resolver

Data type	SINT
Range	[0,15]
Unit	n/a
Default	—

Output

Done	Description	Indicates the reference move and position adjustment is complete
	Data type	BOOL
Busy	Description	Indicates this function block is executing
	Data type	BOOL
Active	Description	Indicates this move is the Active move
	Data type	BOOL
CommandAborted	Description	Indicates the move was aborted
	Data type	BOOL
Error	Description	Indicates an invalid input, or the move was terminated due to an error
	Data type	BOOL
ErrorID	Description	Indicates the error if the Error output is high
	Data type	INT

Usage

The following lists the steps for homing a PLCopen axis, using the MC_Reference function block. Not all of the steps are necessary depending on the configuration and the homing cycle design.

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

- Ensure Axis is not on Reference switch.
  If a switch is used in the homing cycle for the event or precondition to the event, check to ensure the axis is not already tripping the switches that trigger the event and precondition. If it is, move the axis off the switches.
- Configure AKD capture engine
  Configuration of the AKD capture engine is performed by writing drive CAN objects via SDO. It is accomplished with the ECATWriteSdo function. **The AKD Capture mode must be set to POSITION CAPTURE.**
  The available configurations are discussed in paragraph "**AKD Capture Engine Configuration**" .
  Example AKD capture engine configurations and reference examples are discussed in paragraph "**PLCopen Homing Methods**".

- Call the MC_REFERENCE function to initiate optional homing motion and to arm the AKD capture engine
  The MC_Reference function block selects the trigger edge (rising or falling edge) and arm the capture. Then, it optionally moves the axis to the reference location as directed by inputs to this function. When the AKD indicates that the capture event has occurred, the coordinate system is shifted so that the reference position input to this function block is set to the reference location. Then, the reference motion is stopped.
- Wait for the completion of the MC_Reference function block
  The application is notified by the completion, abort or error of the homing by the MC_Reference function block.
- Upon completion of the MC_Reference function block, the axis can be moved to the home position with a MC_MoveAbsolute function block.

TIP

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

Related Functions

ECATWriteSdo

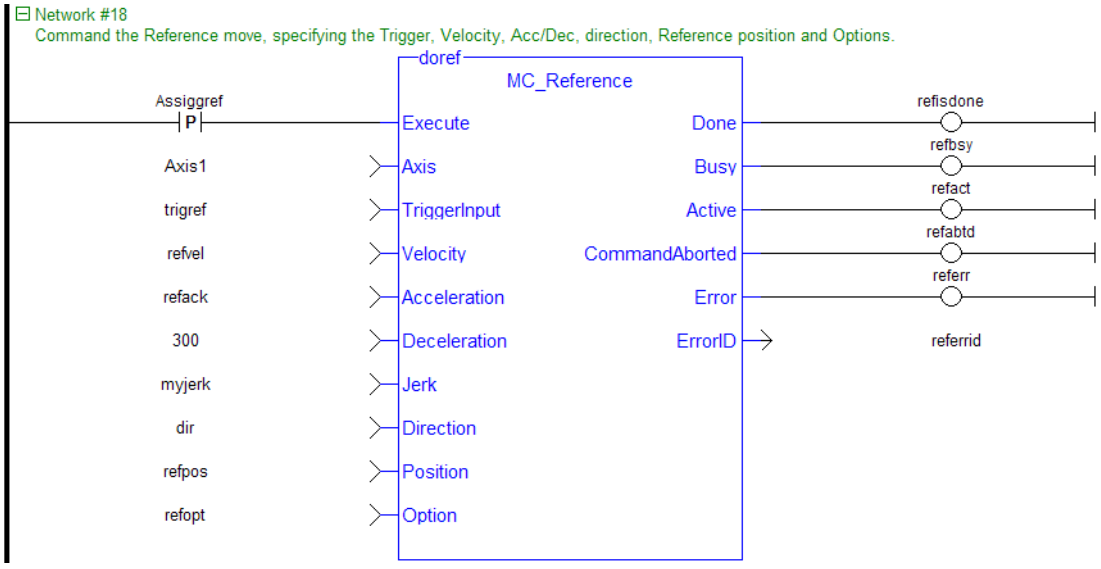
MC_MoveAbsolute

Example

Structured Text

```
(* MC_Reference ST example *)
TriggerInput.InputID := 0; //configure the reference InputID
TriggerInput.DIRECTION := 1; //configure the reference direction
Inst_MC_Reference( RefReq, Axis1, TriggerInput, 20.0, 100.0, 100.0,
100.0, 0, 0.0, 0 );
```

Ladder Diagram



2.2.6.2 MC_SetPos

Description

This function block sets the axis position to the position specified at the Position input. It is a no-motion reference. This function block replaces the MC_SetPosition function. This function also clears the accumulated Superimposed distance value for the input axis see table in Axis Parameters. This function block replaces the MC_SetPosition function.

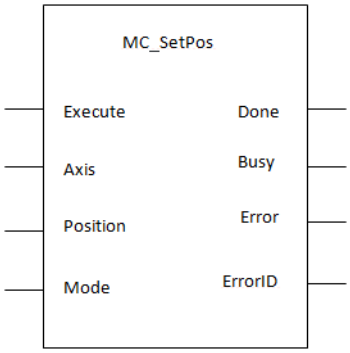


Figure 1-88: MC_SetPos

Arguments

For more detail on how inputs and outputs work, refer to PLCopen Function Blocks - General Rules.

Inputs

En	**Description**	Requests to change the axis position
	Data type	BOOL
	Range	0,1
	Unit	n/a
	Default	—
Axis	**Description**	Name of a declared instance of the AXIS_REF library function. For more details Modify PLCopen Axis.
	Data type	AXIS_REF Structure
	Range	[1,256]
	Unit	n/a
	Default	—
Position	**Description**	New axis position (absolute or relative)
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—
Mode	**Description**	LOW = value at Position is an absolute position HIGH = value at Position is a relative position
	Data type	BOOL
	Range	—
	Unit	n/a
	Default	—

Outputs

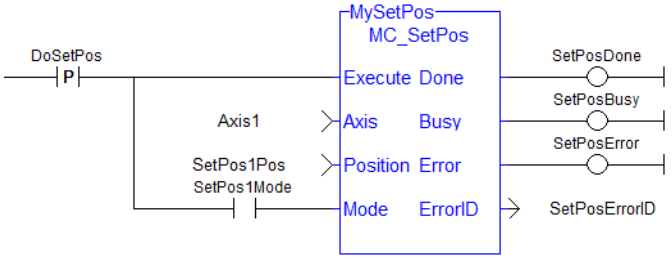
Done	**Description**	Indicates the reference move and position adjustment is complete
	Data type	BOOL
Busy	**Description**	Indicates this function block is executing
	Data type	BOOL
Error	**Description**	Indicates an invalid input, or the move was terminated due to an error
	Data type	BOOL
ErrorID	**Description**	Indicates the error if the Error output is high See table in PLCopen Function Block ErrorID Output
	Data type	INT

Example

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
```

Ladder Diagram



2.2.6.3 MC_SetPosition (Function)

Description

This function has been deprecated by the "MC_SetPos" (➜ p. 394) function block.

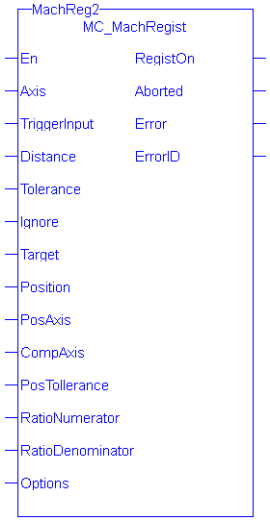
2.2.7 Registration Function Blocks

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

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.

Arguments

Input

En	**Description**	Rising edge of EN enables execution
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	**Description**	Axis whose position is used to determine a good mark.
	Data type	Axis_Ref
	Range	The range of .AXIS_NUM is [1,256]
	Unit	n/a
	Default	n/a

TriggerInput	**Description**	Structure specifying the fast input. The structure elements are: **InputID** INT 0 = Capture Engine 0 1 = Capture Engine 1 Range is [0,1] For information on configuring the capture engines, refer to AKD Capture Engine Configuration. **Direction** 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] **TrigID** INT Axis number of the fast input. Zero indicates this trigger axis is to be the same as the Axis input. range = [0,256]
	Data type	TRIGGER_REF
	Range	
	Unit	
	Default	
Distance	**Description**	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.
	Data type	LREAL
	Range	When converted to feedback units, the range is $[-2^{51},2^{51}-1]$. This value must have the same sign as Ignore.
	Unit	user units
	Default	n/a
Tolerance	**Description**	This value specifies the distance, plus or minus, about Distance to determine if the mark detected by the fast input is a good mark.
	Data type	LREAL
	Range	When converted to feedback units, the range is $[0 ,2^{51}-1]$
	Unit	user units
	Default	n/a
Ignore	**Description**	This value specifies the distance after the previous good mark in which any detected marks are ignored.
	Data type	LREAL
	Range	When converted to feedback units, the range is $[-2^{51},2^{51}-1]$. This value must have the same sign as Distance.
	Unit	user units
	Default	n/a
Target	**Description**	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.
	Data type	LREAL
	Range	When converted to feedback units, the range is: • $[-2^{51},2^{51}-1]$ if PosAxis' rollover value is zero • [0, PosAxis' Rollover Value] if PosAxis' rollover value is non-zero (i.e., ≥ 0 < PosAxis' Rollover Value)

	Unit	user units
	Default	n/a
Position	**Description**	The position the axis is set to when a good registration mark occurs.
	Data type	LREAL
	Range	When converted to feedback units, the range is: • $[-2^{51},2^{51}-1]$ if PosAxis' rollover value is zero • [0, PosAxis' Rollover Value] if PosAxis' rollover value is non-zero (i.e., ≥ 0 < PosAxis' Rollover Value)
	Unit	user units
	Default	n/a
PosAxis	**Description**	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.
	Data type	AXIS_REF
	Range	The range of .AXIS_NUM is [1,256]
	Unit	n/a
	Default	n/a
CompAxis	**Description**	The calculated registration compensation is applied to this axis.
	Data type	AXIS_REF
	Range	The range of .AXIS_NUM is [1,256]
	Unit	n/a
	Default	n/a
PosTolerance	**Description**	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.
	Data type	LREAL
	Range	When converted to feedback units, the range is $[-2^{51},2^{51}-1]$
	Unit	user units
	Default	n/a
RatioNumerator	**Description**	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.
	Data type	DINT
	Range	When converted to feedback units, the range is [1,4294967295]
	Unit	user units
	Default	n/a
RatioDenominator	**Description**	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.
	Data type	DINT
	Range	When converted to feedback units, the range is [1,4294967295]
	Unit	user units

	Default	n/a
Options	**Description**	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.
	Data type	UINT
	Range	refer to the following options table
	Unit	n/a
	Default	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
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

Outputs

RegistOn	**Description**	Indicates registration is activated
	Data type	BOOL
Aborted	**Description**	Indicates registration has been terminated by MC_StopRegist.
	Data type	BOOL
Error	**Description**	Indicates an invalid input was specified or registration was terminated due to an error
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is TRUE. See table in PLCopen Function Block ErrorID Output.
	Data type	INT

Related Functions

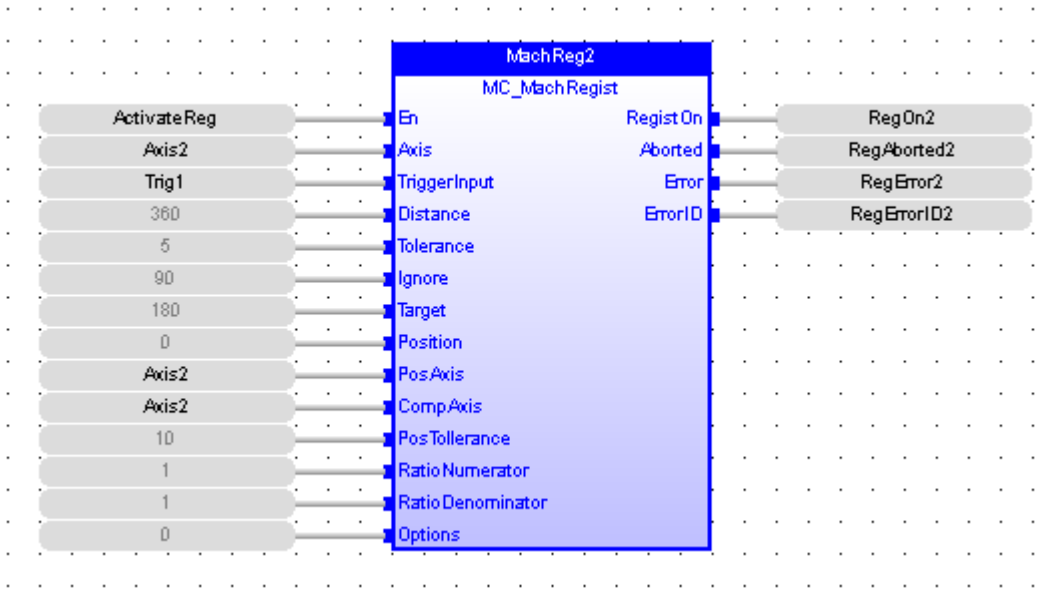
MC_ReadParam

MC_StopRegist

MC_WriteParam

Examples

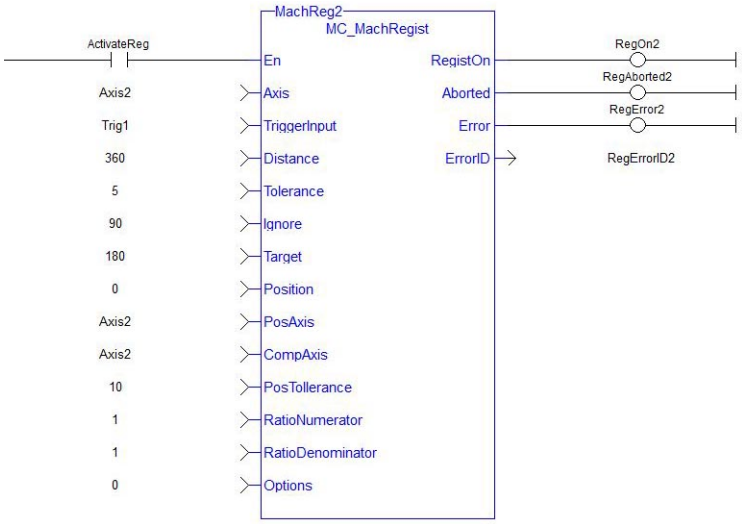
Function Block



Instruction List

```
CAL Inst_MC_MachRegist( ActivateReg, Axis2, Trig1, 360, 5, 90, 180, 0, Axis2, Axis2, 5, 1, 1, 0 )
```

Ladder Diagram



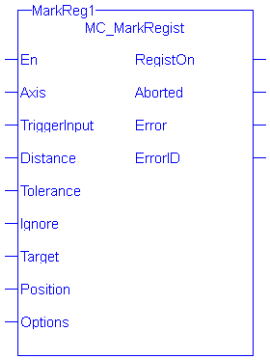
Structured Text

```
Inst_MC_MachRegist( ActivateReg, Axis2, Trig1, 360, 5, 90, 180, 0, Axis2, Axis2, 10,
```

2.2.7.3 MC_MarkRegist

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.

Arguments

Input

En	**Description**	Rising edge of EN enables execution
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	**Description**	Axis to apply registration to
	Data type	AXIS_REF
	Range	The range of .AXIS_NUM is [1,256]
	Unit	n/a
	Default	n/a
TriggerInput	**Description**	Structure specifying the fast input. The structure elements are: **InputID** INT 0 = Capture Engine 0 1 = Capture Engine 1 Range is [0,1] For information on configuring the capture engines, refer to AKD Capture Engine Configuration. **Direction** 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] **TrigID** INT Axis number of the fast input. Zero indicates this trigger axis is to be the same as the Axis input. range = [0,256]
	Data type	TRIGGER_REF
	Range	
	Unit	
	Default	
Distance	**Description**	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.
	Data type	LREAL

	Range	When converted to feedback units, the range is [-2^{51},2^{51}-1]. This value must have the same sign as Ignore.
	Unit	user units
	Default	n/a
Tolerance	**Description**	This value specifies the distance, plus or minus, about Distance to determine if the mark detected by the fast input is a good mark.
	Data type	LREAL
	Range	When converted to feedback units, the range is [0 ,2^{51}-1]
	Unit	user units
	Default	n/a
Ignore	**Description**	This value specifies the distance after the previous good mark in which any detected marks are ignored.
	Data type	LREAL
	Range	When converted to feedback units, the range is [-2^{51},2^{51}-1]. This value must have the same sign as Distance.
	Unit	user units
	Default	n/a
Target	**Description**	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.
	Data type	LREAL
	Range	When converted to feedback units, the range is [-2^{51},2^{51}-1]. This value must have the same sign as Distance.
	Unit	user units
	Default	n/a
Position	**Description**	The position the axis is set to when a good registration mark occurs
	Data type	LREAL
	Range	When converted to feedback units, the range is: • [-2^{51},2^{51}-1] if PosAxis' rollover value is zero • [0, PosAxis' Rollover Value] if PosAxis' rollover value is non-zero (i.e., ≥ 0 < PosAxis' Rollover Value)
	Unit	user units
	Default	n/a
Options	**Description**	Each bit enables/disables an option. The following table defines the bits. Any bits not defined are reserved.
	Data type	UINT
	Range	Refer to the following options table.
	Unit	n/a
	Default	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

Hexadecimal	Decimal	Option	Description
0008 H	8	Inhibit Reference on Good Mark	0 = Perform reference, 1 = inhibit reference
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.

Outputs

RegistOn	**Description**	Indicates that registration is active.
	Data type	BOOL
Aborted	**Description**	Indicates registration has been terminated by MC_StopRegist.
	Data type	BOOL
Error	**Description**	Indicates an invalid input was specified or registration was terminated due to an error
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is TRUE. See table in PLCopen Function Block ErrorID Output.
	Data type	INT

Related Functions

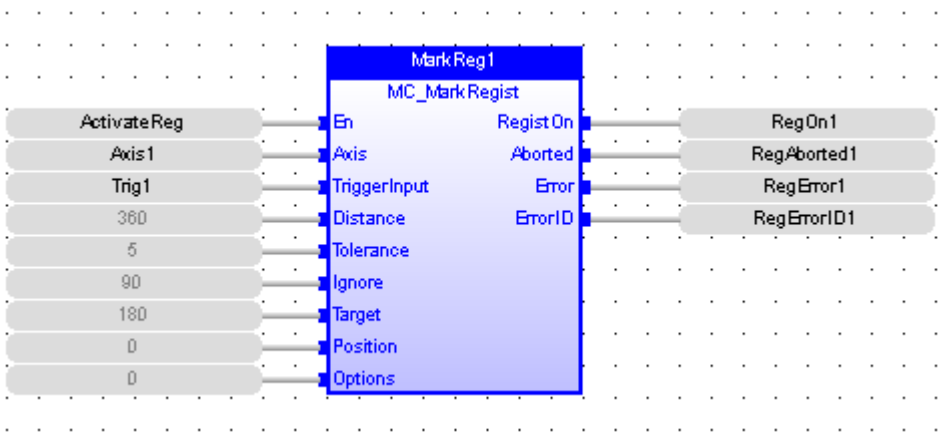
MC_ReadParam

MC_Stop

MC_WriteParam

Examples

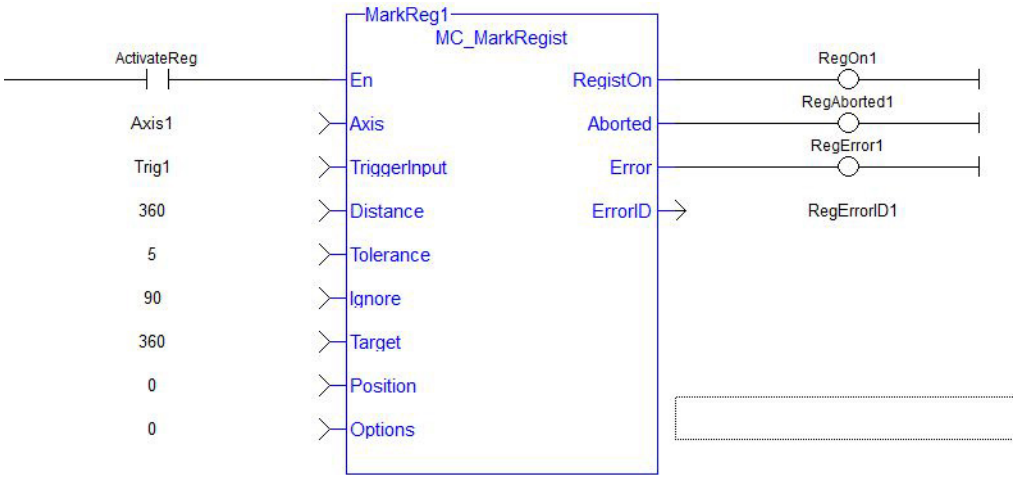
Function Block



Instruction List

```
CAL Inst_MC_MarkRegist( ActivateReg, Axis1, Trig1, 360, 5, 90, 360, 0, 0 )
```

Ladder Diagram



Structured Text

```
Inst_MC_MarkRegist( ActivateReg, Axis1, Trig1, 360, 5, 90, 360, 0, 0 );
```

2.2.7.4 MC_StopRegist

Description

This function will turn off registration for the specified axis and disarm the specified fast input.

Arguments

Input

En	**Description**	Enables execution
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	**Description**	Axis registration to turn off
	Data type	AXIS_REF
	Range	The range of .AXIS_NUM is [1,256]
	Unit	n/a
	Default	n/a

TriggerInput	**Description**	Structure specifying the fast input to disarm. The structure elements are: InputID INT 0 = Capture Engine 0 1 = Capture Engine 1 Range is [0,1] For information on configuring the capture engines, refer to AKD Capture Engine Configuration. Direction 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] TrigID INT Axis number of the fast input. 0 indicates this trigger axis is to be the same as the Axis input. range = [0,256]
	Data type	TRIGGER_REF
	Range	
	Unit	
	Default	

Outputs

OK	**Description**	Indicates function executed successfully
	Data type	BOOL

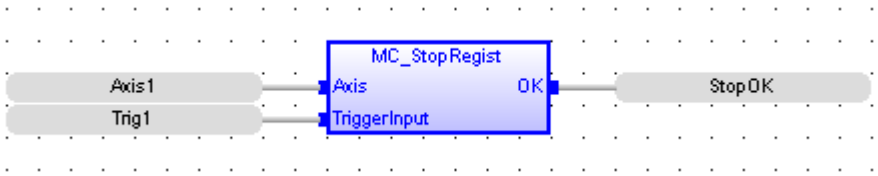
Related Functions

MC_MachRegist

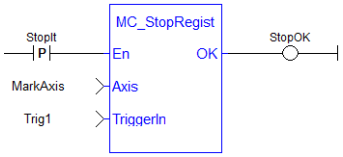
MC_MarkRegist

Examples

Function Block



Ladder Diagram



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. 408), " MC_RemSuperAxis" (➜ p. 409) 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

Inputs

En	**Description**	Enables Execution
	Data Type	BOOL
	Range	n/a
	Unit	n/a
	Default	n/a
Axis	**Description**	Axis to receive the additional superimposed axis's command delta
	Data Type	AXIS_REF
	Range	.AXIS_NUM [1,256]
	Unit	n/a
	Default	n/a
SuperimposedAxis	**Description**	Axis number of the superimposed axis whose command delta will be added to delta of `Axis`
	Data Type	UINT
	Range	[1,256]
	Unit	n/a
	Default	n/a

Outputs

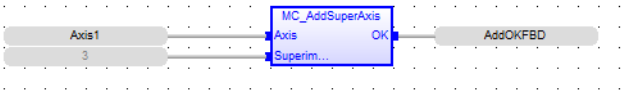
OK	**Description**	Execution successful
	Data Type	BOOL
	Range	n/a

Examples

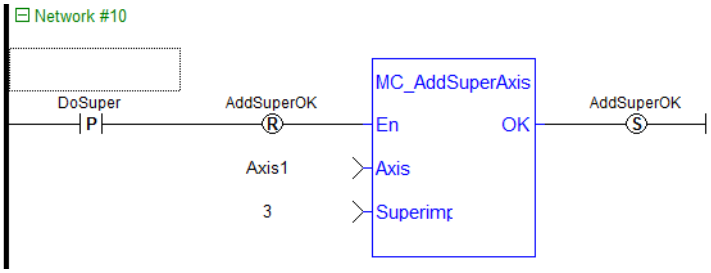
Structured Text

```
AddOKST := MC_AddSuperAxis( Axis1, 3 );
```

Function Block Diagram



Ladder Diagram



Related Functions

" MC_RemSuperAxis" (➜ p. 409)

2.2.8.2 MC_RemSuperAxis

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

Inputs

En	**Description**	Enables Execution
	Data Type	BOOL
	Range	-
	Unit	n/a
	Default	

Axis	**Description**	Axis whose list of assigned superimposed axes will be updated.
	Data Type	AXIS_REF
	Range	.AXIS_NUM [1,256]
	Unit	n/a
	Default	n/a
SuperimposedAxis	**Description**	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.
	Data Type	UINT
	Range	n/a
	Unit	n/a
	Default	n/a

Outputs

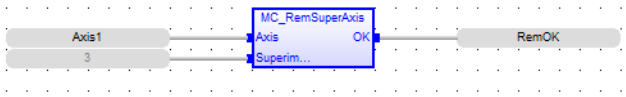
OK	**Description**	Execution successful
	Data Type	BOOL
	Range	n/a

Examples

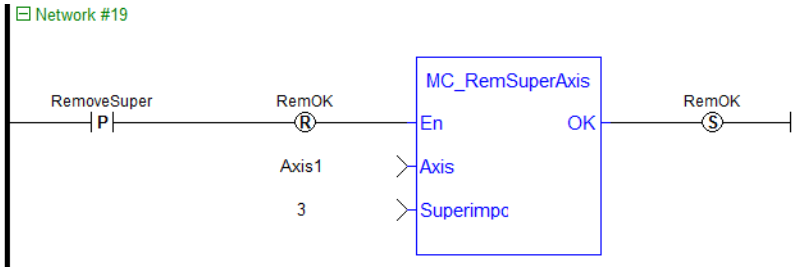
Structured Text

```
RemOK := MC_RemSuperAxis(Axis1,3);
```

Function Block Diagram



Ladder Diagram



Related Functions

" MC_AddSuperAxis" (➜ p. 408)

2.3 MotionLibrary- Common

Functions sorted in alphabetical order.

Name	Description	Return type
"MC_ErrorDescription" (➜ p. 411)	Return a text description corresponding to a motion control error ID code	STRING
"MLMotionCycleTime" (➜ p. 424)	Returns the Motion Base Cycle time in Seconds	
"MLMotionInit" (➜ p. 425)	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. 426)	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. 426)	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. 426)	Returns the status of the motion engine 0: Not initialized 1: Running 2: Stopped 3: Error	None
"MLMotionStop" (➜ p. 426)	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. 426)	Prints the system time to the log	BOOL
"MLProfileBuild" (➜ p. 413)	Builds a cam profile from application data	See "Output" (➜ p. 415)
"MLProfileCreate" (➜ p. 418)	Creates a new cam profile object	None
"MLProfileInit" (➜ p. 419)	Initializes a previously created cam profile object	BOOL
"MLProfileRelease" (➜ p. 421)	Removes a Profile so the Profile ID may be used by a different or new Profile.	See "Output" (➜ p. 422)

2.3.1 Motion Library - Common - Info

Name	Description	Return type
"MC_ErrorDescription" (➜ p. 411)	Converts the PLCopen error IDs into message strings.	String

2.3.1.1 MC_ErrorDescription

This function converts the PLCopen error IDs into message strings which can be used for display or logging.

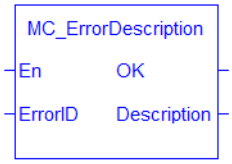


Figure 1-89: MC_ErrorDescription Function Block

Arguments

Inputs

En	**Description**	If True, then this function will convert the Error Id into a string message
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
ErrorID	**Description**	Error ID generated from a PLCopen Function Block. See PLCopen Function Block ErrorID Output for output details.
	Data type	INT
	Range	0,69
	Unit	n/a
	Default	—

Outputs

OK	**Description**	If True, then the command completed successfully.
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Description	**Description**	String error description
	Data type	STRING
	Range	—
	Unit	n/a
	Default	—

Examples

Structured Text

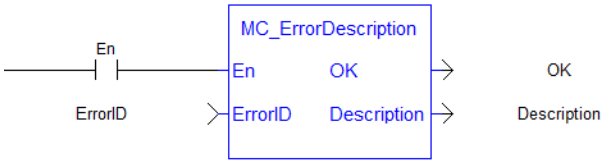
```
Description:= MC_ErrorDescription(ErrorID);
```

IL

Not applicable

Function Block

Ladder Diagram



2.3.2 Motion Library - Common - Profiles

Name	Description	Return type
"MLProfileBuild" (➜ p. 413)	Builds a cam profile from application data	See "Output" (➜ p. 415)
"MLProfileCreate" (➜ p. 418)	Creates a new cam profile object	None
"MLProfileInit" (➜ p. 419)	Initializes a previously created cam profile object	BOOL
"MLProfileRelease" (➜ p. 421)	Removes a Profile so the Profile ID may be used by a different or new Profile.	See "Output" (➜ p. 422)

2.3.2.1 MLProfileBuild

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 5th order polynomials. 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. 419)).

NOTE

Prior to using MLProfileBuild you must call "MLProfileCreate" (➜ p. 418) 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. 426) 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.

CamProps_Ref Structure

The cam properties structure (CamProps_Ref) will contain the following data members:

InputScale	LREAL	The input amplitude or x-axis multiplier applied to the CAM profile
OutputScale	LREAL	The output amplitude or y-axis multiplier applied to the CAM profile
InputOffset	LREAL	input offset or x-axis shift applied to the CAM profile
OutputOffset	LREAL	The output offset or y-axis shift applied to the CAM profile

See Master/Input offset for more information about the parameters which transform the cam profile.

CamData_Ref Structure

The cam data structure (CamData_Ref) will be an array of structures. Each element of the structure will contain the following data members:

MasterIn	LREAL	master position of the point (in the unit range [0 - InputScale])
SlaveOut	LREAL	slave position of the point (in the unit range [0 - OutputScale])
Type	UINT	1 = Point, 2 = Line
Vel	LREAL	velocity at the point (units?)
Accel	LREAL	acceleration at the point (units?)

See Cam Profile Editor's Cam Table for more information.

Arguments

Input

Enable	**Description**	Enable execution. Starts on rising edge.
	Data Type	BOOL
	Range	
	Unit	
	Default	
Cam_Props	**Description**	Structures containing the Cam Profile Properties
	Data Type	CamProps_ref
	Range	
	Unit	
	Default	
Cam_Data	**Description**	Array of Structures containing the Cam Profile data
	Data Type	CamData_ref
	Range	N=3 to 2000 elements
	Unit	
	Default	3 elements minimum size
NumOfPts	**Description**	Number of profile points in the array that are to be used.
	Data Type	UINT
	Range	3 - 2000
	Unit	elements
	Default	3
ProfileID	**Description**	ID number of a created CAM Profile
	Data Type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	--

Cyclic	**Description**	False: one time through the profile; True: repeating profile
	Data Type	BOOL
	Range	0-1 (FALSE/TRUE)
	Unit	
	Default	
Options	**Description**	For future expansion. This must be set to Zero for the time being.
	Data Type	UINT
	Range	
	Unit	
	Default	

Output

Done	**Description**	Indication of whether or not the profile was successfully compiled and built.
	Data Type	BOOL
	Range	0-1 (FALSE/TRUE)
	Unit	
Busy	**Description**	Indication that the function block is executing. TRUE if executing. False if not executing.
	Data Type	BOOL
	Range	0-1 (FALSE/TRUE)
	Unit	
Err	**Description**	Indication that the function did not execute correctly. ErrorID output will be valid and indicate the reason.
	Data Type	BOOL
	Range	0-1 (FALSE/TRUE)
	Unit	
ErrorID	**Description**	Indication of the reason for the failure to execute properly. See "Error Codes" (➔ p. 415) table.
	Data Type	INT
	Range	
	Unit	
ErrorElem	**Description**	The array element number of the cam data where an error is detected
	Data Type	UINT
	Range	
	Unit	

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	Too many points specified. Cam data array does not contain that many points.
101	Invalid master or slave amplitude. Amplitude is less than zero.
102	Point is outside amplitude range.
103	Segment type is not a point or line. Valid type is a 1 or 2.
104	Point too close to previous point i0.0005 * master-amplitude is the minimum distance between points).
105	Line too close to previous point (0.0005 * master-amplitude is the minimum distance between points).
106	Invalid profile ID. Profile ID: 1. does not exist 2. has not been created yet 3. profile ID is not a profile
107	Cam data array exceeds maximum array size of 2000 data points.
109	Attempting to build a profile already containing elements. Profile needs to be released first using MLProfileRelease.
200	First point's X value not equal to zero.
201	Last point's X value does not equal value of X-amplitude.
202	Cannot modify the first point in the cam element table. Y value is less than 0 or greater than Y amplitude.
203	Cannot modify the last point in the cam element table. Y value is less than 0 or greater than Y amplitude.

Related Functions

"MLCamInit" (➜ p. 166)

"MLCamSwitch" (➜ p. 168)

"MLProfileCreate" (➜ p. 418)

"MLProfileInit" (➜ p. 419)

"MLProfileRelease" (➜ p. 421)

"MC_CamIn" (➜ p. 358)

" MC_CamOut" (➜ p. 365)

MC_CamTblSelect

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 points 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 := 1;
```

```
ProfileData[0].Velocity := 0.0;
ProfileData[0].Acceleration := 0.0;

ProfileData[1].MasterIn := 180.0;
ProfileData[1].SlaveOut := 324.0;
ProfileData[1].SegType := 1;
ProfileData[1].Velocity := 0.5;
ProfileData[1].Acceleration := -0.025;

ProfileData[2].MasterIn := 360.0;
ProfileData[2].SlaveOut := 240.0;
ProfileData[2].SegType := 1;
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.

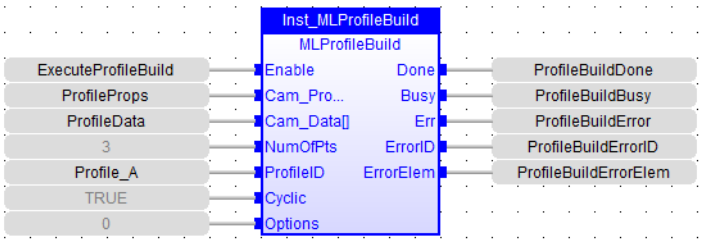
```
// Define the profile properties
ProfileProps.InputScale := 360.0;  // Must be Positive!
ProfileProps.OutputScale := 360.0; // Must be Positive!
ProfileProps.InputOffset := 0.0;
ProfileProps.OutputOffset := 0.0;
```

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.

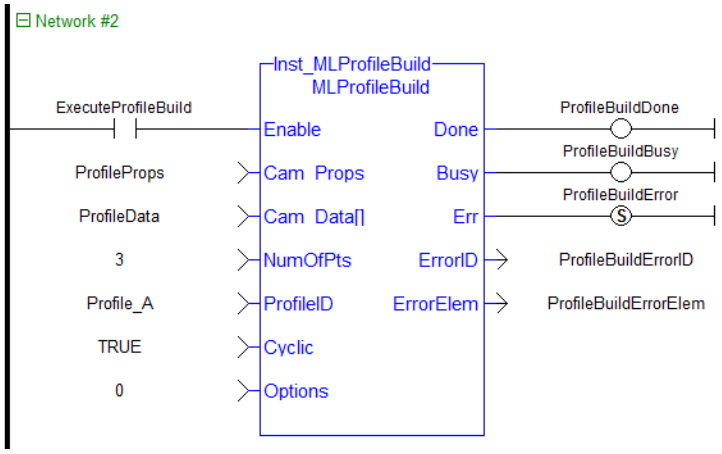
Structured Text:

```
// Build the profile
Inst_MLProfileBuild( TRUE, ProfileProps, ProfileData, 3, ProfileID, TRUE,
0);
```

Function Block Diagram:



Ladder Diagram:



Finally, after verifying that that MLProfileBuild is `Done` and there are no errors you can proceed and use the newly generated profile.

- In PLCopen your next step is to call "MC_CamTblSelect" (➜ p. 372)
- In PN your next step is to call either "MLCamInit" (➜ p. 166) or "MLCamSwitch" (➜ p. 168)

2.3.2.2 MLProfileCreate

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. 426) is called.

Arguments

Input

Name		
	Description	Name of initialized CAM Profile
	Data type	STRING
	Range	—
	Unit	n/a
	Default	—

Output

OK	**Description**	Indicates the profile has been created
	Data type	BOOL
ID	**Description**	Returns the ID number of the created CAM Profile
	Data type	DINT
	Unit	n/a

Related Functions

"MLProfileInit" (➜ p. 419)

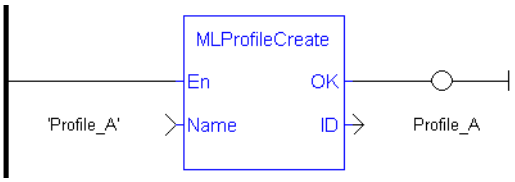
"MLCamInit" (➜ p. 166)

Example

Structured Text

```
//Create a new Profile
Profile_A := MLProfileCreate( 'Profile_A' );
```

Ladder Diagram



Function Block Diagram

2.3.2.3 MLProfileInit

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.

Arguments

Input

ProfileID	**Description**	ID number of a created CAM Profile
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—
FileName	**Description**	Filename used to save Profile on the computer's hard disk
	Data type	STRING
	Range	—
	Unit	n/a
	Default	—
InputScale	**Description**	The input amplitude or x-axis multiplier applied to the CAM Profile
	Data type	LREAL
	Range	Positive
	Unit	n/a
	Default	—
OutputScale	**Description**	The output amplitude or y-axis multiplier applied to the CAM Profile
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—
InputOffset	**Description**	The input offset or x-axis shift applied to the CAM Profile.
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—
OutputOffset	**Description**	The output offset or y-axis shift applied to the CAM Profile
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—

Output

Default (.Q)	**Description**	Returns TRUE if a new CAM Profile is initialized
	Data type	BOOL
	Unit	n/a

Return Type

BOOL

Related Functions

"MLProfileCreate" (➜ p. 418)

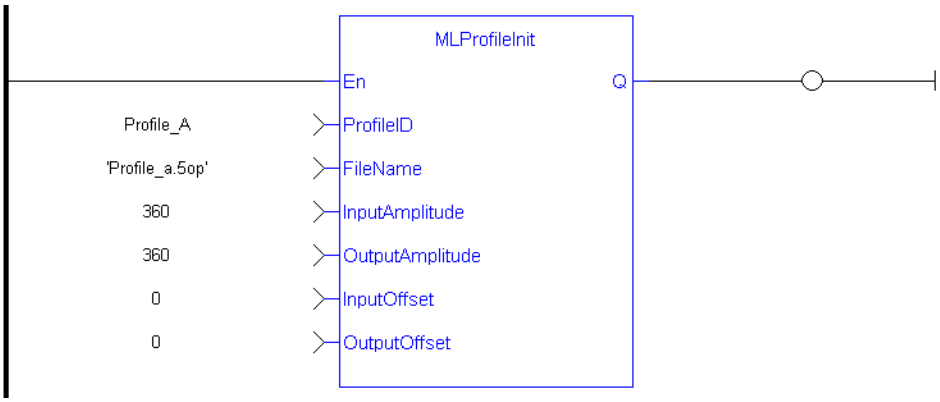
"MLCamInit" (➜ p. 166)

Example

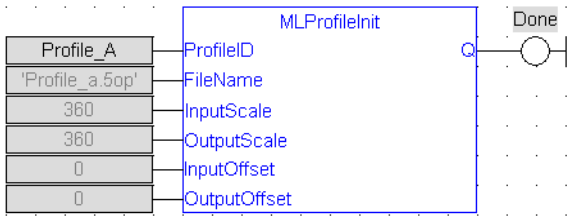
Structured Text

```
//Initialize a previously created CAM Profile
MLProfileCreate( Profile_A , 'Profile_A.5op' , 360, 360, 0, 0 );
```

Ladder Diagram



Function Block Diagram



2.3.2.4 MLProfileRelease

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. 419) or "MLProfileBuild" (➜ p. 413) 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. 168).
- For PLCOpen – selected for use by "MC_CamIn" (➜ p. 358) and has an active move.

There are a number of ways to change an in-use profile to one that is not in-use (deactived):

- For Pipe Network – Perform a "MLCamSwitch" (➜ p. 168) 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. 365) 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.

Arguments

For more information on how Arguments work, refer to PLCopen function blocks - General rules .

Input

Enable	**Description**	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.
	Data Type	BOOL
	Range	0, 1
	Unit	
	Default	0
ProfileID	**Description**	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.
	Data Type	DINT
	Range	1 to 256
	Unit	
	Default	0

Output

Done	**Description**	If high, Successful completion. The Profile can now be reused.
	Data Type	BOOL
	Range	0, 1
Err	**Description**	If high, the Function Block did not complete successfully. Reason is given in Error ID.
	Data Type	BOOL
	Range	0, 1

ErrorID

Description Indicates the reason for the failure. See "Error Codes" (➜ p. 423) table for possible reasons.

Data Type INT

Range

Error Codes

ErrorID	Description
106	Invalid profile ID. Profile ID: 1. does not exist 2. has not been created yet 3. profile ID is not a profile
108	Profile cannot be released because it is in use by the motion engine or currently selected by an active CAM block.

Related Functions

"MLProfileCreate" (➜ p. 418)

"MLProfileInit" (➜ p. 419)

"MLProfileBuild" (➜ p. 413)

"MLCamInit" (➜ p. 166)

"MC_CamTblSelect" (➜ p. 372)

"MC_CamIn" (➜ p. 358)

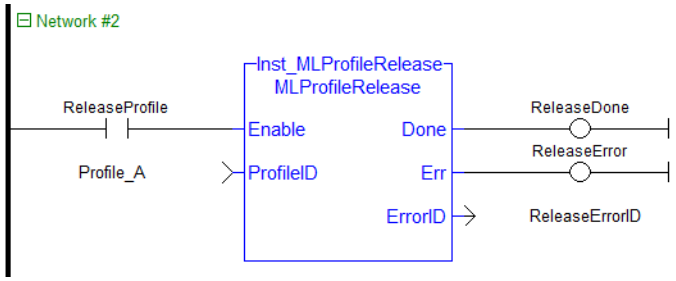
" MC_CamOut" (➜ p. 365)

Example

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;
```

Ladder Diagram



Function Block Diagram

2.3.3 Motion Library

Name	Description	Return type
"MLMotionInit" (➜ p. 425)	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. 426)	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. 426)	Returns the status of the motion engine 0: Not initialized 1: Running 2: Stopped 3: Error	None
"MLMotionStop" (➜ p. 426)	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. 426)	Prints the system time to the log	BOOL
"MLMotionCycleTime" (➜ p. 424)	Returns the Motion Base Cycle time in seconds.	

2.3.3.1 State Machine

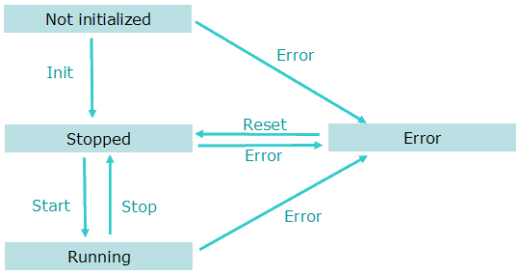


Figure 1-90: Motion State Machine

The Motion State Machine is driven by the IEC 61131-3 application with the help of the "Motion Library" (➜ p. 424) function blocks.

Each arrow represents a transition from one State to another one.

2.3.3.2 MLMotionCycleTime

Returns the Motion Base Cycle time in seconds.

Arguments

Input

Enable	**Description**	
	Data type	
	Range	
	Unit	
	Default	

Output

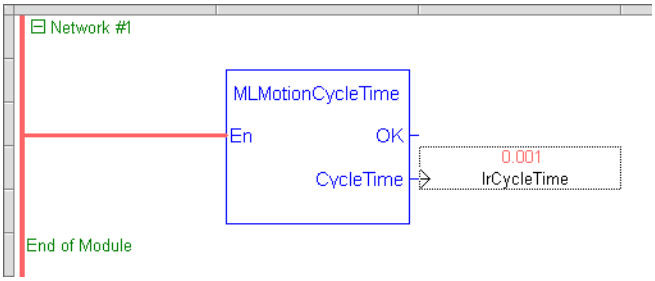
OK	**Description**	
	Data type	
CycleTime	**Description**	Cycle time in seconds
	Data type	
	Unit	

Related Functions

Example

Structured Text

Ladder Diagram



Function Block Diagram

2.3.3.3 MLMotionInit

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.

Parameter

BasePeriod : LREAL (input)

Return Type

BOOL

2.3.3.4 MLMotionRstErr

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. 426), "MLMotionStop" (➜ p. 426), "MLMotionStart" (➜ p. 426)

Return Type

BOOL

2.3.3.5 MLMotionStart

Description

Starts the motion engine, motion bus driver, 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. 426), "MLMotionRstErr" (➜ p. 426), "MLMotionStatus" (➜ p. 426)

Return Type

BOOL

2.3.3.6 MLMotionStatus

Description

Returns the status of the motion engine

0: Not initialized
1: Running
2: Stopped
3: Error

Parameter

Status : DINT (output)

Return Type

None

2.3.3.7 MLMotionStop

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. 426), "MLMotionRstErr" (➜ p. 426), "MLMotionStatus" (➜ p. 426)

Return Type

BOOL

2.3.3.8 MLMotionSysTime

Description

Prints the system time to the log. Returns always TRUE.

Return Type

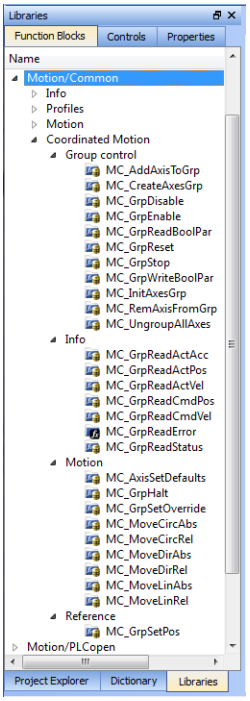
BOOL

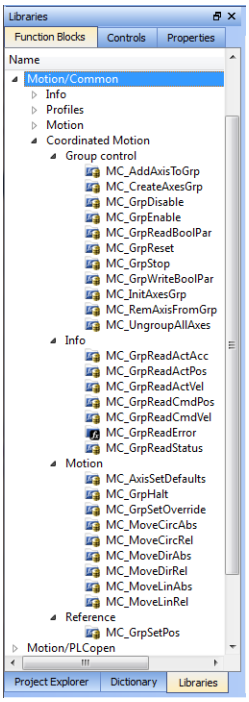
Units

milliseconds

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_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_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
"MC_AddAxisToGrp" (➜ p. 429)	Adds an axis to an axes group.
"MC_CreateAxesGrp" (➜ p. 431)	Create an axis group for coordinated motion.
"MC_GrpDisable" (➜ p. 433)	Changes the state of a group to GroupDisabled.

Function	Description
"MC_GrpEnable" (➜ p. 435)	Changes the state of a group from GroupDisabled to GroupStandby.
"MC_GrpReadBoolPar" (➜ p. 437)	Reads a value from the specified boolean group parameter
"MC_GrpReset" (➜ p. 439)	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. 441)	Performs a non-aborted, controlled motion stop on all axes in an AxesGroup.
"MC_GrpWriteBoolPar" (➜ p. 443)	Writes a value to the specified boolean group parameter.
"MC_InitAxesGrp" (➜ p. 445)	Initializes the kinematic limits for the axis group.
"MC_RemAxisFromGrp" (➜ p. 447)	Removes an individual axis from an axis group.
"MC_UngroupAllAxes" (➜ p. 449)	Removes all axes from an axes group.

MC_AddAxisToGrp

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. 431) 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. 464). 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.

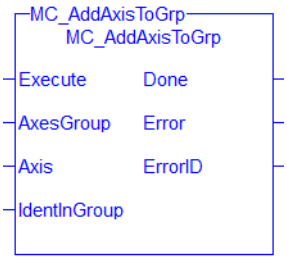


Figure 1-91: MC_AddAxisToGrp

2.3.5 Related Functions

"MC_CreateAxesGrp" (➜ p. 431), "MC_GrpReadStatus" (➜ p. 464) , "MC_RemAxisFromGrp" (➜ p. 447), "MC_UngroupAllAxes" (➜ p. 449), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

2.3.6 Input

Execute	**Description**	On the rising edge the axis is added to the group.
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	Reference to an axes group
	Data type	AXES_GROUP_REF
	Range	—
	Unit	n/a
	Default	—
Axis	**Description**	Reference to the axis to be added. An axes group cannot contain more than one instance of an axis.
	Data type	AXIS_REF
	Range	—
	Unit	n/a
	Default	—
IdentInGroup	**Description**	The zero-based index of the axis in the group. • The axis slot in the group cannot be occupied by another axis. • The index must be less than the maximum number of axes the group can contain. `MaxNumberOfAxes` is a property of the axes group and is set when the group is created. To remove an axis from a group see "MC_RemAxisFromGrp" (➜ p. 447).
	Data type	UINT
	Range	[0, MaxNumberOfAxes - 1]
	Unit	n/a
	Default	—

2.3.7 Output

Done	**Description**	If True, then the command completed successfully.
	Data type	BOOL
Error	**Description**	If True, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error identifier if Error output is set to True. See the table in PLCopen Function Block ErrorID Output.
	Data type	INT

Example

2.3.8 Structured Text

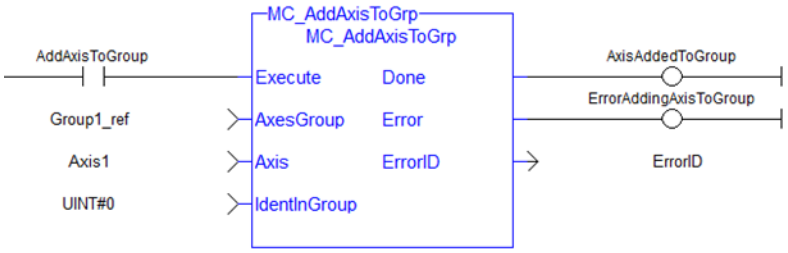
```
(*MC_AddAxisToGrp ST example *)
Inst_MC_AddAxisToGrp (AddAxisToGrp, Group1_ref, Axis_1, 0);
```

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



MC_CreateAxesGrp

Description

MC_CreateAxesGrp creates an axes group for coordinated motion.

IMPORTANT

MC_CreateAxesGrp must be called between "MLMotionInit" (➜ p. 425) and "MLMotionStart" (➜ p. 426).

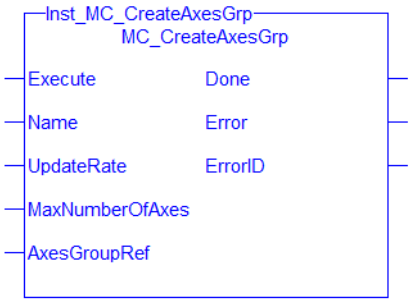


Figure 1-92: MC_CreateAxesGrp

2.3.12 Related Function Blocks

"MC_InitAxesGrp" (➜ p. 445), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

For more detail on how inputs and outputs work, refer to PLCopen Function Blocks - General Rules.

Input

Execute	**Description**	On the rising edge, this function block will create a coordinated motion axes group
	Data Type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Name	**Description**	Axes Group Name
	Data Type	STRING
	Range	—
	Unit	n/a
	Default	—
UpdateRate	**Description**	Update rate of the axes group. The group update rate will be the same as the **Base Period** specified in "MLMotionInit" (➜ p. 425). 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
	Data Type	UINT
	Range	[3,9]
	Unit	n/a
	Default	—
MaxNumberOfAxes	**Description**	The maximum number of axes that can be controlled by the group.
	Data Type	UINT
	Range	[0,6]
	Unit	n/a
	Default	—
AxesGroupRef	**Description**	The axes group reference variable to be initialized with a reference to the new axes group.
	Data Type	AXES_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—

Output

Done	**Description**	If True, then the command completed successfully.
	Date Type	BOOL
Error	**Description**	If True, then an error has occurred.
	Date Type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See the table in PLCopen Function Block ErrorID Output.
	Date Type	INT

Example

Calls to this function block are automatically generated when the application is compiled. Users should not manually call this function block.

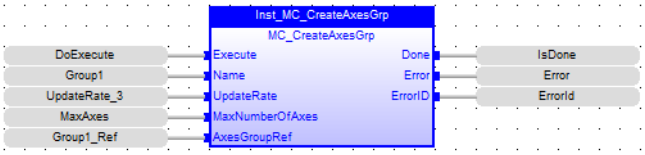
2.3.13 Structured Text

```
Inst_MC_CreateAxesGrp( DoExecute, 'Group1', UpdateRate_3, MaxAxes,
Group1_Ref);
```

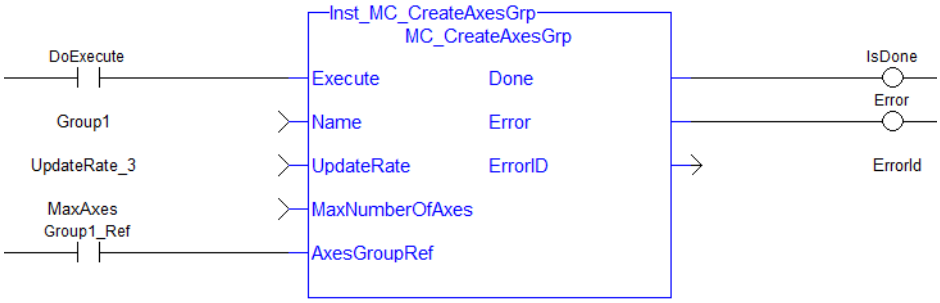
2.3.14 Instruction List

```
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



MC_GrpDisable

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.

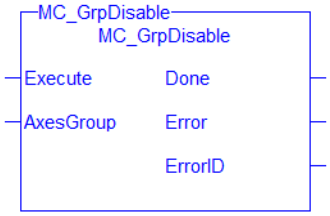


Figure 1-93: MC_GrpDisable

2.3.17 Related Functions

"MC_GrpEnable" (➜ p. 435), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

2.3.18 Input

Execute	**Description**	On the rising edge, request to disable the axis group.
	Data Type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axis group to be disabled
	Data Type	AXIS_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—

2.3.19 Output

Done	**Description**	If True, then the command completed successfully.
	Data Type	BOOL
Error	**Description**	If True, then an error has occurred.
	Data Type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See the table in PLCopen Function Block ErrorID Output
	Data Type	INT

Example

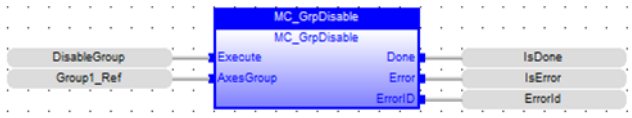
2.3.20 ST

```
(* Inst_MC_GrpDisableST example *)
Inst_MC_GrpDisable( DisableGroup, Group1_Ref );
```

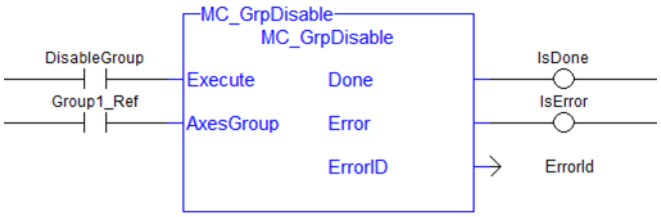
2.3.21 IL

```
BEGIN_IL
     CAL Inst_MC_GrpDisable( DisableGroup, Group1_Ref )
END_IL
```

2.3.22 FBD



2.3.23 FFLD



MC_GrpEnable

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.

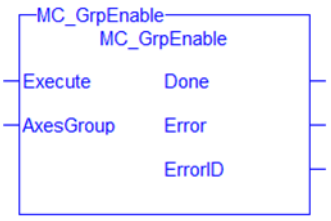


Figure 1-94: MC_GrpEnable

2.3.24 Related Functions

"MC_GrpDisable" (➜ p. 433), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

2.3.25 Input

Execute	**Description**	On the rising edge, request to enable the axis group
	Data Type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axis group to be enabled
	Data Type	AXIS_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—

2.3.26 Output

Done	**Description**	If True, then the command completed successfully.
	Data Type	BOOL
Error	**Description**	If True, an error has occurred.
	Data Type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See the table in PLCopen Function Block ErrorID Output
	Data Type	INT

Example

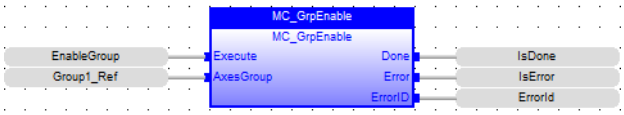
2.3.27 Structured Text

```
(* Inst_MC_GrpEnableST example *)
Inst_MC_GrpEnable( EnableGroup, Group1_Ref );
```

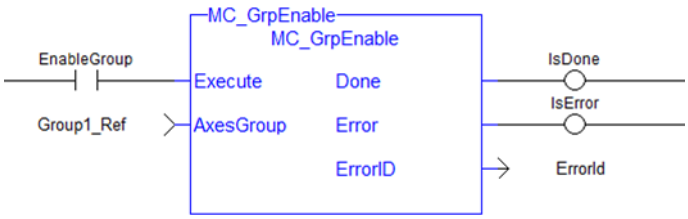
2.3.28 IL

```
BEGIN_IL
     CAL Inst_MC_GrpEnable( EnableGroup, Group1_Ref )
END_IL
```

2.3.29 FBD



2.3.30 FFLD



MC_GrpReadBoolPar

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.

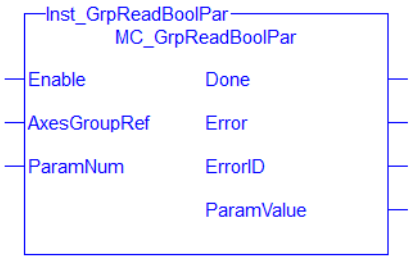


Figure 1-95: MC_GrpReadBoolPar

2.3.31 Related Function Blocks

"MC_GrpWriteBoolPar" (➜ p. 443),"MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

For more details on how inputs and outputs work, refer to PLCopen Function Blocks - General Rules.

2.3.32 Input

Enable	**Description**	If True, then request to read a value from the specified boolean group parameter.
	Data Type	BOOL
	Range	0, 1

	Unit	n/a
	Default	—
AxesGroupRef	**Description**	The axis group that the boolean parameter value will be read from.
	Data Type	AXES_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—
ParamNum	**Description**	ParamNum can be one of the following two 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.
	Data Type	UINT
	Range	1000, 1001
	Unit	UINT
	Default	—

2.3.33 Output

Done	**Description**	If True, then the command completed successfully.
	Data Type	BOOL
Error	**Description**	If True, an error has occurred.
	Data Type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See the table in PLCopen Function Block ErrorID Output.
	Data Type	INT
ParamValue	**Description**	True or False
	Data Type	BOOL

Example

2.3.34 ST

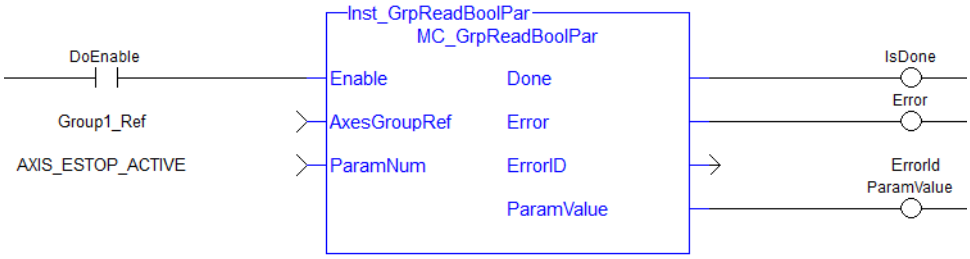
```
Inst_GrpReadBoolPar( DoEnable, Group1_Ref, AXIS_ESTOP_ACTIVE );
```

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



MC_GrpReset

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.

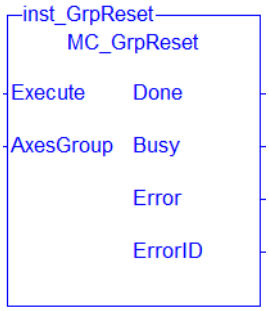


Figure 1-96: MC_GrpReset

2.3.38 Related Functions

"MC_GrpReadError" (➜ p. 463), "MC_GrpReadStatus" (➜ p. 464), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

For more details on how inputs and outputs work, refer to PLCopen Function Blocks - General Rules.

2.3.39 Input

Execute	**Description**	On the rising edge, this FB resets group-related errors and all of the axes in the group.
	Data Type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—

AxesGroup	**Description**	The axes group in which the axes will be reset.
	Data Type	AXES_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—

2.3.40 Output

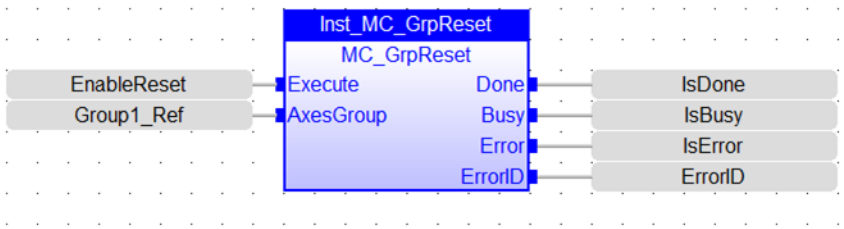
Done	**Description**	If True, then the reset completed successfully.
	Data Type	BOOL
Busy	**Description**	If True, then the FB is executing.
	Data Type	BOOL
Error	**Description**	If True, an error has occurred.
	Data Type	BOOL
ErrorID	**Description**	Indicates the error identifier if Error output is set to TRUE. See table in PLCopen Function Block ErrorID Output.
	Data Type	INT

Example

2.3.41 ST

```
Inst_MC_GrpReset ( EnableReset, Group1_Ref );
```

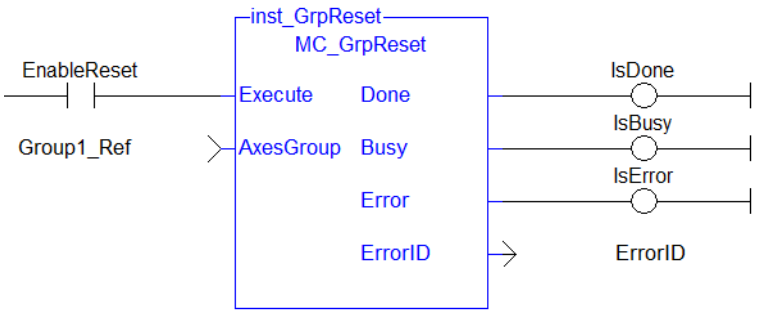
2.3.42 FBD



2.3.43 IL

```
BEGIN_IL
     CAL Inst_MC_GrpReset ( EnableReset, Group1_Ref )
END_IL
```

2.3.44 FFLD



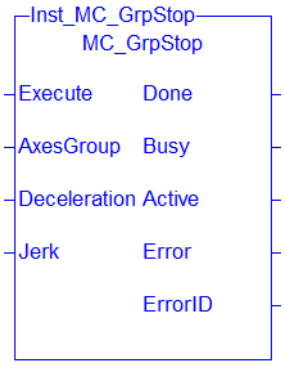
MC_GrpStop

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.



2.3.45 Related Functions

"MC_GrpHalt" (➜ p. 470), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

For more details on how inputs and outputs work, refer to PLCopen Function Blocks - General Rules.

2.3.46 Input

Execute	**Description**	On the rising edge the command to stop all of the axes in the group is initiated.
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axes group in which the axes will be stopped.
	Data type	AXES_GROUP_REF

	Range	n/a
	Unit	n/a
	Default	—
Deceleration	**Description**	The path deceleration rate for all of the axes in the group
	Data type	LREAL
	Range	0 < Deceleration See Limitations on Acceleration and Jerk for more information.
	Unit	n/a
	Default	—
Jerk	**Description**	Not supported
	Data type	LREAL
	Range	0 ≤ Jerk See Limitations on Acceleration and Jerk for more information.
	Unit	n/a
	Default	—

2.3.47 Output

Done	**Description**	If True, then the command completed successfully.
	Data type	BOOL
Busy	**Description**	TRUE from the moment the EXECUTE input is TRUE until the stop is complete.
	Data type	BOOL
Active	**Description**	If True, then the stop is still executing.
	Data type	BOOL
Error	**Description**	If True, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error identifier if Error output is set to TRUE. See table in PLCopen Function Block ErrorID Output.
	Data type	INT

Example

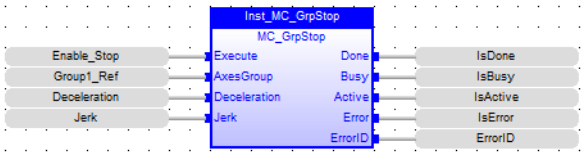
2.3.48 Structured Text

```
Inst_MC_GrpStop ( EnableStop, Group1_Ref, Deceleration, Jerk );
```

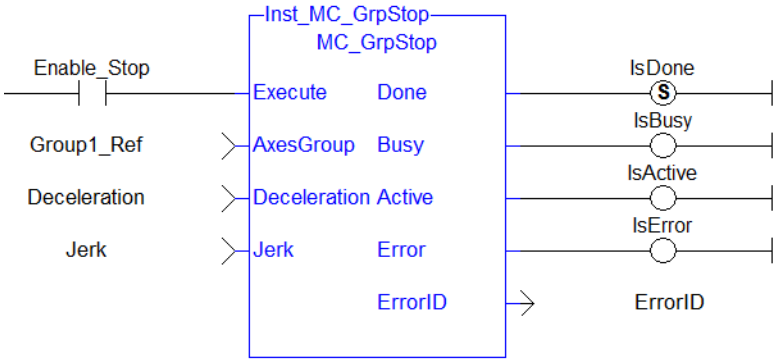
2.3.49 IL

```
BEGIN_IL
     CAL Inst_MC_GrpStop ( EnableStop, Group1_Ref, Deceleration, Jerk )
END_IL
```

2.3.50 FBD



2.3.51 FFLD



MC_GrpWriteBoolPar

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.

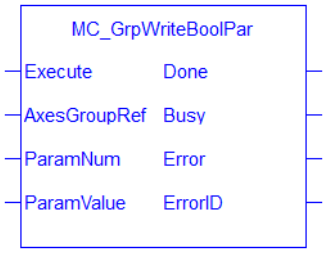


Figure 1-97: MC_GrpWriteBoolPar

2.3.52 Related Function Blocks

"MC_GrpReadBoolPar" (➜ p. 437),"MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

For more details on how inputs and outputs work, refer to PLCopen Function Blocks - General Rules.

2.3.53 Input

Execute	**Description**	On the rising edge, request to write a value to the specified boolean group parameter.
	Data Type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroupRef	**Description**	The axis group that the boolean parameter value will be written to.
	Data Type	AXES_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—
ParamNum	**Description**	The ID number of the boolean parameter that is to be written IGNORE_AXIS_ESTOP (BoolID = 1000)
	Data Type	UINT
	Range	
	Unit	
	Default	
ParamValue	**Description**	True or false
	Data Type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—

2.3.54 Output

Done	**Description**	If True, then the command completed successfully.
	Data Type	BOOL
Busy	**Description**	If True, then the function block is executing.
	Data Type	BOOL
Error	**Description**	If True, an error has occurred.
	Data Type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See the table in PLCopen Function Block ErrorID Output.
	Data Type	INT

Example

2.3.55 ST

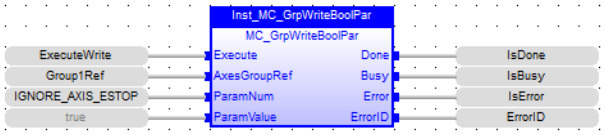
```
Inst_MC_GrpWriteBoolPar( ExecuteWrite, Group1Ref, IGNORE_AXIS_ESTOP, true
 );
```

2.3.56 IL

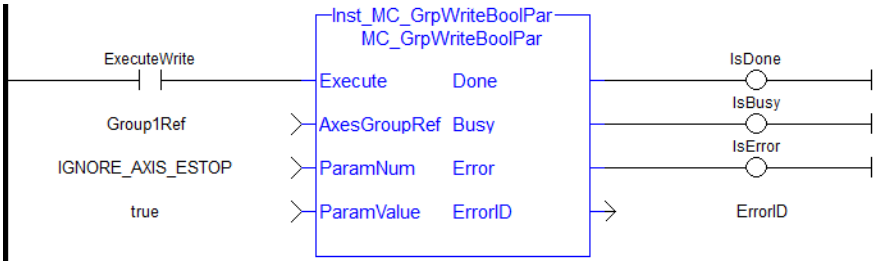
```
BEGIN_IL
     Cal Inst_MC_GrpWriteBoolPar( ExecuteWrite, Group1Ref, IGNORE_AXIS_
```

```
ESTOP, true )
END_IL
```

2.3.57 FBD



2.3.58 FFLD



MC_InitAxesGrp

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.

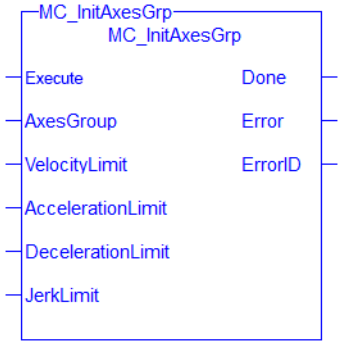


Figure 1-98: MC_InitAxesGrp

2.3.59 Related Function Blocks

"MC_CreateAxesGrp" (➜ p. 431), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

2.3.60 Inputs

Execute	**Description**	On the rising edge, this function block will initialize the axis group.
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axis group to be initialized
	Data type	AXIS_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—
VelocityLimit	**Description**	Velocity limit
	Data type	LREAL
	Range	0 < Velocity < (20 * Acceleration) and 0 < Velocity < (20 * Deceleration) See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second
	Default	—
AccelerationLimit	**Description**	Acceleration limit
	Data type	LREAL
	Range	0 < Velocity < (20 * Acceleration) See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second^2
	Default	—
DecelerationLimit	**Description**	Deceleration limit
	Data type	LREAL
	Range	0 < Velocity < (20 * Deceleration) See Limitations on Acceleration and Jerk for more information.
	Unit	User units per second^2
	Default	—
JerkLimit	**Description**	Jerk limit
	Data type	LREAL
	Range	(Velocity / 20) < Acceleration < (2 * Jerk) and (Velocity / 20) < Deceleration < (2 * Jerk) See Limitations on Acceleration and Jerk for more information.
	Unit	User units per second^3
	Default	

2.3.61 Outputs

Done	**Description**	If True, then the command completed successfully.
	Data type	BOOL
Error	**Description**	If True, then an error has occurred.
	Data type	BOOL

ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See the table in PLCopen Function Block ErrorID Output
	Data type	INT

Example

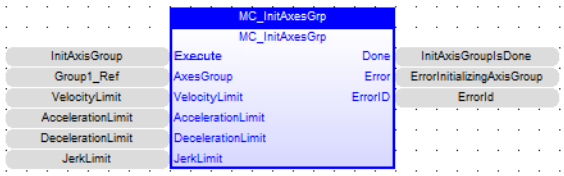
2.3.62 Structured Text

```
(* Inst_MC_InitAxesGrpST example *)
Inst_MC_InitAxesGrp( initAxesGrp, grp, velLim, accelLim, decelLim,
jerkLim );
```

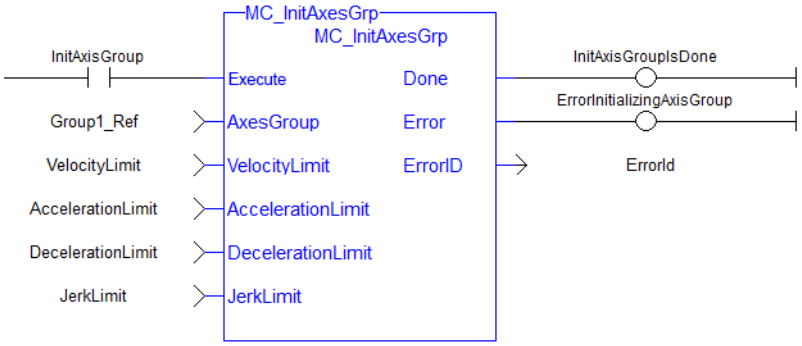
2.3.63 IL

```
BEGIN_IL
     CAL Inst_MC_InitAxesGrp( initAxesGrp, grp, velLim, accelLim,
decelLim, jerkLim )
END_IL
```

2.3.64 FBD



2.3.65 FFLD



MC_RemAxisFromGrp

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.

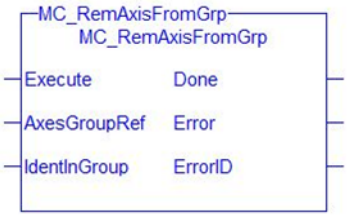


Figure 1-99: MC_RemAxisFromGrp

2.3.66 Related Functions

"MC_AddAxisToGrp" (➜ p. 429), "MC_UngroupAllAxes" (➜ p. 449), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

2.3.67 Input

Execute	**Description**	On the rising edge, request to remove an axis from the group
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroupRef	**Description**	The axis group from which the axis will be removed
	Data type	AXIS_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—
IdentInGroup	**Description**	The zero-based index of the axis in the group. • The axis index in the group must contain a valid axis • The index must be less than the maximum number of axes the group can contain. `MaxNumberOfAxes` is a property of the axes group and is set when the group is created.
	Data type	UINT
	Range	[0, MaxNumberOfAxes - 1]
	Unit	n/a
	Default	—

2.3.68 Output

Done	**Description**	If True, then the command completed successfully.
	Data type	BOOL
Error	**Description**	If True, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error identifier if Error output is set to True. See the table in PLCopen Function Block ErrorID Output.
	Data type	INT

Example

2.3.69 ST

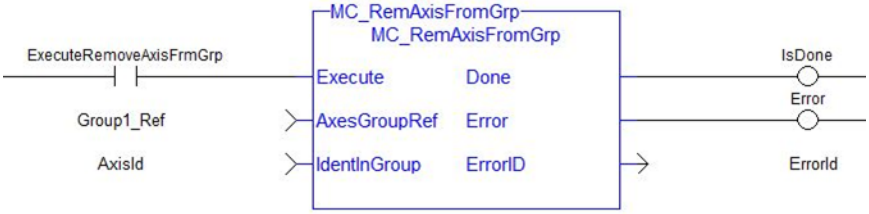
```
(* Inst_MC_InitAxisGrpST example *)
Inst_MC_RemAxisFromGrp( ExecuteRemAxisFromGrp, Group1_Ref, AxisId );
```

2.3.70 IL

```
BEGIN_IL
     CAL Inst_MC_RemAxisFromGrp( ExecuteRemAxisFromGrp, Group1_Ref,
AxisId )
END_IL
```

2.3.71 FBD

2.3.72 FFLD



MC_UngroupAllAxes

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.

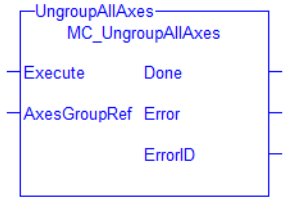


Figure 1-100: MC_UngroupAllAxes

2.3.73 Related Functions

"MC_AddAxisToGrp" (➜ p. 429), "MC_RemAxisFromGrp" (➜ p. 447), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

2.3.74 Input

Execute	**Description**	On the rising edge, request to remove all axes in the axes group
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroupRef	**Description**	The axis group from which to remove all axes
	Data type	AXIS_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—

2.3.75 Output

Done	**Description**	If True, then the command completed successfully.
	Data type	BOOL
Error	**Description**	If True, an error has occurred
	Data type	BOOL
ErrorID	**Description**	Indicates the error identifier if 'Error' output is set to TRUE. See the table in PLCopen Function Block ErrorID Output.
	Data type	INT

Examples

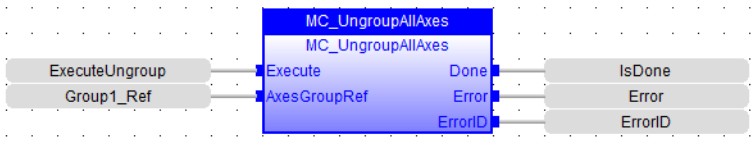
2.3.76 ST

```
Inst_MC_UngroupAllAxes( ExecuteUngroup, Group1_Ref );
```

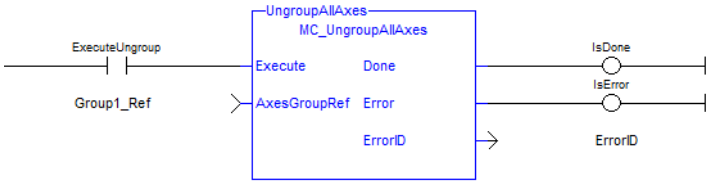
2.3.77 IL

```
BEGIN_IL
     CAL Inst_MC_UngroupAllAxes( ExecuteUngroup, Group1_Ref )
END_IL
```

2.3.78 FBD



2.3.79 FFLD



2.3.79.1 Coordinated Motion Info Library

Function	Description
"MC_GrpReadActAcc" (➜ p. 451)	Reads the actual acceleration of the group and the axes in the group.
"MC_GrpReadActPos" (➜ p. 453)	Reads the actual position of the axes in the group.
"MC_GrpReadActVel" (➜ p. 456)	Reads the actual velocity of the group and the axes in the group.
"MC_GrpReadCmdPos" (➜ p. 458)	Reads the command position of the axes in the group.
"MC_GrpReadCmdVel" (➜ p. 460)	Reads the command velocity of the axes in the group and the path velocity.
"MC_GrpReadError" (➜ p. 463)	Reads the Group ErrorID in State ERRORSTOP.
"MC_GrpReadStatus" (➜ p. 464)	Returns the status of an axes group.

MC_GrpReadActAcc

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.

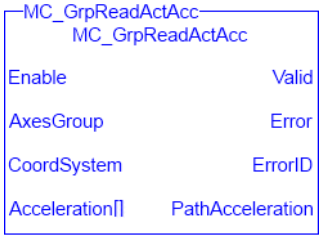
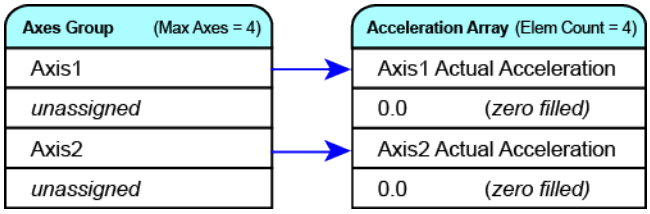


Figure 1-101: MC_GrpReadActAcc

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:



2.3.80 Related Functions

"MC_GrpReadActPos" (➜ p. 453), "MC_GrpReadActVel" (➜ p. 456), "MC_GrpReadCmdPos" (➜ p. 458), "MC_GrpReadCmdVel" (➜ p. 460)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

For more detail on how inputs and outputs work, refer to PLCopen Function Blocks - General Rules.

2.3.81 Input

Enable	**Description**	If True, then this function block will read the current actual acceleration of the group and the axes in the group
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axes group from which the actual acceleration will be read.
	Data type	AXES_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—
CoordSystem	**Description**	The coordinate system used when reading the actual acceleration
	Data type	SINT
	Range	One of the following enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2
	Unit	n/a
	Default	—
Acceleration[]	**Description**	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 "MC_CreateAxesGrp" (➜ p. 431) that is used to create axes groups.
	Data type	LREAL
	Range	n/a
	Unit	User units per second^2
	Default	—

2.3.82 Output

Valid	**Description**	If true, the accelerations have been read without error.
	Data type	BOOL
Error	**Description**	If true, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output was set to TRUE. See the table PLCopen Function Block ErrorID Output
	Data type	INT
PathAcceleration	**Description**	The current measured path acceleration of the group, measured by taking the square root of the sum of the squared accelerations of each axis.
	Data type	LREAL
	Unit	User units per second^2

Example

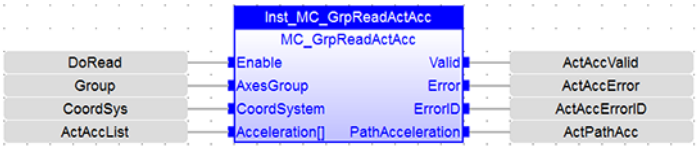
2.3.83 Structured Text

```
Inst_MC_GrpReadActAcc( DoRead, Group, CoordSys, AccList );
```

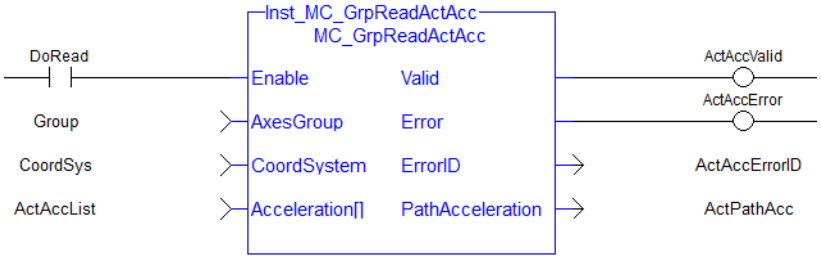
2.3.84 IL

```
BEGIN_IL
     CAL Inst_MC_GrpReadActAcc( DoRead, Group, CoordSys, AccList )
END_IL
```

2.3.85 FBD



2.3.86 Ladder Diagram



MC_GrpReadActPos

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.

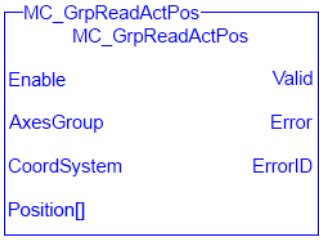
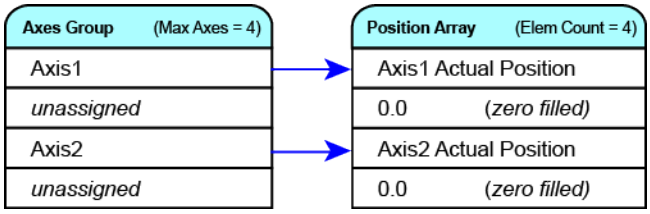


Figure 1-102: MC_GrpReadActPos

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:



2.3.87 Related Functions

"MC_GrpReadActVel" (➜ p. 456), "MC_GrpReadActAcc" (➜ p. 451), "MC_GrpReadCmdPos" (➜ p. 458), "MC_GrpReadCmdVel" (➜ p. 460)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

For more detail on how inputs and outputs work, refer to PLCopen Function Blocks - General Rules.

2.3.88 Input

Enable	**Description**	If True, then this function block will read the current actual position of the axes in the group
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axes group from which the actual position will be read
	Data type	AXES_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—

CoordSystem	**Description**	The coordinate system used when reading the actual position
	Data type	SINT
	Range	One of the following enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2
	Unit	n/a
	Default	—
Position[]	**Description**	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.
	Data type	LREAL
	Range	n/a
	Unit	User units
	Default	—

2.3.89 Output

Valid	**Description**	If true, the positions have been read without error
	Data type	BOOL
Error	**Description**	If true, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See the table PLCopen Function Block ErrorID Output
	Data type	INT

Example

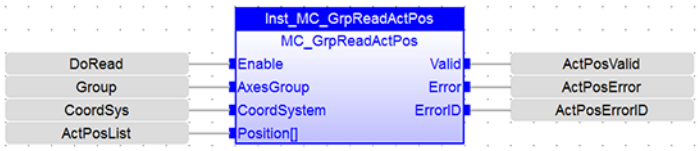
2.3.90 Structured Text

```
Inst_MC_GrpReadActPos( DoRead, Group, CoordSys, PosList );
```

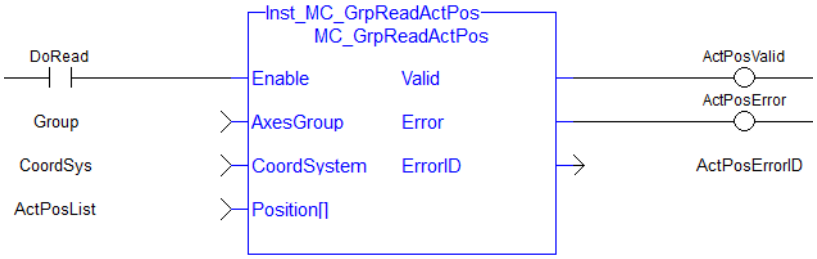
2.3.91 IL

```
BEGIN_IL
     CAL Inst_MC_GrpReadActPos( DoRead, Group, CoordSys, PosList )
END_IL
```

2.3.92 FBD



2.3.93 Ladder Diagram



MC_GrpReadActVel

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.

NOTE

- The actual velocity is smoothed over the last 10 samples. This reduces the error in velocity estimation, but introduces a small amount of phase delay in the reported velocities.
- Currently, only the ACS coordinate system is supported. See Coordinate Systems to learn more.

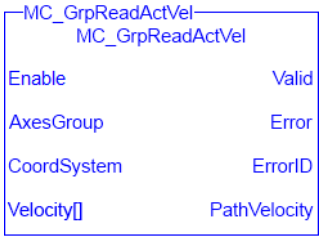
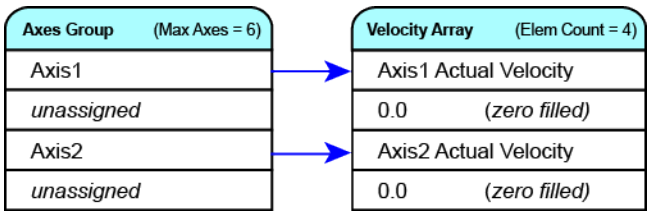


Figure 1-103: MC_GrpReadActVel

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:



2.3.94 Related Functions

"MC_GrpReadActPos" (➜ p. 453), "MC_GrpReadActAcc" (➜ p. 451), "MC_GrpReadCmdPos" (➜ p. 458), "MC_GrpReadCmdVel" (➜ p. 460)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

For more detail on how inputs and outputs work, refer to PLCopen Function Blocks - General Rules.

2.3.95 Input

Enable	**Description**	If True, then this function block will read the current actual velocity of the group and the axes in the group
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axes group from which the actual velocity will be read
	Data type	AXES_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—
CoordSystem	**Description**	The coordinate system used when reading the actual velocity
	Data type	SINT
	Range	One of the following enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2
	Unit	n/a
	Default	—
Velocity[]	**Description**	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 "MC_CreateAxesGrp" (➜ p. 431) that is used to create axes groups.
	Data type	LREAL
	Range	n/a
	Unit	User units per second
	Default	—

2.3.96 Output

Valid	**Description**	If true, the velocities have been read without error.
	Data type	BOOL
Error	**Description**	If true, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See the table PLCopen Function Block ErrorID Output
	Data type	INT
PathVelocity	**Description**	The current measured path velocity of the group, measured by taking the square root of the sum of the squared velocities of each axis.
	Data type	LREAL
	Unit	User units per second

Example

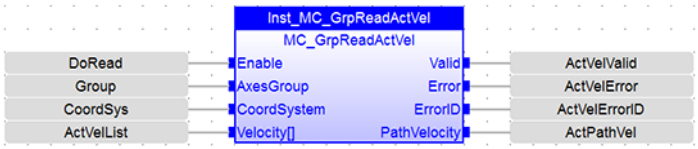
2.3.97 Structured Text

```
Inst_MC_GrpReadActVel(DoRead, Group, CoordSys, VelList);
```

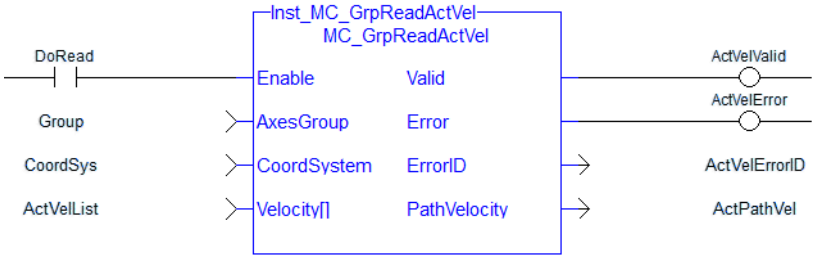
2.3.98 IL

```
BEGIN_IL
     CAL Inst_MC_GrpReadActVel(DoRead, Group, CoordsSys, VelList)
END_IL
```

2.3.99 FBD



2.3.100 FFLD



MC_GrpReadCmdPos

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.

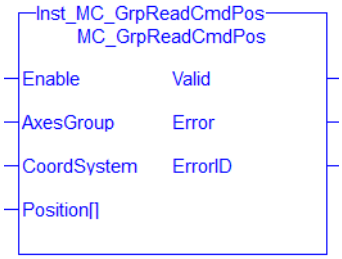
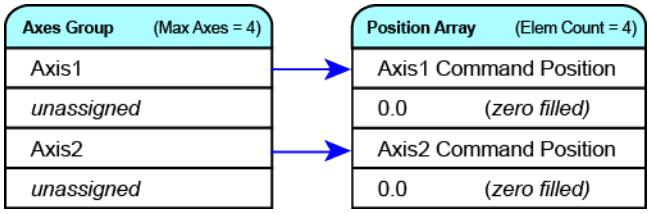


Figure 1-104: MC_GrpReadCmdPos

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:



2.3.101 Related Function Blocks

"MC_GrpReadActPos" (➜ p. 453), "MC_GrpReadActVel" (➜ p. 456), "MC_GrpReadActAcc" (➜ p. 451), "MC_GrpReadCmdVel" (➜ p. 460)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

For more detail on how inputs and outputs work, refer to PLCopen Function Blocks - General Rules.

2.3.102 Input

Enable	**Description**	If True, then this function block will read the current commanded position of the axes in the group
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axes group from which the commanded position will be read.
	Data type	AXES_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—
CoordSystem	**Description**	The coordinate system used when reading the commanded position.
	Data type	SINT
	Range	One of the following enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2
	Unit	n/a
	Default	—
Position[]	**Description**	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 "MC_CreateAxesGrp" (➜ p. 431), which is used to create axes groups.
	Data type	LREAL
	Range	n/a
	Unit	User units
	Default	—

2.3.103 Output

Valid	**Description**	If true, that the positions have been read without error.
	Data type	BOOL
Error	**Description**	If true, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See the table in PLCopen Function Block ErrorID Output
	Data type	INT

Example

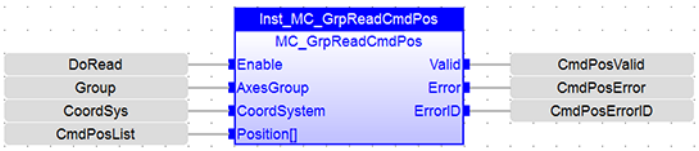
2.3.104 Structured Text

```
(*MC_GrpReadCmdPos ST example *)
Inst_MC_GrpReadCmdPos(DoRead, Group, CoordSys, PosList );
```

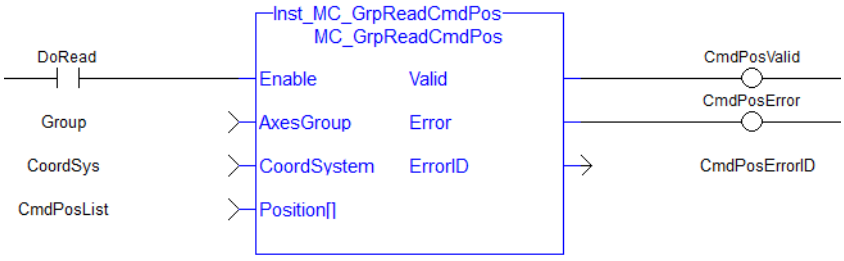
2.3.105 IL

```
BEGIN_IL
     CAL Inst_MC_GrpReadCmdPos( DoRead, Group, CoordSys, PosList )
END_IL
```

2.3.106 FBD



2.3.107 FFLD



MC_GrpReadCmdVel

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.

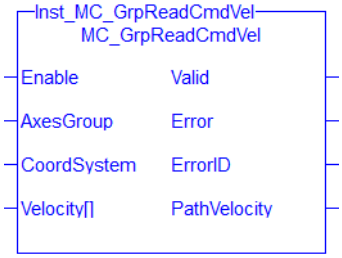
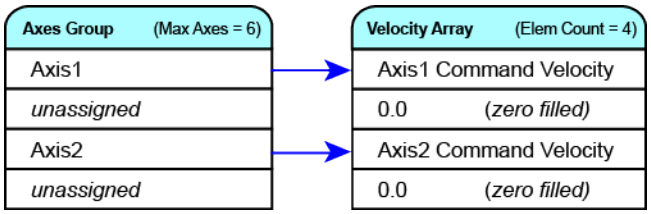


Figure 1-105: MC_GrpReadCmdVel

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:



2.3.108 Related Function Blocks

"MC_GrpReadActPos" (➜ p. 453), "MC_GrpReadActVel" (➜ p. 456), "MC_GrpReadActAcc" (➜ p. 451), "MC_GrpReadCmdPos" (➜ p. 458)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

For more detail on how inputs and outputs work, refer to PLCopen Function Blocks - General Rules.

2.3.109 Input

Enable	**Description**	If True, then this function block will read the current commanded velocity of the group and the axes in the group
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axes group from which the commanded velocity will be read.
	Data type	AXES_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—
CoordSystem	**Description**	The coordinate system used when reading the commanded velocity.
	Data type	SINT

	Range	One of the following enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2
	Unit	n/a
	Default	—
Velocity[]	**Description**	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 "MC_CreateAxesGrp" (➜ p. 431), which is used to create axes groups.
	Data type	LREAL
	Range	n/a
	Unit	User units per second
	Default	—

2.3.110 Output

Valid	**Description**	If true, that the velocities have been read without error.
	Data type	BOOL
Error	**Description**	If true, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See the table in PLCopen Function Block ErrorID Output
	Data type	INT
PathVelocity	**Description**	The current commanded path velocity of the group, measured by taking the square root of the sum of the squared velocities of each axis.
	Data type	LREAL
	Unit	User units per second

Example

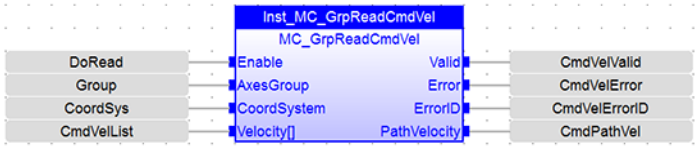
2.3.111 Structured Text

```
(*MC_GrpReadCmdVel ST example *)
Inst_MC_GrpReadCmdVel(DoRead, Group, CoordSys, VelList );
```

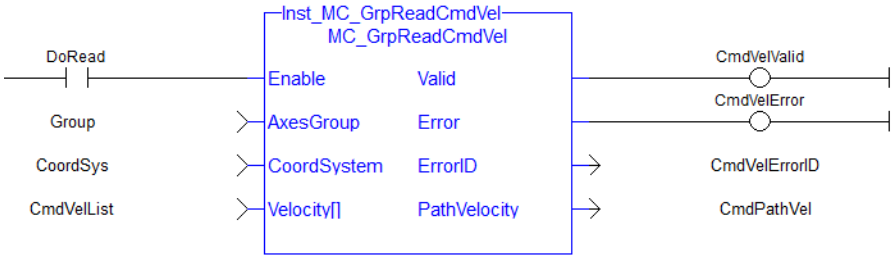
2.3.112 IL

```
BEGIN_IL
     CAL Inst_MC_GrpReadCmdVel( DoRead, Group, CoordSys, VelList )
END_IL
```

2.3.113 FBD



2.3.114 FFLD



MC_GrpReadError

Description

This function describes general axes group errors. This function does not cause any motion.

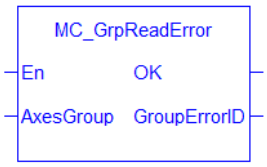


Figure 1-106: MC_GrpReadError

2.3.115 Related Functions

"MC_GrpReset" (➜ p. 439), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

2.3.116 Input

En	**Description**	Enables execution
	Data Type	BOOL
	Range	0,1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axes group from which the GroupErrorID will be read.
	Data Type	AXES_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—

2.3.117 Output

OK	**Description**	Indicates the function executed successfully
	Data Type	BOOL

GroupErrorID	**Description**	Displays the Error ID for the given Axis Group. See table in PLCopen Function Block ErrorID Output
	Data Type	INT

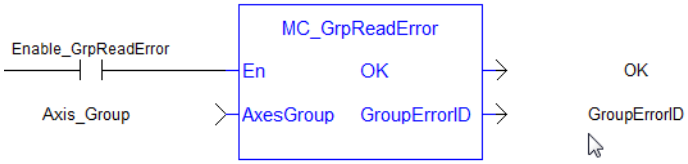
Examples

2.3.118 Structured Text

```
(* MC_GrpReadError example *)
 MC_GrpReadError(Axis_Group);
```

2.3.119 FBD

2.3.120 FFLD



MC_GrpReadStatus

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

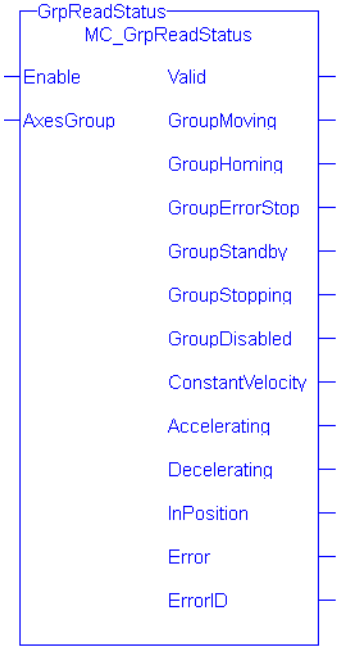


Figure 1-107: MC_GrpReadStatus

2.3.121 Related Functions

"MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

Input

Enable	**Description**	If True, then the axes group status will be read.
	Data type	BOOL
	Range	0. 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axis group from which the status will be read
	Data type	AXIS_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—

Output

Valid	**Description**	True if valid outputs are available
	Data type	BOOL
GroupMoving [1]	**Description**	The axes group is in the Moving state, indicating that the group is enabled and currently executing a coordinated motion command.
	Data type	BOOL
GroupHoming [1]	**Description**	Not supported
	Data type	BOOL
GroupErrorStop [1]	**Description**	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.
	Data type	BOOL
GroupStandby [1]	**Description**	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.
	Data type	BOOL
GroupStopping [1]	**Description**	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.
	Data type	BOOL

GroupDisabled [1]	**Description**	The axis group is in the Disabled state and cannot accept coordinated motion commands.
	Data type	BOOL
ConstantVelocity	**Description**	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.
	Data type	BOOL
Accelerating	**Description**	True if the commanded path velocity is accelerating between the current scan of the application program and the previous scan.
	Data type	BOOL
Decelerating	**Description**	True if the commanded path velocity is decelerating between the current scan of the application program and the previous scan.
	Data type	BOOL
InPosition	**Description**	True indicates that the axes group is “in position”. The following must be true for the axes group to be “in position”: • The axes group is enabled. • There are no moves in the group’s queue. • The servo loop is closed for each axis in the group. • There are no moves in the individual axis queue for each axis in the group. • The command delta is zero for each axis in the group. • The actual position is within the In-Position Bandwidth of the command position for each axis in the group.
	Data type	BOOL
Error	**Description**	If True, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error identifier if the Error output is set to TRUE. See the table in PLCopen Function Block ErrorID Output.
	Data type	BOOL

[1] 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.

Example

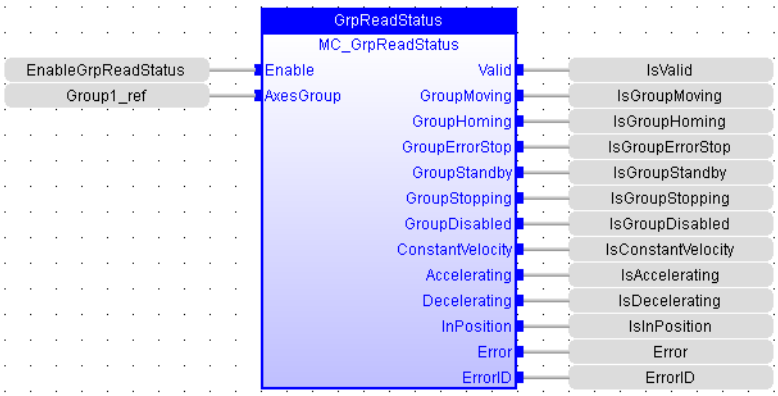
2.3.122 Structured Text

```
(*Inst MC_GrpReadStatusST example *)
Inst_MC_GrpReadStatus( EnableGrpReadStatus, Group1_Ref );
```

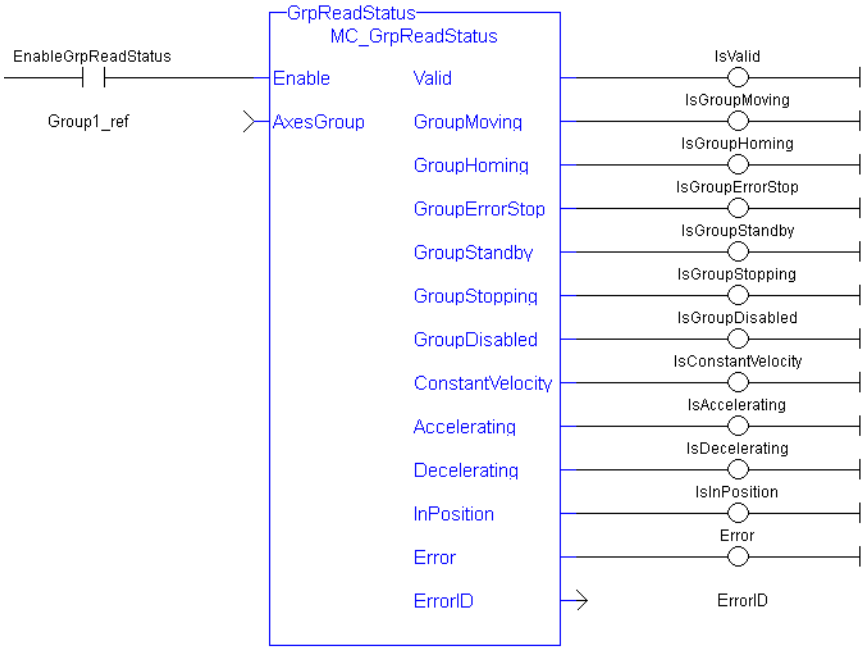
2.3.123 IL

```
BEGIN_IL
     CAL Inst_MC_GrpReadStatus( EnableGrpReadStatus, Group1_Ref )
END_IL
```

2.3.124 FBD



2.3.125 FFLD



2.3.125.1 Coordinated Motion Motion Library

Function	Description
"MC_AxisSetDefaults" (➜ p. 468)	Sets the default kinematic parameters for an axis.
"MC_GrpHalt" (➜ p. 470)	Performs a controlled motion stop of all the axes in the group
"MC_GrpSetOverride" (➜ p. 472)	Sets the velocity factor that is multiplied to the commanded velocity of all axes in the group.
"MC_MoveCircAbs" (➜ p. 474)	Commands interpolated circular movement on an axes group to the specified absolute positions.
"MC_MoveCircRel" (➜ p. 481)	Commands interpolated circular movement on an axes group to the specified relative positions.
"MC_MoveDirAbs" (➜ p. 486)	Commands movement of an axes group to an absolute position regardless of path.
"MC_MoveDirRel" (➜ p. 489)	Commands movement of an axes group to a relative position regardless of path.

Function	Description
"MC_MoveLinAbs" (➜ p. 492)	Commands interpolated linear movement on an axes group to the specified absolute positions.
"MC_MoveLinRel" (➜ p. 496)	Commands interpolated linear movement on an axes group to the specified relative positions.

MC_AxisSetDefaults

Description

MC_AxisSetDefaults sets the default kinematic variables for "MC_MoveDirAbs" (➜ p. 486) and "MC_MoveDirRel" (➜ p. 489). 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.

The function block returns an error if the group state is not GroupStandby or GroupDisabled.

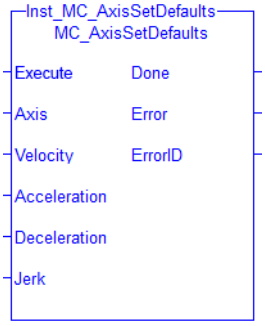


Figure 1-108: MC_AxisSetDefaults

2.3.126 Related Functions

"MC_MoveDirAbs" (➜ p. 486), "MC_MoveDirRel" (➜ p. 489), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

See also:

- Differences Between Functions and Function Blocks
- Calling a function

Arguments

2.3.127 Input

Execute	**Description**	On the rising edge, request to set the default kinematic parameters.
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Axis	**Description**	Reference to the axis which will have its default kinematic parameters set.
	Data type	AXIS_REF
	Range	—
	Unit	n/a

	Default	—
Velocity	**Description**	The default velocity.
	Data type	LREAL
	Range	0 < Velocity < (20 * Acceleration) and 0 < Velocity < (20 * Deceleration) See Limitations on Acceleration and Jerk for more information.
	Unit	User units per second
	Default	—
Acceleration	**Description**	Trapezoidal: Acceleration rate S-curve: Maximum acceleration see "Selection of Acceleration and Jerk Parameters for Function Blocks"
	Data type	LREAL
	Range	(Velocity / 20) < Acceleration < (2 * Jerk) See Limitations on Acceleration and Jerk for more information.
	Unit	User units per second^2
	Default	—
Deceleration	**Description**	Trapezoidal: Deceleration rate S-curve: Unused
	Data type	LREAL
	Range	(Velocity / 20) < Deceleration < (2 * Jerk)
	Unit	User unit per second^2
	Default	—
Jerk	**Description**	Trapezoidal: 0 S-curve: Constant jerk see "Selection of Acceleration and Jerk Parameters for Function Blocks"
	Data type	LREAL
	Range	(Velocity / 20) < Acceleration < (2 * Jerk) and (Velocity / 20) < Deceleration < (2 * Jerk)
	Unit	User units per second^3
	Default	—

2.3.128 Output

Done	**Description**	If True, then the command completed successfully.
	Data type	BOOL
Error	**Description**	If True, then an error has occurred
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See the table in PLCopen Function Block ErrorID Output.
	Data type	DINT

Example

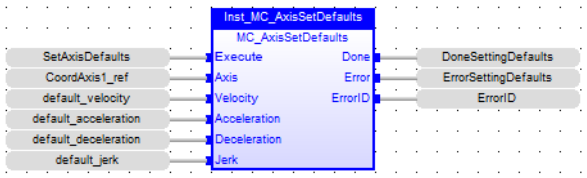
2.3.129 Structured Text

```
 (* 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);
```

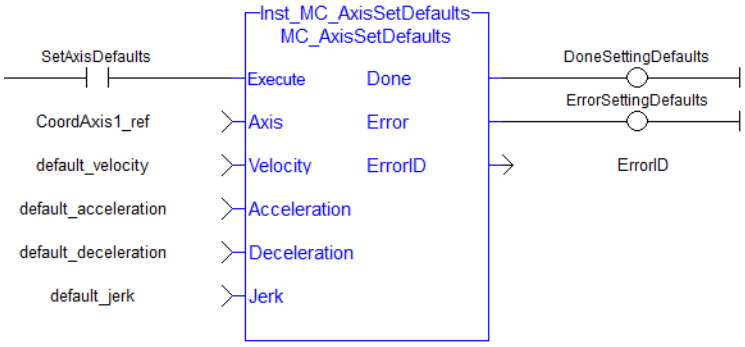
2.3.130 Instruction List

```
BEGIN_IL
     CAL Inst_MC_AxisSetDefaults( TRUE, CoordAxis1_Ref, default_velocity,
default_acceleration, default_deceleration, default_jerk)
END_IL
```

2.3.131 Function Block Diagram



2.3.132 Ladder Diagram



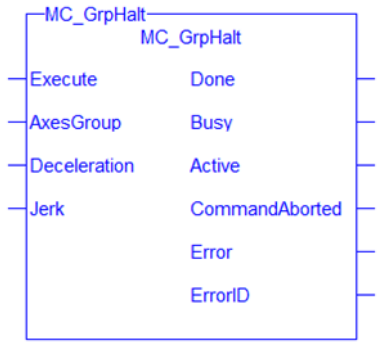
MC_GrpHalt

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.



2.3.133 Related Functions

"MC_GrpStop" (➜ p. 441), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

For more details on how inputs and outputs work, refer to PLCopen Function Blocks - General Rules.

2.3.134 Input

Execute	**Description**	On the rising edge the command to halt all of the axes in the group is initiated.
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axes group in which the axes will be stopped.
	Data type	AXES_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—
Deceleration	**Description**	The path deceleration rate for all of the axes in the group
	Data type	LREAL
	Range	0 < Deceleration See Limitations on Acceleration and Jerk for more information.
	Unit	n/a
	Default	—
Jerk	**Description**	Not supported
	Data type	LREAL
	Range	0 ≤ Jerk See Limitations on Acceleration and Jerk for more information.
	Unit	n/a
	Default	—

2.3.135 Output

Done	**Description**	If True, then the command completed successfully.
	Data type	BOOL
Busy	**Description**	TRUE from the moment the EXECUTE input is TRUE until the time the halt is completed.
	Data type	BOOL
Active	**Description**	Indicates that the halt is still executing.
	Data type	BOOL
CommandAborted	**Description**	If True, command was aborted by another FB.
	Data type	BOOL
Error	**Description**	IIf True, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error identifier if Error output is set to TRUE.. See table in PLCopen Function Block ErrorID Output.
	Data type	INT

Example

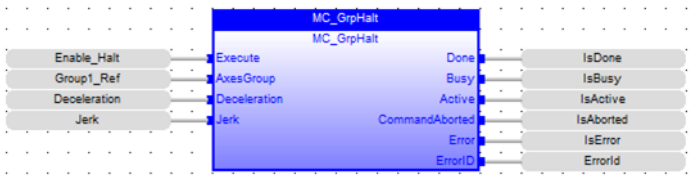
2.3.136 Structured Text

```
Inst_MC_GrpHalt ( EnableHalt, Group1_Ref, Deceleration, Jerk );
```

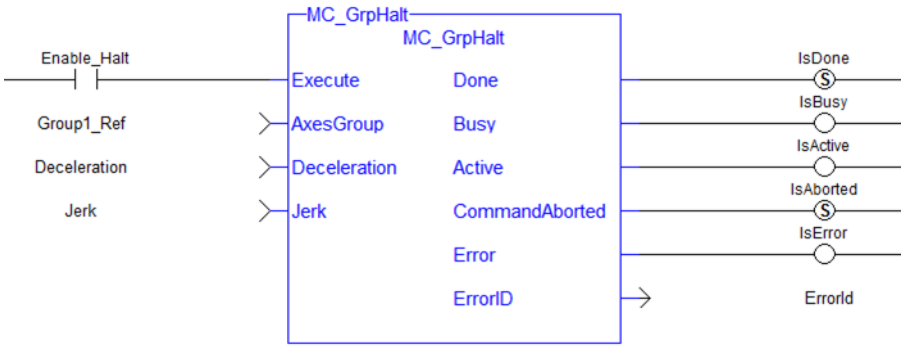
2.3.137 IL

```
BEGIN_IL
     CAL Inst_MC_GrpHalt ( EnableHalt, Group1_Ref, Deceleration, Jerk )
END_IL
```

2.3.138 FBD



2.3.139 FFLD



MC_GrpSetOverride

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.

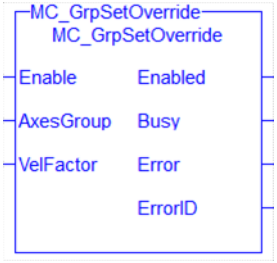


Figure 1-109: MC_GrpSetOverride

2.3.140 Related Functions

"MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

For more detail on how inputs and outputs work, refer to PLCopen Function Blocks - General Rules.

2.3.141 Input

Enable	**Description**	On the rising edge, changes the velocity multiplier for the axes group.
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axes group in which the velocity multiplier will be applied.
	Data type	AXES_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—
VelFactor	**Description**	The new multiplier factor for the commanded velocity of the axes group.
	Data type	REAL
	Range	[0.0 .. 2.0]
	Unit	n/a
	Default	—

2.3.142 Output

Enabled	**Description**	Indicates that the override was successful.
	Data type	BOOL
Busy	**Description**	If True, then the FB is executing.
	Data type	BOOL

Error	**Description**	If True, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error identifier if Error output is set to TRUE. See the table PLCopen Function Block ErrorID Output.
	Data type	INT

Example

2.3.143 ST

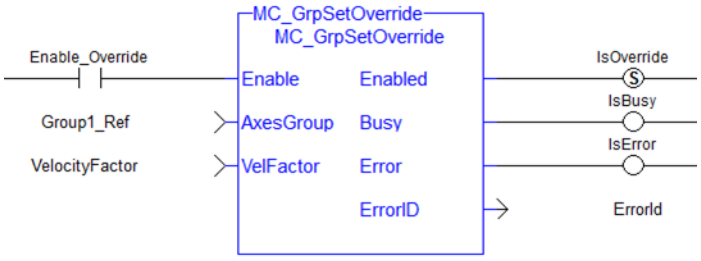
```
Inst_MC_GrpSetOverride( EnableOverride, Group1_Ref, VelocityFactor );
```

2.3.144 IL

```
BEGIN_IL
     CAL Inst_MC_GrpSetOverride( EnableOverride, Group1_Ref, Velo-
cityFactor )
END_IL
```

2.3.145 FBD

2.3.146 FFLD



MC_MoveCircAbs

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.

When all motion has completed successfully, the state of the axes group goes to GroupStandby.

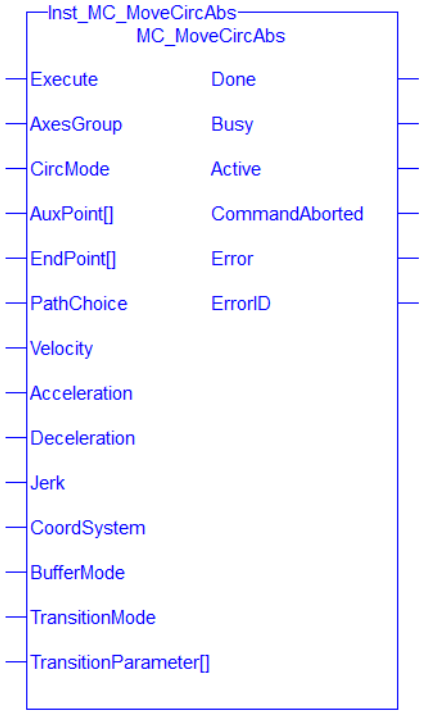


Figure 1-110: MC_MoveCircAbs

2.3.147 Related Functions

"MC_MoveCircRel" (➜ p. 481), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

2.3.148 Input

Execute	**Description**	On the rising edge request to perform a circular absolute move
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axis group that will perform the circular absolute move
	Data type	AXIS_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—
CircMode	**Description**	Specifies the meaning of the AuxPoint[ ] input.

	Data type	SINT One of the following enumeration values: • MC_CIRC_MODE_BORDER = 0 • MC_CIRC_MODE_CENTER = 1
	Range	n/a
	Unit	n/a
	Default	—
AuxPoint[]	**Description**	Array of absolute positions for each axis in the group. The meaning depends on the value of the CircMode input: • MC_CIRC_MODE_BORDER: AuxPoint defines a point on the circle which is crossed on the path from the starting to the end point. • MC_CIRC_MODE_CENTER: AuxPoint defines the center point of the circle.
	Data type	LREAL
	Range	[0, Number of axes in group - 1]
	Unit	n/a
	Required Precision	1 part in 100,000. See Precision Requirements for Circular Move Input Parameters.
	Default	—
EndPoint[]	**Description**	Array of absolute end positions for each axis in the group
	Data type	LREAL
	Range	[0, Number of axes in group - 1]
	Unit	n/a
	Default	—
	Required Precision	1 part in 100,000. See Precision Requirements for Circular Move Input Parameters.
PathChoice	**Description**	Specifies the direction of the path. This argument is only relevant when CircMode is MC_CIRC_MODE_CENTER.
	Data type	SINT One of the following enumeration values: • MC_CIRC_PATHCHOICE_CLOCKWISE = 0 = Clockwise • MC_CIRC_PATHCHOICE_COUNTERCLOCKWISE = 1 = Counterclockwise

	Range	n/a
	Unit	n/a
	Default	—
Velocity	**Description**	Maximum velocity of the defined path
	Data type	LREAL
	Range	0 < Velocity < (20 * Acceleration) and 0 < Velocity < (20 * Deceleration) See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second
	Default	—
Acceleration	**Description**	Maximum acceleration
	Data type	LREAL
	Range	0 < Velocity < (20 * Acceleration) See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second^2
	Default	—
Deceleration	**Description**	Maximum Deceleration
	Data type	LREAL
	Range	0 < Velocity < (20 * Deceleration) See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second^2
	Default	—
Jerk	**Description**	Maximum jerk
	Data type	LREAL
	Range	(Velocity / 20) < Acceleration < (2 * Jerk) and (Velocity / 20) < Deceleration < (2 * Jerk) See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second^3
	Default	—
CoordSystem	**Description**	The coordinate system used when commanding the circular absolute move. Currently, only the ACS coordinate system is supported. See Coordinate Systems to learn more.
	Data type	SINT

	Range	One of the following enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2
	Unit	n/a
	Default	—
BufferMode	**Description**	Defines the chronological sequence of the function block relative to the previous block. The blending modes (2, 3, 4, & 5) match the **path velocity** at the active move's endpoint. Some individual **axis velocities** 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 `TransitionMode` input to avoid this. See the table in Buffer Modes.
	Data type	SINT One of the following enumeration values: • MC_BUFFER_MODE_BUFFERED = 1 = Buffered • MC_BUFFER_MODE_BLENDING_PREVIOUS = 2 = Blending Previous • MC_BUFFER_MODE_BLENDING_NEXT = 3 = Blending Next • MC_BUFFER_MODE_BLENDING_LOW = 4 = Blending Low • MC_BUFFER_MODE_BLENDING_HIGH = 5 = Blending High *BufferMode = Abort = 0 is not allowed with this function block*.
	Range	—
	Unit	n/a
	Default	—

TransitionMode	**Description**	Coupled with the TransitionParameter[ ], this input defines the shape and dynamics of the inserted contour to connect the current motion with the next motion in the queue. See Transition Between Moves for additional information.
	Data type	SINT
	Range	The value is limited to the following: • MC_TRANSITION_MODE_NONE = 0 • MC_TRANSITION_MODE_CORNER_DISTANCE = 3
	Unit	n/a
	Default	—
TransitionParameter[ ]	**Description**	This array is dependent on the TransitionMode specified. The transition parameter values are applied to the axis group. See table: "**Transition Mode Parameters**" for details.
	Data type	LREAL
	Range	[1, N] N values are supplier specified dependent on the TransitionMode selected.
	Unit	n/a
	Default	—

2.3.149 Output

Done	**Description**	If True, then the command completed successfully.
	Data type	BOOL
Busy	**Description**	If True, then the function block is executing.
	Data type	BOOL
Active	**Description**	If True, then the function block is controlling motion.
	Data type	BOOL
CommandAborted	**Description**	If True, command was aborted by another function block.
	Data type	BOOL
Error	**Description**	If True, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See table in PLCopen Function Block ErrorID Output
	Data type	INT

Example

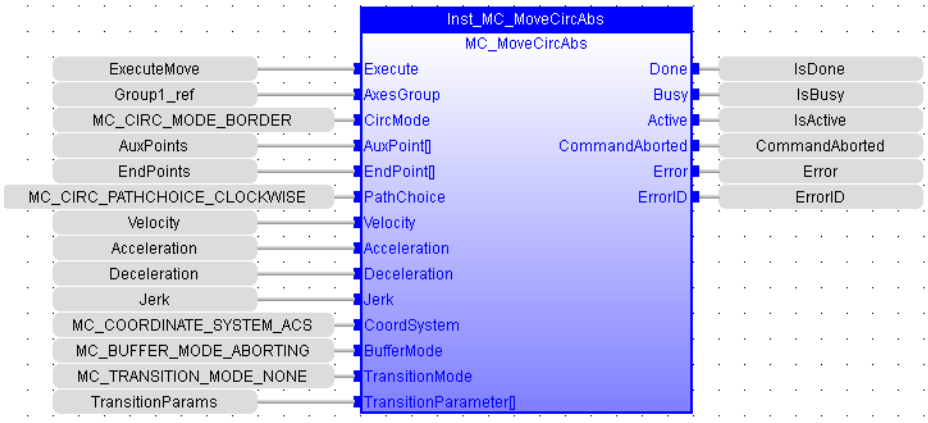
2.3.150 ST

```
Inst_MC_MoveCircAbs( ExecuteMove, Group1_Ref, MC_CIRC_MODE_BORDER,
AuxPoints, EndPoints, MC_CIRC_PATHCHOICE_CLOCKWISE, Velocity, Accel-
eration, Deceleration, Jerk, MC_COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_
ABORTING, MC_TRANSITION_MODE_NONE, TransitionParams );
```

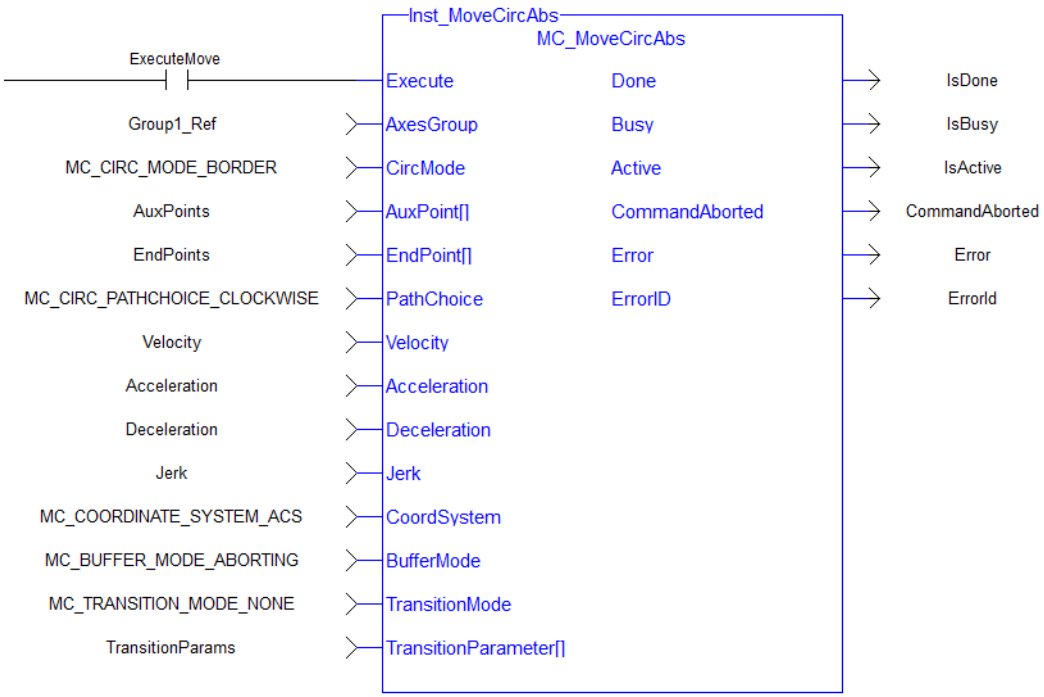
2.3.151 IL

```
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
```

2.3.152 FBD



2.3.153 FFLD



MC_MoveCircRel

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.

When all motion has completed successfully, the state of the axes group goes to GroupStandby.

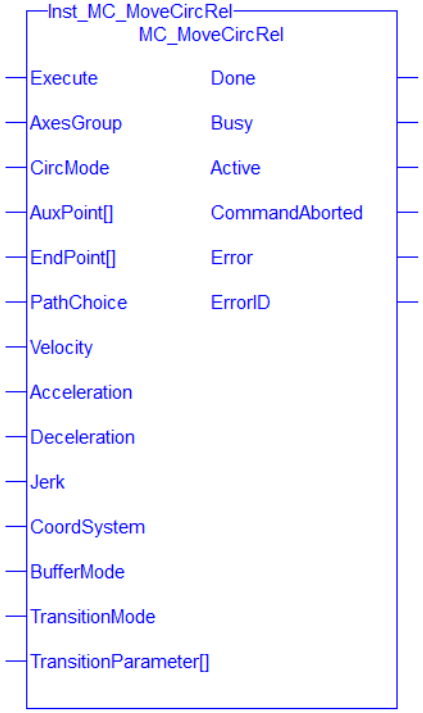


Figure 1-111: MC_MoveCircRel

2.3.154 Related Functions

"MC_MoveCircAbs" (➜ p. 474), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

2.3.155 Input

Execute

Description	On the rising edge request to perform a circular relative move.
Data type	BOOL
Range	0, 1
Unit	n/a

	Default	—
AxesGroup	**Description**	The axis group that will perform the circular relative move
	Data type	AXIS_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—
CircMode	**Description**	Specifies the meaning of the AuxPoint[ ] input (see AuxPoint[ ] below).
	Data type	SINT One of the following enumeration values: • MC_CIRC_MODE_BORDER = 0 • MC_CIRC_MODE_CENTER = 1
	Range	n/a
	Unit	n/a
	Default	—
AuxPoint[]	**Description**	Array of relative positions for each axis in the group. The meaning depends on the value of the CircMode input: • MC_CIRC_MODE_BORDER: AuxPoint defines a point on the circle which is crossed on the path from the starting to the end point. • MC_CIRC_MODE_CENTER: AuxPoint defines the center point of the circle. In all cases the points are relative to the starting point.
	Data type	LREAL
	Range	[0, Number of axes in group - 1]
	Unit	n/a
	Default	—
EndPoint[]	**Description**	Array of relative end positions for each axis in the group.
	Data type	LREAL
	Range	[0, Number of axes in group - 1]
	Unit	n/a
	Default	—
PathChoice	**Description**	Specifies the direction of the path. This argument is only relevant when CircMode is MC_CIRC_MODE_CENTER.

	Data type	SINT
		One of the following enumeration values: • MC_CIRC_PATHCHOICE_ CLOCKWISE = 0 = Clockwise • MC_CIRC_PATHCHOICE_ COUNTERCLOCKWISE = 1 = Counterclockwise
	Range	n/a
	Unit	n/a
	Default	—
Velocity	**Description**	Maximum velocity of the defined path
	Data type	LREAL
	Range	0 < Velocity < (20 * Acceleration) and 0 < Velocity < (20 * Deceleration) See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second
	Default	—
Acceleration	**Description**	Maximum acceleration
	Data type	LREAL
	Range	0 < Velocity < (20 * Acceleration) See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second^2
	Default	—
Deceleration	**Description**	Maximum Deceleration
	Data type	LREAL
	Range	0 < Velocity < (20 * Deceleration) See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second^2
	Default	—
Jerk	**Description**	Maximum jerk
	Data type	LREAL
	Range	(Velocity / 20) < Acceleration < (2 * Jerk) and (Velocity / 20) < Deceleration < (2 * Jerk) See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second^3
	Default	—
CoordSystem	**Description**	The coordinate system used when commanding the circular relative move Currently, only the ACS coordinate system is supported. See Coordinate Systems to learn more.
	Data type	SINT

	Range	One of the following enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2
	Unit	n/a
	Default	—
BufferMode	**Description**	Defines the chronological sequence of the function block relative to the previous block. The blending modes (2, 3, 4, & 5) match the **path velocity** at the active move's endpoint. Some individual **axis velocities** 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 `TransitionMode` input to avoid this. See the table in Buffer Modes.
	Data type	SINT One of the following enumeration values: • MC_BUFFER_MODE_ABORTING = 0 = Abort • MC_BUFFER_MODE_BUFFERED = 1 = Buffered • MC_BUFFER_MODE_BLENDING_PREVIOUS = 2 = Blending Previous • MC_BUFFER_MODE_BLENDING_NEXT = 3 = Blending Next • MC_BUFFER_MODE_BLENDING_LOW = 4 = Blending Low • MC_BUFFER_MODE_BLENDING_HIGH = 5 = Blending High
	Range	—
	Unit	n/a
	Default	—
TransitionMode	**Description**	Coupled with the TransitionParameter[ ], this input defines the shape and dynamics of the inserted contour to connect the current motion with the next motion in the queue. See Transition Between Moves for additional information.
	Data type	SINT
	Range	The value is limited to the following: • MC_TRANSITION_MODE_NONE = 0 • MC_TRANSITION_MODE_CORNER_DISTANCE = 3
	Unit	n/a
	Default	—
TransitionParameter[ ]	**Description**	This array is dependent on the TransitionMode specified. The transition parameter values are applied to the axis group. See table: "**Transition Mode Parameters**" for details.

Data type	LREAL
Range	[1, N] N values are supplier specified dependent on the TransitionMode selected.
Unit	n/a
Default	—

2.3.156 Output

Done	**Description**	If True, then the command completed successfully.
	Data type	BOOL
Busy	**Description**	If True, then the function block is executing.
	Data type	BOOL
Active	**Description**	If True, then the function block is controlling motion.
	Data type	BOOL
CommandAborted	**Description**	If True, command was aborted by another function block.
	Data type	BOOL
Error	**Description**	If True, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See table in PLCopen Function Block ErrorID Output
	Data type	INT

Example

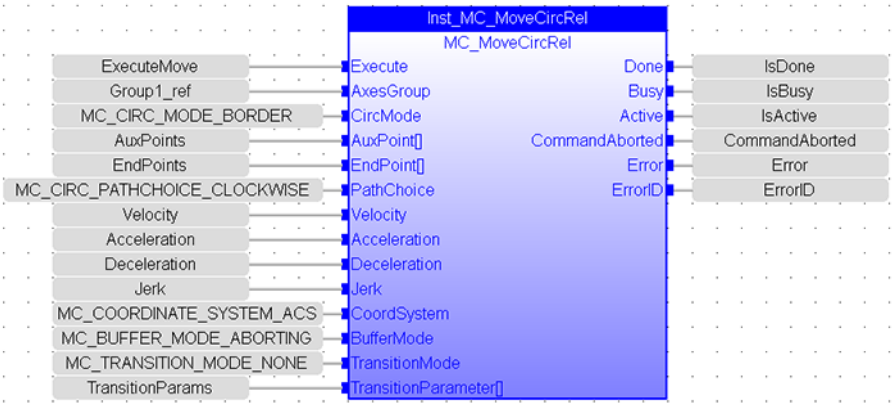
2.3.157 ST

```
Inst_MC_MoveCircRel( ExecuteMove, Group1_Ref, MC_CIRC_MODE_BORDER,
AuxPoints, EndPoints, MC_CIRC_PATHCHOICE_CLOCKWISE, Velocity, Accel-
eration, Deceleration, Jerk, MC_COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_
ABORTING, MC_TRANSITION_MODE_NONE, TransitionParams );
```

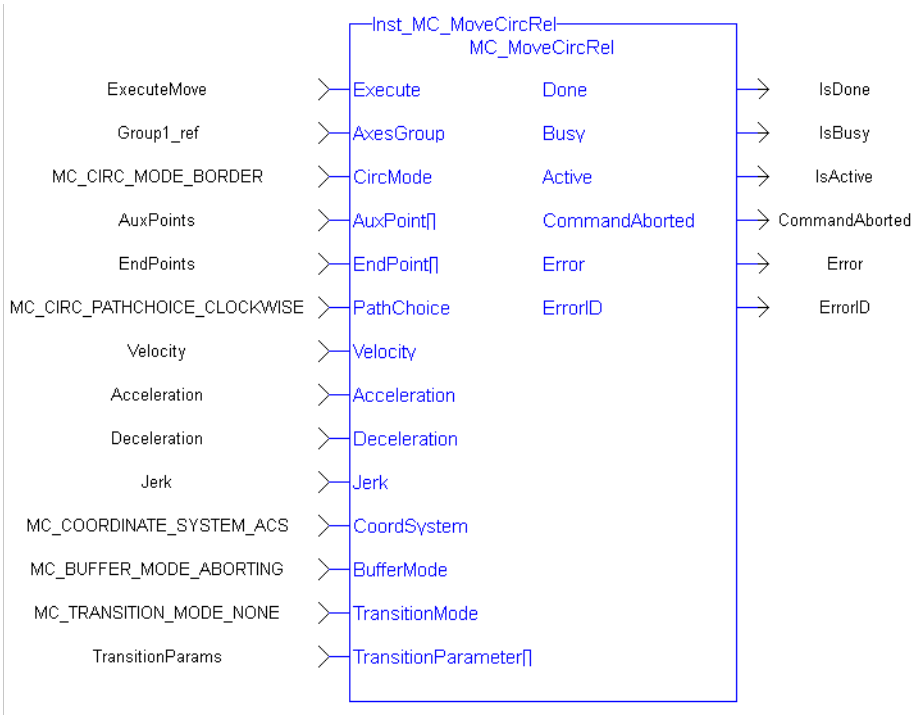
2.3.158 IL

```
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
```

2.3.159 FBD



2.3.160 FFLD



MC_MoveDirAbs

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. 468).

When all motion is completed successfully, the state becomes GroupStandby.

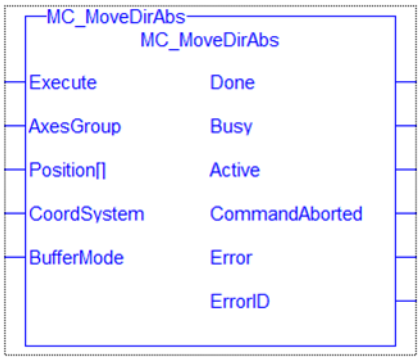


Figure 1-112: MC_MoveDirAbs

2.3.161 Related Functions

"MC_MoveDirRel" (➜ p. 489), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

2.3.162 Input

Execute	**Description**	On the rising edge, request to perform a direct absolute move.
	Data Type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	Reference to an axes group
	Data Type	AXES_GROUP_REF
	Range	—
	Unit	n/a
	Default	—
Position[ ]	**Description**	Array of absolute end positions for each axis in the group.
	Data Type	LREAL
	Range	[0, Number of axes in group - 1]
	Unit	n/a
	Default	—
CoordSystem	**Description**	The coordinate system used when commanding the direct absolute move Currently, only the ACS coordinate system is supported. See Coordinate Systems to learn more.
	Data Type	SINT
	Range	• MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2
	Unit	n/a
	Default	—

BufferMode	**Description**	Defines the chronological sequence of the function block relative to the previous block. See the table in Buffer Modes.
	Data Type	SINT MC_BUFFER_MODE_ABORTING = 0 = Abort MC_BUFFER_MODE_BUFFERED = 1 = Buffered
	Range	—
	Unit	n/a
	Default	—

2.3.163 Output

Done	**Description**	If True, then the command completed successfully.
	Data type	BOOL
Busy	**Description**	If True, then the function block is executing.
	Data type	BOOL
Active	**Description**	If True, then the function block is controlling motion.
	Data type	BOOL
CommandAborted	**Description**	If True, command was aborted by another function block.
	Data type	BOOL
Error	**Description**	If True, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See table in PLCopen Function Block ErrorID Output.
	Data type	INT

Example

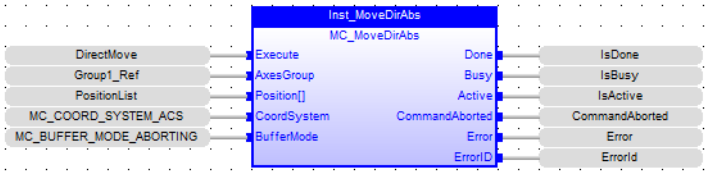
2.3.164 Structure Text

```
Inst_MC_MoveDirAbs( DirectMove, Group1_Ref, PositionList, MC_COORDSYSTEM_
ACS, MC_BUFFER_MODE_ABORTING);
```

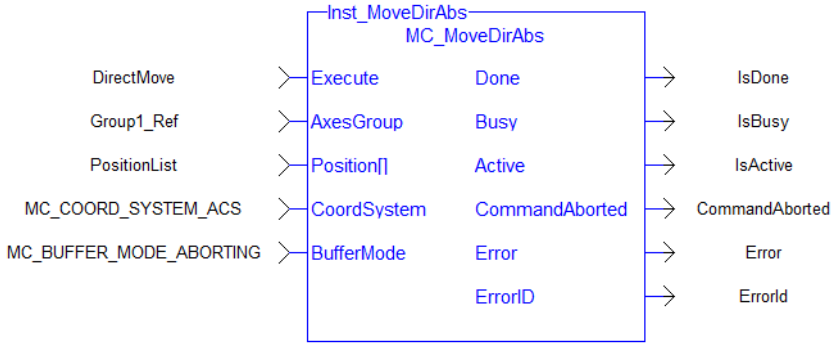
2.3.165 IL

```
BEGIN_IL
     CAL Inst_MC_MoveDirAbs( DirectMove, Group1_Ref, PositionList, MC_
COORD_SYSTEM_ACS, MC_BUFFER_MODE_ABORTING)
END_IL
```

2.3.166 Function Block Diagram



2.3.167 Ladder Diagram



MC_MoveDirRel

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. 468).

When all motion has completed successfully, the state of the axes group goes to GroupStandby.

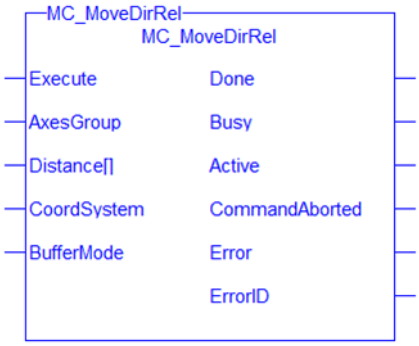


Figure 1-113: MC_MoveDirRel

2.3.168 Related Functions

"MC_MoveDirAbs" (➜ p. 486), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

2.3.169 Input

Execute	**Description**	On the rising edge request to perform a direct relative move.
	Data type	BOOL
	Range	0, 1

	Unit	n/a
	Default	—
AxesGroup	**Description**	Reference to an axes group
	Data type	AXES_GROUP_REF
	Range	—
	Unit	n/a
	Default	—
Distance[ ]	**Description**	An array containing the distance for each axis in the group.
	Data type	LREAL
	Range	[0, Number of axes in group - 1]
	Unit	n/a
	Default	—
CoordSystem	**Description**	The coordinate system used when commanding the direct relative move. Currently, only the ACS coordinate system is supported. See Coordinate Systems to learn more.
	Data type	SINT
	Range	• MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2
	Unit	n/a
	Default	—
BufferMode	**Description**	Defines the chronological sequence of the function block relative to the previous block. See the table in Buffer Modes

Data type	SINT MC_BUFFER_MODE_ ABORTING = 0 = Abort MC_BUFFER_MODE_ BUFFERED = 1 = Buffered
Range	—
Unit	n/a
Default	—

2.3.170 Output

Done	**Description**	If True, then the command completed successfully.
	Data type	BOOL
Busy	**Description**	If True, then the function block is executing. .
	Data type	BOOL
Active	**Description**	If True, then the function block is controlling motion.
	Data type	BOOL
CommandAborted	**Description**	If True, command was aborted by another function block.
	Data type	BOOL
Error	**Description**	If True, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See table in PLCopen Function Block ErrorID Output.
	Data type	INT

Example

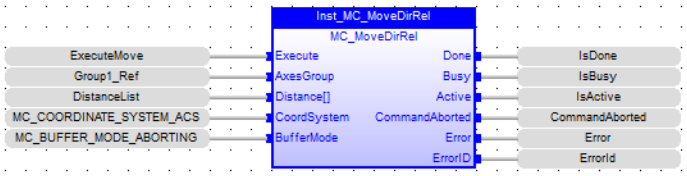
2.3.171 Structure Text

```
Inst_MC_MoveDirRel( ExecuteMove, Group1_Ref, DistanceList, MC_COORDINATE_
SYSTEM_ACS, MC_BUFFER_MODE_ABORTING );
```

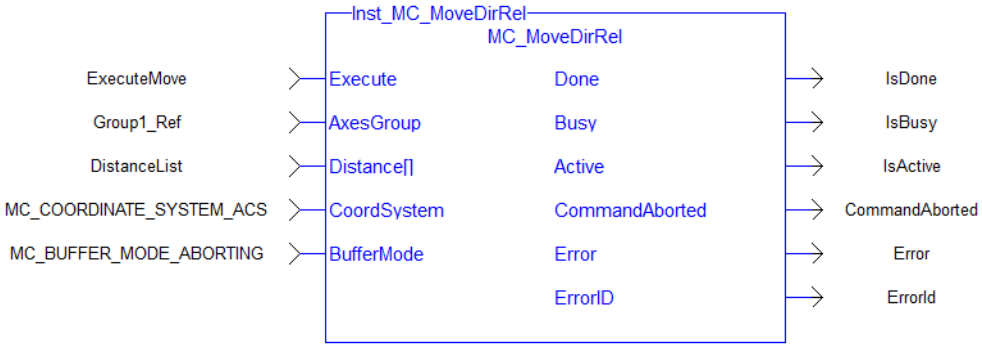
2.3.172 IL

```
BEGIN_IL
     CAL Inst_MC_MoveDirRel( ExecuteMove, Group1_Ref, DistanceList, MC_
COORDINATE_SYSTEM_ACS, MC_BUFFER_MODE_ABORTING )
END_IL
```

2.3.173 Function Block Diagram



2.3.174 Ladder Diagram



MC_MoveLinAbs

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.

When all motion has completed successfully, the state of the axes group goes to GroupStandby.

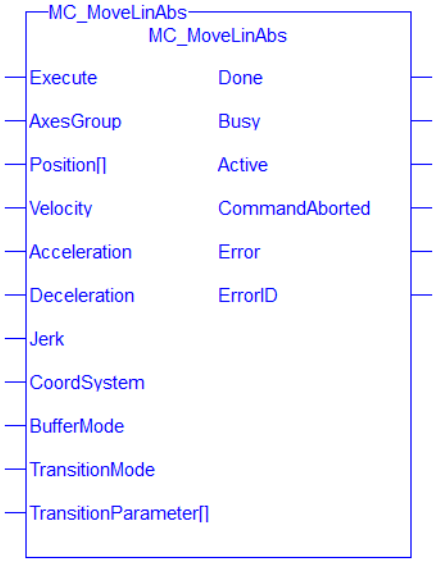


Figure 1-114: MC_MoveLinAbs

2.3.175 Related Functions

"MC_MoveLinRel" (➜ p. 496), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

2.3.176 Input

Execute	**Description**	On the rising edge request to perform a linear absolute move.
	Data Type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axis group that will perform the linear absolute move
	Data Type	AXIS_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—
Position[]	**Description**	Array of absolute end positions for each axis in the group.
	Data Type	LREAL
	Range	n/a
	Unit	user units
	Default	—
Velocity	**Description**	Maximum velocity of the defined path
	Data Type	LREAL
	Range	0 < Velocity < (20 * Acceleration) and 0 < Velocity < (20 * Deceleration) See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second
	Default	—
Acceleration	**Description**	Maximum acceleration
	Data Type	LREAL
	Range	0 < Velocity < (20 * Acceleration) See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second^2
	Default	—
Deceleration	**Description**	Maximum deceleration
	Data Type	LREAL
	Range	0 < Velocity < (20 * Deceleration) See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second^2
	Default	—
Jerk	**Description**	Maximum jerk

	Data Type	LREAL
	Range	(Velocity / 20) < Acceleration < (2 * Jerk) and (Velocity / 20) < Deceleration < (2 * Jerk) See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second^3
	Default	—
CoordSystem	**Description**	The coordinate system used when commanding the linear absolute move. Currently, only the ACS coordinate system is supported. See Coordinate Systems to learn more.
	Data Type	SINT
	Range	One of the following enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2
	Unit	n/a
	Default	—
BufferMode	**Description**	Defines the chronological sequence of the function block relative to the previous block. MC_BUFFER_MODE_ABORTING = 0 = Abort MC_BUFFER_MODE_BUFFERED = 1 = Buffered MC_BUFFER_MODE_BLENDING_PREVIOUS = 2 = Blending Previous MC_BUFFER_MODE_BLENDING_NEXT = 3 = Blending Next MC_BUFFER_MODE_BLENDING_LOW = 4 = Blending Low MC_BUFFER_MODE_BLENDING_HIGH = 5 = Blending High The buffer mode is limited to MC_BUFFER_MODE_BUFFERED for groups with more than two axes. The blending modes (2, 3, 4, & 5) match the **path velocity** at the active move's endpoint. Some individual **axis velocities** 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 `TransitionMode` input to avoid this. See the table in Buffer Modes
	Data Type	SINT
	Range	—
	Unit	n/a
	Default	—

TransitionMode	**Description**	Coupled with the TransitionParameter[ ], this input defines the shape and dynamics of the inserted contour to connect the current motion with the next motion in the queue. See Transition Between Moves for additional information.
	Data type	SINT
	Range	The value is limited to the following: • MC_TRANSITION_MODE_NONE = 0 • MC_TRANSITION_MODE_CORNER_DISTANCE = 3 The transition mode is limited to MC_TRANSITION_MODE_NONE for groups with more than two axes.
	Unit	n/a
	Default	—
TransitionParameter[ ]	**Description**	This array is dependent on the TransitionMode specified. The transition parameter values are applied to the axis group. See table: "**Transition Mode Parameters**" for details.
	Data Type	LREAL
	Range	[1, N] N values are supplier specified dependent on the TranstitionMode selected.
	Unit	n/a
	Default	—

2.3.177 Output

Done	**Description**	If True, then the command completed successfully.
	Data type	BOOL
Busy	**Description**	If True, then the function block is executing.
	Data type	BOOL
Active	**Description**	If True, then the function block is controlling motion.
	Data type	BOOL
CommandAborted	**Description**	If True, then the command was aborted by another function block.
	Data type	BOOL
Error	**Description**	If True, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See table in PLCopen Function Block ErrorID Output
	Data type	INT

Example

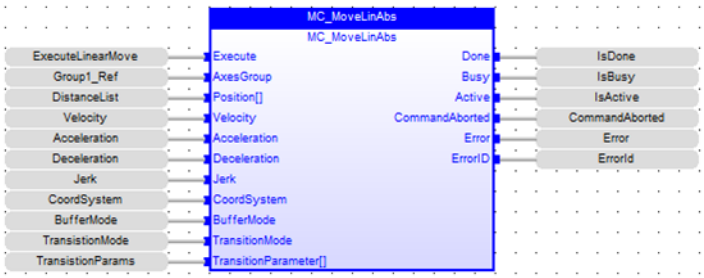
2.3.178 Structured Text

```
(* Inst_MC_MoveLinAbsST example *)
Inst_MC_MoveLinAbs( ExecuteLinearMove, Group1_Ref, PositionList, Velo-
city, Acceleration, Deceleration, Jerk, CoordSystem, BufferMode, Trans-
itionMode, TransitionParams );
```

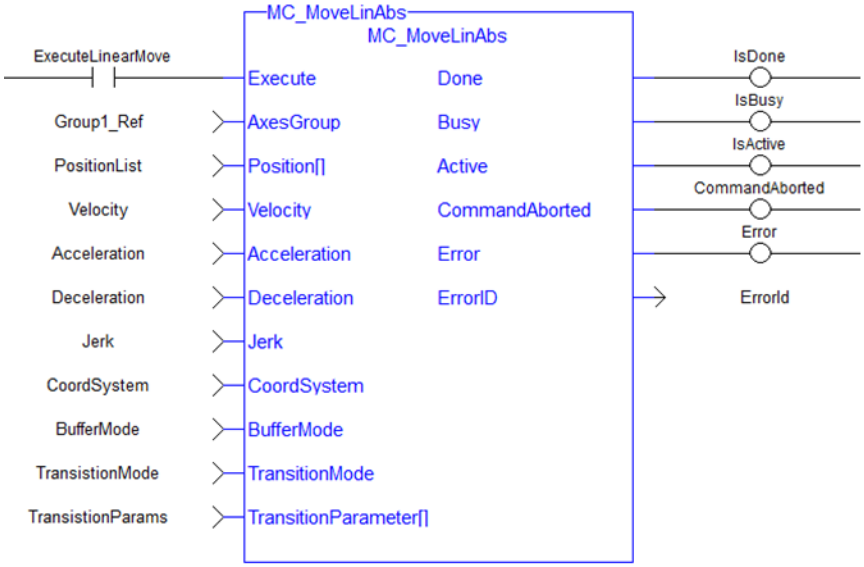
2.3.179 IL

```
BEGIN_IL
     CAL Inst_MC_MoveLinAbs( ExecuteLinearMove, Group1_Ref, PositionList,
Velocity, Acceleration, Deceleration, Jerk, CoordSystem, BufferMode,
TransitionMode, TransitionParams )
END_IL
```

2.3.180 FBD



2.3.181 FFLD



MC_MoveLinRel

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.

When all motion has completed successfully, the state of the axes group goes to GroupStandby.

See Transition Between Moves for additional information.

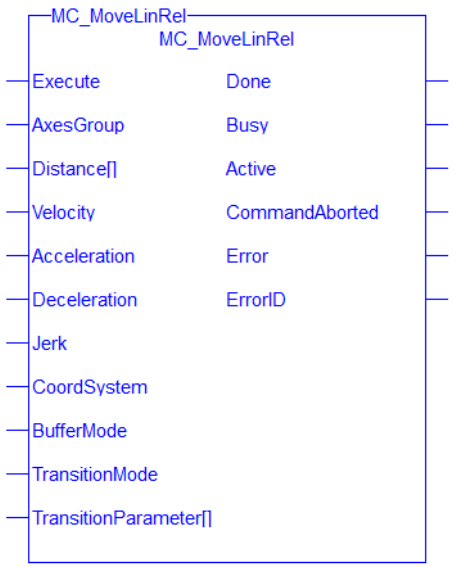


Figure 1-115: MC_MoveLinRel

2.3.182 Related Functions

"MC_MoveLinAbs" (➜ p. 492), "MC_ErrorDescription" (➜ p. 411)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

2.3.183 Input

Execute	**Description**	On the rising edge request to perform a linear relative move
	Data Type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axis group that will perform the linear relative move
	Data Type	AXIS_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—
Distance[]	**Description**	Array of distances for each axis in the group.
	Data Type	LREAL
	Range	n/a
	Unit	user units
	Default	—
Velocity	**Description**	Maximum velocity of the defined path

	Data Type	LREAL
	Range	0 < Velocity < (20 * Acceleration) and 0 < Velocity < (20 * Deceleration)
		See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second
	Default	—
Acceleration	**Description**	Maximum acceleration
	Data Type	LREAL
	Range	0 < Velocity < (20 * Acceleration)
		See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second^2
	Default	—
Deceleration	**Description**	Maximum deceleration
	Data Type	LREAL
	Range	0 < Velocity < (20 * Deceleration)
		See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second^2
	Default	—
Jerk	**Description**	Maximum jerk
	Data Type	LREAL
	Range	(Velocity / 20) < Acceleration < (2 * Jerk) and (Velocity / 20) < Deceleration < (2 * Jerk)
		See Limitations on Acceleration and Jerk for more information.
	Unit	user units per second^3
	Default	—
CoordSystem	**Description**	The coordinate system used when commanding the linear relative move
		Currently, only the ACS coordinate system is supported. See Coordinate Systems to learn more.
	Data Type	SINT
	Range	One of the following enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2
	Unit	n/a
	Default	—

BufferMode	**Description**	Defines the chronological sequence of the function block relative to the previous block. MC_BUFFER_MODE_ABORTING = 0 = Abort MC_BUFFER_MODE_BUFFERED = 1 = Buffered MC_BUFFER_MODE_BLENDING_PREVIOUS = 2 = Blending Previous MC_BUFFER_MODE_BLENDING_NEXT = 3 = Blending Next MC_BUFFER_MODE_BLENDING_LOW = 4 = Blending Low MC_BUFFER_MODE_BLENDING_HIGH = 5 = Blending High The buffer mode is limited to MC_BUFFER_MODE_BUFFERED for groups with more than two axes. The blending modes (2, 3, 4, & 5) match the **path velocity** at the active move's endpoint. Some individual **axis velocities** 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 `TransitionMode` input to avoid this. See the table in Buffer Modes
	Data Type	SINT
	Range	—
	Unit	n/a
	Default	—
TransitionMode	**Description**	Coupled with the TransitionParameter[ ], this input defines the shape and dynamics of the inserted contour to connect the current motion with the next motion in the queue. See Transition Between Moves for additional information.
	Data Type	SINT
	Range	The value is limited to the following: • MC_TRANSITION_MODE_NONE = 0 • MC_TRANSITION_MODE_CORNER_DISTANCE = 3 The transition mode is limited to MC_TRANSITION_MODE_NONE for groups with more than two axes.
	Unit	n/a
	Default	—
TransitionParameter[ ]	**Description**	This array is dependent on the TransitionMode specified. The transition parameter values are applied to the axis group. See table: "**Transition Mode Parameters**" for details.
	Data Type	LREAL

Range	[0, N] The value of N is dependent on the TransitionMode specified.
Unit	n/a
Default	—

2.3.184 Output

Done	**Description**	If True, then the command completed successfully.
	Data type	BOOL
Busy	**Description**	If True, then the function block is executing.
	Data type	BOOL
Active	**Description**	If True, then the function block is still controlling motion.
	Data type	BOOL
CommandAborted	**Description**	If True, then the command was aborted by another function block.
	Data type	BOOL
Error	**Description**	If True, then an error has occured
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE. See table in PLCopen Function Block ErrorID Output
	Data type	INT

Example

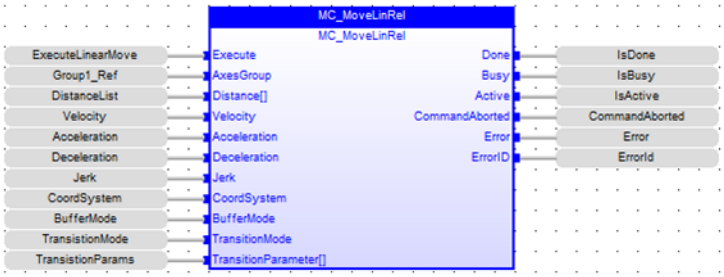
2.3.185 Structured Text

```
(* Inst_MC_MoveLinRelST example *)

Inst_MC_MoveLinRel( ExecuteLinearMove, Group1_Ref, DistanceList, Velo-
city, Acceleration, Deceleration, Jerk, CoordSystem, BufferMode, Tran-
stionMode, TransitionParams );
```

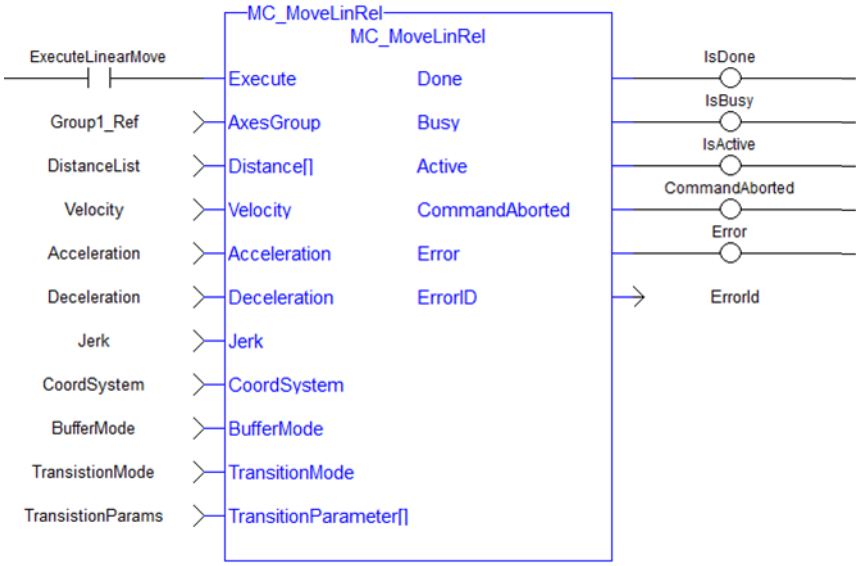
2.3.186 IL

```
BEGIN_IL
     CAL Inst_MC_MoveLinRel( ExecuteLinearMove, Group1_Ref, DistanceList,
Velocity, Acceleration, Deceleration, Jerk, CoordSystem, BufferMode,
TransitionMode, TransitionParams )
END_IL
```

2.3.187 FBD



2.3.188 FFLD



2.3.188.1 Coordinated Motion Reference Library

Function	Description
"MC_GrpSetPos" (➜ p. 501)	Sets the position of the group.

MC_GrpSetPos

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.

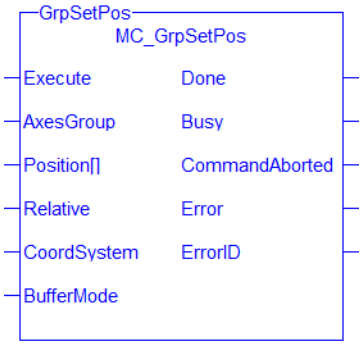


Figure 1-116: MC_GrpSetPos

2.3.189 Related Functions

"MC_ErrorDescription" (➜ p. 411), "MC_GrpReadCmdPos" (➜ p. 458)

See also "Coordinated Motion", the top-level topic for Coordinated Motion.

Arguments

2.3.190 Input

Execute	**Description**	On the rising edge, request to set the position of the group
	Data Type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
AxesGroup	**Description**	The axis group for which to set the positions
	Data Type	AXIS_GROUP_REF
	Range	n/a
	Unit	n/a
	Default	—
Position[]	**Description**	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.
	Data Type	LREAL
	Range	[0, Number of axes in group-1]
	Unit	n/a
	Default	—
Relative	**Description**	Request to set relative (1) or absolute (0) position
	Data Type	BOOL
	Range	1, 0
	Unit	n/a
	Default	—
CoordSystem	**Description**	The coordinate system used when setting the positions.
	Data Type	SINT
	Range	One of the following enumeration values: • MC_COORDINATE_SYSTEM_ACS = 0 • MC_COORDINATE_SYSTEM_MCS = 1 • MC_COORDINATE_SYSTEM_PCS = 2 Currently, only the ACS coordinate system is supported. See Coordinate Systems for more information.
	Unit	n/a
	Default	—
BufferMode	**Description**	Currently unused
	Data Type	SINT

	Range	[0, 0]
	Unit	n/a
	Default	—

2.3.191 Output

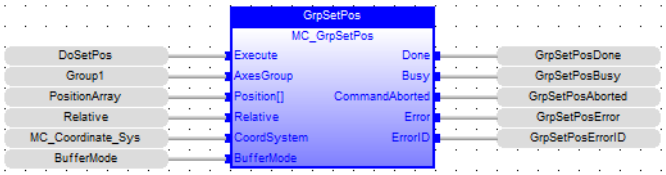
Done	**Description**	If True, then the command completed successfully.
	Data Type	BOOL
Busy	**Description**	Currently unused, returns FALSE
	Data Type	BOOL
CommandAborted	**Description**	Currently unused, returns FALSE
	Data Type	BOOL
Error	**Description**	If True, an error has occurred.
	Data Type	BOOL
ErrorID	**Description**	Indicates the error identifier if Error output is set to TRUE. See the table in PLCopen Function Block ErrorID Output.
	Data Type	INT

Example

2.3.192 ST

```
Inst_MC_GrpSetPos( DoSetPos, Group1, PositionArray, Relative, MC_
COORDINATE_SYSTEM_ACS, 0 );
```

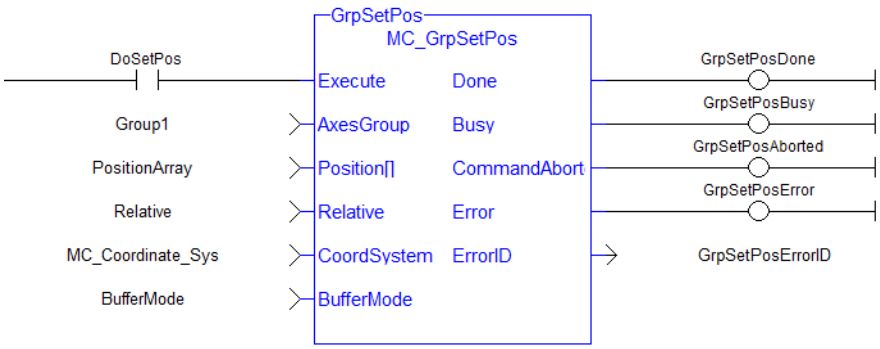
2.3.193 FBD



2.3.194 IL

```
BEGIN_IL
     CAL   Inst_MC_GrpSetPos( DoSetPos, Group1, PositionArray, Relative,
MC_COORDINATE_SYSTEM_ACS, BufferMode);
END_IL
```

2.3.195 FFLD


GrpSetPos
MC_GrpSetPos
DoSetPos
Execute
Done
GrpSetPosDone
Group1
AxesGroup
Busy
GrpSetPosBusy
PositionArray
Position[]
CommandAbort
GrpSetPosAborted
Relative
Relative
Error
GrpSetPosError
MC_Coordinate_Sys
CoordSystem
ErrorID
GrpSetPosErrorID
BufferMode
BufferMode

3 Fieldbus Library

3.1 EtherCAT Library

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

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

(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.

Execution Time

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

NOTE

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

Reason

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

Result

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

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

Solution

When this happens we recommend to:

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

See some **stats** about the FB execution time

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

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

- When sending multiple commands to a single drive, only one command can be sent at a time. Therefore the time to execute multiple commands is:
  Number of commands x *Execution time of a single command*

3.1.1.1 DriveParamRead

Description

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

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

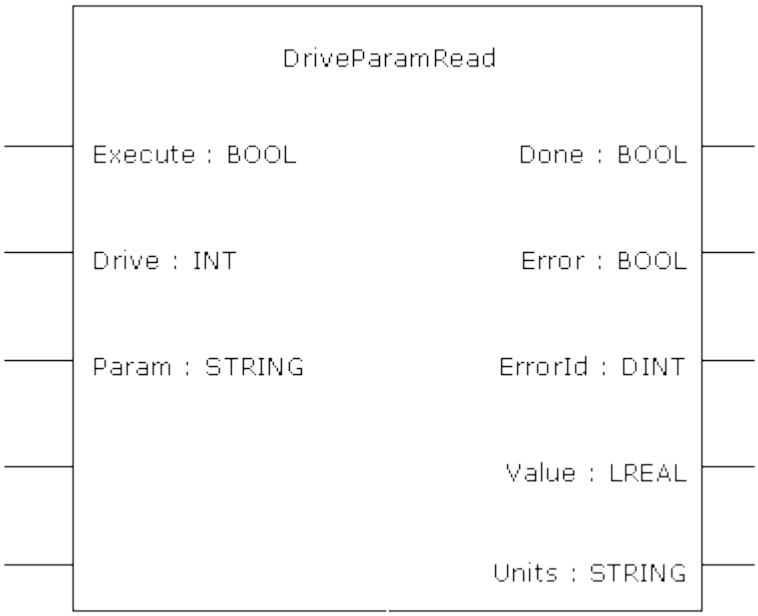


Figure 1-117: DriveParamRead

NOTE

This function block uses *and reserves* the EtherCAT SDO Channel. The SDO Channel will remain reserved until the done output is "true". Therefore, this FB should be called at each cycle until the done output is true. If it is not called at each cycle the rest of SDO communication (the AKD GUI Views, for example) will be blocked.

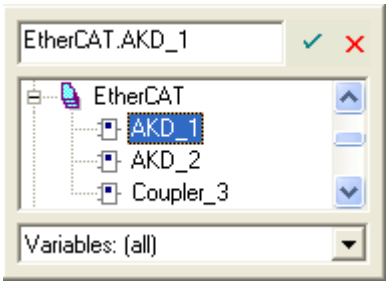
Using this FB in SFC P0 or P1 steps is not recommended as these steps are executed only once. If this FB is used in P0 or P1 then it must be used in an SFC N step to ensure the FB completes.

Arguments

Input

Execute

Description	On the rising edge of Execute, a drive parameter is read. The function block only handles one request at a time. If Execute is toggled quickly so that another rising edge occurs before the function block has completed, the function block does not issue a second read command.
Data type	BOOL
Range	0, 1
Unit	n/a
Default	—

Drive	**Description**	The address of the drive from which data is read. The first node usually has the value '1001'. The second node usually has the value '1002'. Alternately, you can use the members of the EtherCAT structure to specify a drive's address when you create the variable.
		
	Data type	INT
	Range	—
	Unit	n/a
	Default	—
Param	**Description**	The parameter to read.
	Data type	STRING
	Range	—
	Unit	n/a
	Default	—

Output

Done	**Description**	Indicates whether the DriveParamRead function block has completed without error.
	Data type	BOOL
	Unit	n/a
Error	**Description**	Indicates whether the DriveParamRead function block call has completed with error:
	Data type	BOOL
	Unit	n/a
ErrorID	**Description**	The DriveParamRead error result if Error is TRUE (see list of Error Codes in table below). Upon success, Error is set to zero.
	Data type	DINT
	Unit	n/a
Value	**Description**	The value of the drive parameter. Value is only set when the function block has successfully completed.
	Data type	LREAL
	Unit	n/a
Units	**Description**	The units of the drive parameter. Value is only set when the function block has successfully completed.
	Data type	STRING
	Unit	n/a

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

Table 1-5: List of EtherCAT Error Codes

Usage

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

Related Functions

DriveParamWrite

Example

Structured Text

```
(* Read PL.KP on first AKD Drive on EtherCAT network *)
(* The code continually calls the FB (without re-executing it) until the
first execution is done, then reads the returned value from the drive and
reset the FB *)

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

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

See example with animation

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


On Inst_DriveParamRead1.Done TRUE do
   Inst_DriveParamRead1( 0, 1001, 'PL.KP' );
    PositionProportionalGain 94.999000 := Inst_DriveParamRead1.Value 94.9
   ReadPropGain FALSE := 0;
End_DO;
```

3.1.1.2 DriveParamWrite

Description

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

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

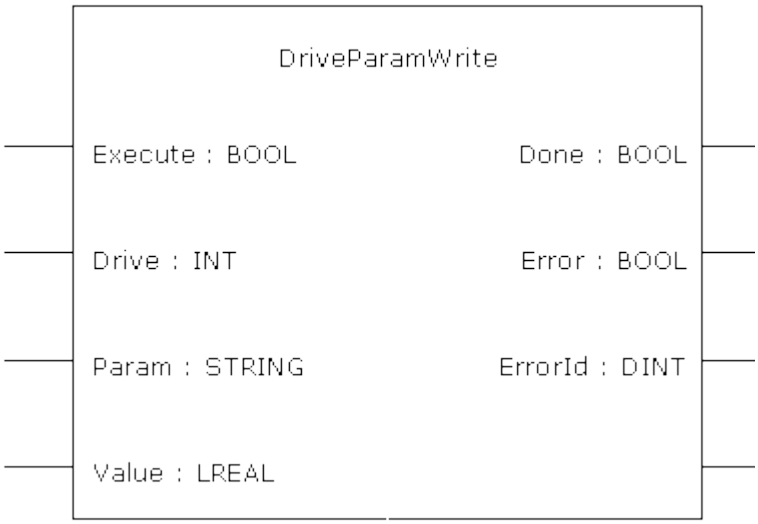


Figure 1-118: DriveParamWrite

NOTE

This function block uses *and reserves* the EtherCAT SDO Channel. The SDO Channel will remain reserved until the done output is "true". Therefore, this FB should be called at each cycle until the done output is true. If it is not called at each cycle the rest of SDO communication (the AKD GUI Views, for example) will be blocked.

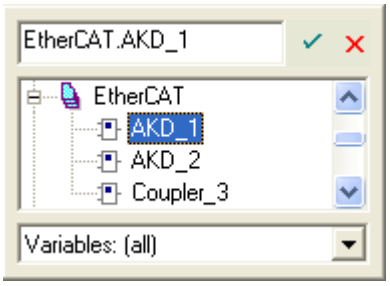
Using this FB in SFC P0 or P1 steps is not recommended as these steps are executed only once. If this FB is used in P0 or P1 then it must be used in an SFC N step to ensure the FB completes.

Arguments

Input

Execute

Description	On the rising edge of Execute, a drive parameter is set. The function block only handles one request at a time. If Execute is toggled quickly so that another rising edge occurs before the function block has completed, the function block does not issue a second write command.
Data type	BOOL
Range	0, 1
Unit	n/a
Default	—

Drive	**Description**	The address of the drive to which data is written to. The first node usually has the value '1001'. The second node usually has the value '1002'. Alternately, you can use the members of the EtherCAT structure to specify a drive's address when you create the variable.
		
	Data type	INT
	Range	—
	Unit	n/a
	Default	—
Param	**Description**	The parameter to write.
	Data type	STRING
	Range	—
	Unit	n/a
	Default	—
Value	**Description**	The value to set the drive parameter to.
	Data type	LREAL
	Range	—
	Unit	n/a
	Default	—

Output

Done	**Description**	Indicates whether the DriveParamWrite function block has completed without error.
	Data type	BOOL
	Unit	n/a
Error	**Description**	Indicates whether the DriveParamWrite function block call has completed with error.
	Data type	BOOL
	Unit	n/a
ErrorID	**Description**	The DriveParamWrite error result if Error is TRUE (see "List of EtherCAT Error Codes" (➜ p. 510)) Upon success, Error is set to zero.
	Data type	DINT
	Unit	n/a

Usage

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

Related Functions

DriveParamRead

Example

Structured Text

```
(* Write 58.000 to PL.KP of first AKD Drive on EtherCAT network *)
Inst_DriveParamWrite( TRUE, 1001, 'PL.KP', 58 );
```

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**.
- 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

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.

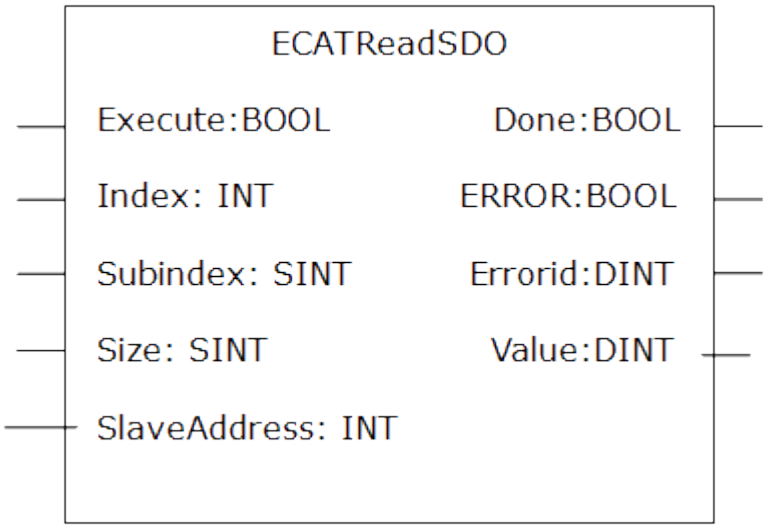


Figure 1-119: ECATReadSdo

State Diagram

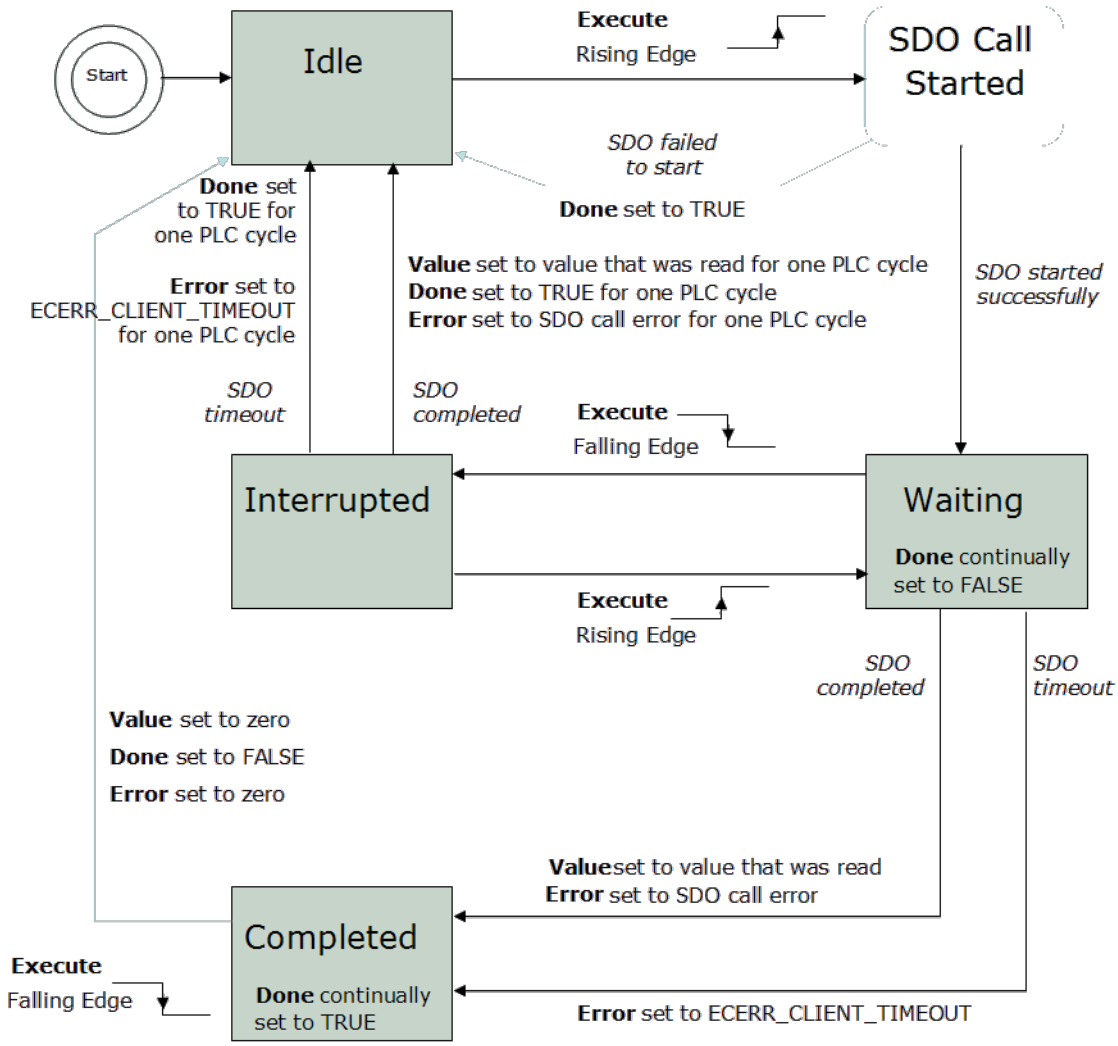


Figure 1-120: ECATReadSdo State Diagram

NOTE

This function block uses *and reserves* the EtherCAT SDO Channel. The SDO Channel will remain reserved until the done output is "true". Therefore, this FB should be called at each cycle until the done output is true. If it is not called at each cycle the rest of SDO communication (the AKD GUI Views, for example) will be blocked.

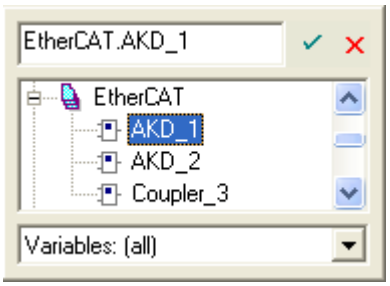
Using this FB in SFC P0 or P1 steps is not recommended as these steps are executed only once. If this FB is used in P0 or P1 then it must be used in an SFC N step to ensure the FB completes.

Arguments

Input

Execute	**Description**	On the rising edge of Execute, an SDO read command is issued. The function block only handles one SDO command at a time. If Execute is toggled quickly so that another rising edge occurs before the SDO command has completed, the function block does not issue a second SDO command.
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Index	**Description**	The object directory index of the data to be read. For more details, refer to: • Communication SDOs • Manufacturer specific SDOs • Profile specific SDOs
	Data type	INT
	Range	—
	Unit	n/a
	Default	—
Subindex	**Description**	The sub-index of the object directory variable to be read. For more details, refer to: • Communication SDOs • Manufacturer specific SDOs • Profile specific SDOs
	Data type	SINT
	Range	—
	Unit	n/a
	Default	—
Size	**Description**	The size (number of bytes) to write.
	Data type	SINT
	Range	1 - 4
	Unit	n/a
	Default	—

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



	Data type	INT
	Range	—
	Unit	n/a
	Default	—

Output

Done	**Description**	Indicates whether the SDO call has completed without error.
	Data type	BOOL
	Unit	n/a
Error	**Description**	Indicates whether the SDO call has completed with error:
	Data type	BOOL
	Unit	n/a
ErrorID	**Description**	The SDO call error result, if Error is TRUE (see). Upon success, Error is set to zero.
	Data type	DINT
	Unit	n/a
Value	**Description**	The value of the object directory variable being read. Value is only set when an SDO read command has successfully completed.
	Data type	DINT
	Unit	n/a

Related Functions

ECATWriteSDO

Example

Structured Text

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

3.1.2.2 ECATWriteSDO

Description

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

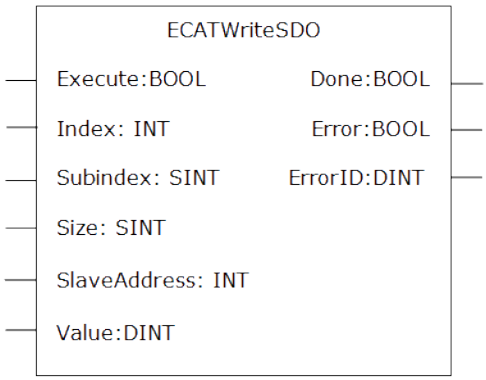


Figure 1-121: ECATWriteSdo

State Diagram

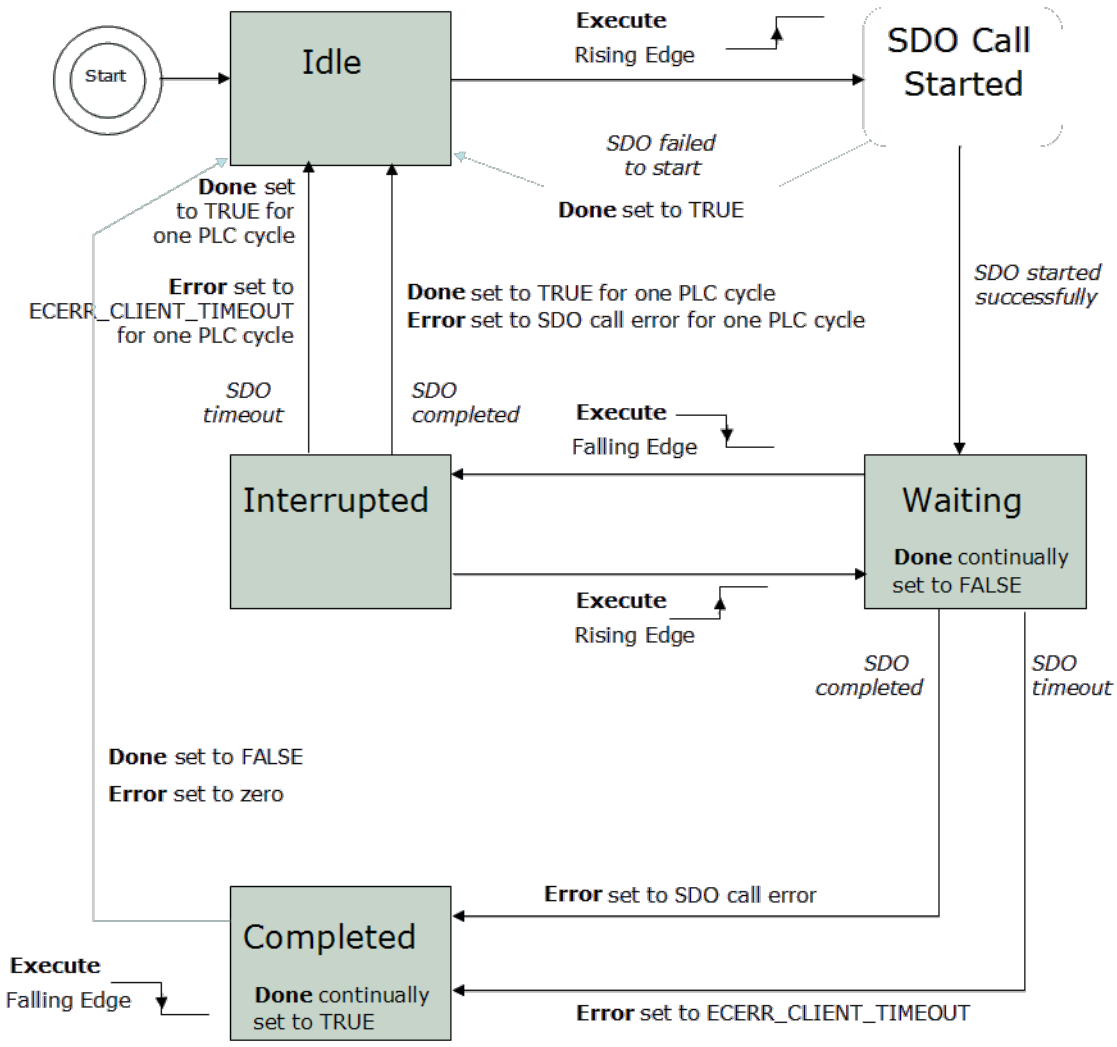


Figure 1-122: ECATWriteSdo State Diagram

NOTE

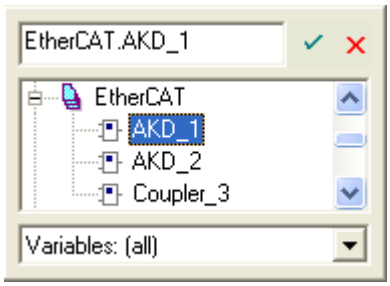
This function block uses *and reserves* the EtherCAT SDO Channel. The SDO Channel will remain reserved until the done output is "true". Therefore, this FB should be called at each cycle until the done output is true. If it is not called at each cycle the rest of SDO communication (the AKD GUI Views, for example) will be blocked.

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

Arguments

Input

Execute	**Description**	On the rising edge of Execute, an SDO 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.
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Index	**Description**	The object directory index of the data to be written to. For more details, refer to: • Communication SDOs • Manufacturer specific SDOs • Profile specific SDOs
	Data type	INT
	Range	—
	Unit	n/a
	Default	—
Subindex	**Description**	The sub-index of the object directory variable to be written to. For more details, refer to: • Communication SDOs • Manufacturer specific SDOs • Profile specific SDOs
	Data type	SINT
	Range	—
	Unit	n/a
	Default	—
Size	**Description**	The size (number of bytes) to write.
	Data type	SINT
	Range	1 - 4
	Unit	n/a
	Default	—

SlaveAddress	**Description**	The EtherCAT address of the slave from which data will be written to. The first node usually has the value '1001'. The second node usually has the value '1002'. Alternately, you can use the members of the EtherCAT structure to specify a drive's address when you create the variable.
		
	Data type	INT
	Range	—
	Unit	n/a
	Default	—
Value	**Description**	The value to write to the object directory variable.
	Data type	DINT
	Range	[-2147483648, 2147483648]
	Unit	n/a
	Default	—

Output

Done	**Description**	Indicates whether the SDO call has completed without error.
	Data type	BOOL
	Unit	n/a
Error	**Description**	Indicates whether the SDO call has completed with error:
	Data type	BOOL
	Unit	n/a
ErrorID	**Description**	The SDO call error result, if Error is TRUE (see). Upon success, Error is set to zero.
	Data type	DINT
	Unit	n/a

Related Functions

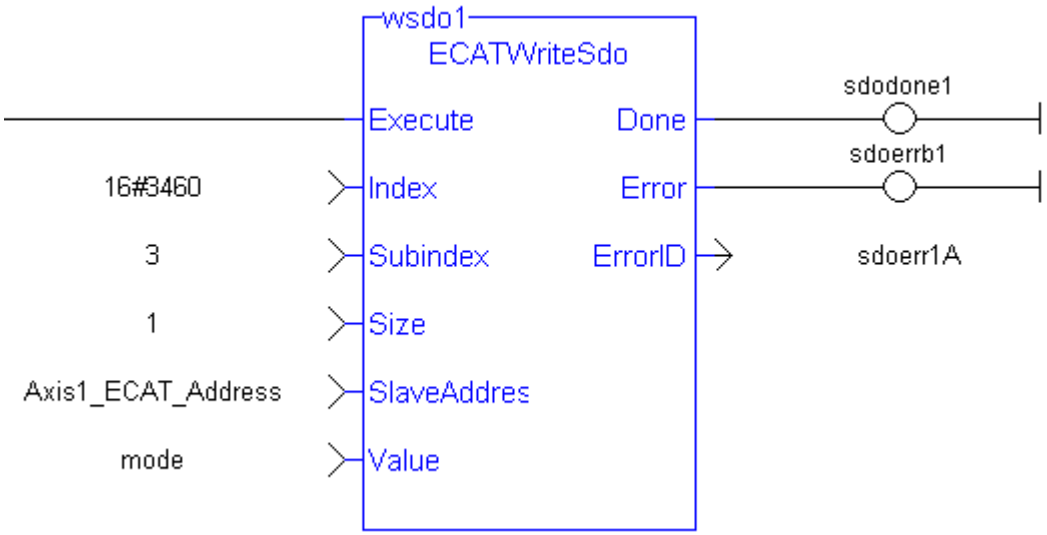
ECATReadSDO

Example

Structured Text

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

Ladder Diagram



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 ECATReadData

IMPORTANT

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

Description

This function allows a direct access to the memory image of the EtherCAT frame which is sent or received when you need to debug your application. You access the EtherCAT image element by giving the offset in the image and the size of the element.

If you have a device other than the drive, ECATReadData is used for more than just debug. It is used to get the status of the module (e.g. Stepper I/O slice).

Arguments

Input

Offset	**Description**	Offset in bytes from the beginning of the frame
	Data type	UINT
	Range	0-*size of frame* (maximum size of an Ethernet frame is 1500)
	Unit	bytes
	Default	—
Nbytes	**Description**	Number of bytes to read
	Data type	SINT
	Range	1, 2 or 4
	Unit	bytes

	Default	—
Direction	**Description**	Direction of the frame (true = output image, false = input image).
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—

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)

Output

Value	**Description**	Value of the EtherCAT frame
	Data type	DINT
	Unit	n/a

Related Functions

ECATGetObjVal

Example

Structured Text

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

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

3.1.3.2 ECATWriteData

IMPORTANT

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

Description

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

If you have a device other than the drive, ECATWriteData is used for more than just debug. It is used to set the status of the module (e.g. Stepper I/O slice) in the case your project is based on an external XML file because it contains unsupported EtherCAT Device.

Arguments

Input

Offset	**Description**	Offset in bytes from the beginning of the frame

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

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)

Output

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

Related Functions

ECATReadData

3.1.3.3 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.

3.1.4 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 ECATGetStatus (Function)

Description

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

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

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

Arguments

Input

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

Output

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

Related Functions

ECATSetControl

Example

Structured Text

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


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

3.1.4.2 ECATSetControl (Function)

Description

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

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

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

Arguments

Input

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

Output

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

Related Functions

ECATGetStatus

3.2 Profibus Library

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
PBGetExchPrio **PBSetExchPrio**	Return or change the priority of the I/O exchange thread. Which can be: • **#define CIF_EXCHANGE_PRIORITY_NORMAL 0** /* Profibus exchange thread priority lower than VM thread priority */ • **#define CIF_EXCHANGE_PRIORITY_HIGHER 1** /* Profibus exchange thread priority equal to VM thread priority */
PBGetExchPeriod **PBSetExchPeriod**	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. *Example*: 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)
PBGetExchNb	Return the amount of exchange performed with the HW
PBGetLastPeriod	Return the last period between two exchanges
PBGetMaxPeriod	idem but max period

Name	Description
PBResetStats	Reset the previous statistics of the I/O exchange thread. Which are: • the number of exchanges done • the last exchange period • the maximum exchange period Statistics are reset each time the driver is Open (i.e. when the program start)

Table 1-6: List of Profibus FB

4 System Library

Name	Description
"PrintMessage" (➜ p. 528)	Generate an output message in the log windows.
"GetCtrlErrors" (➜ p. 529)	Get a list of the active errors and alarms on the controller.
"ClearCtrlErrors" (➜ p. 530)	Clears the list of active errors and alarms on the controller.
"GetCtrlPerf" (➜ p. 532)	Generate a text file with performance statistics of the controller.
"GetCtrlInfo" (➜ p. 531)	Get the serial, model, and/or part number of the controller.

Table 1-7: 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.
Data type	DINT
Range	[0 , 4] Defines are: LEVEL_DEBUG, LEVEL_INFO, LEVEL_WARNING, LEVEL_ERROR, LEVEL_CRITICAL
Unit	n/a

	Default	—
Message	**Description**	Content of the message. A string of 255 characters maximum.
	Data type	String
	Range	1 to 255 characters
	Unit	n/a
	Default	—

4.1.2.2 Output

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

4.1.3 Usage

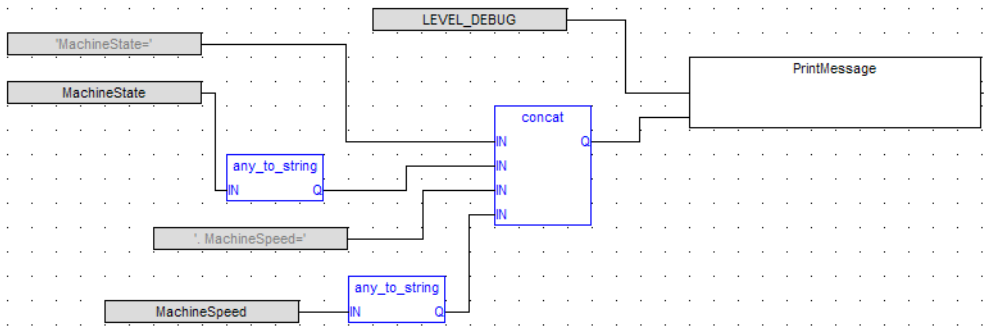
```
PrintMessage(  LEVEL_DEBUG, 'Message string to be logged' );
```

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



4.2 GetCtrlErrors

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

EN	**BOOL**	Enable
ActiveError	**BOOL[100]**	Array of bool with the size equal to 100
ActiveAlarm	**BOOL[100]**	Array of bool with size equal to 100

4.2.1.2 Output

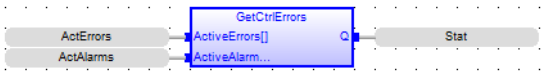
OK	**BOOL**	
Q	**DINT**	Status of the execution

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

4.2.2.1 FBD



4.2.2.2 FFLD

4.2.2.3 ST

```
GetCtrlErrors( ActErrors(*BOOL*), ActAlarms(*BOOL*) );
```

4.3 ClearCtrlErrors

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

EN	**Data Type**	Enable the function

4.3.3 Output

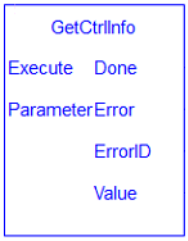
Q	**Data Type**	BOOL

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.

4.4 GetCtrlInfo

This function block returns a String containing the value of the control parameter requested.



4.4.1 Arguments

4.4.1.1 Input

Execute	**Description**	Rising edge of enable initiates read of parameter
	Data Type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
Parameter Number	**Description**	Parameter number to read
	Data Type	INT
	Range	[1,3]
	Unit	n/a
	Default	1 = AKD PDMM serial number 2 = AKD PDMM model number 3 = AKD PDMM part number0 These parameters are also Internal Defines, as shown in the table below.

#define	CTRLINFO_SERIAL_NUMBER	1
#define	CTRLINFO_MODEL_NUMBER	2
#define	CTRLINFO_PART_NUMBER	3

4.4.1.2 Output

Done	**Description**	Indication that read completed without error
	Data Type	BOOL
Error	**Description**	Indication that read completed with error
	Data Type	BOOL
ErrorID	**Description**	Error value to indicate error condition
	Data Type	INT NO Error : 0 Invalid parameter : 1 Error reading data : 2 Not valid on non-PDMM : 3 These parameters are also Internal Defines, as shown in the table below.

#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

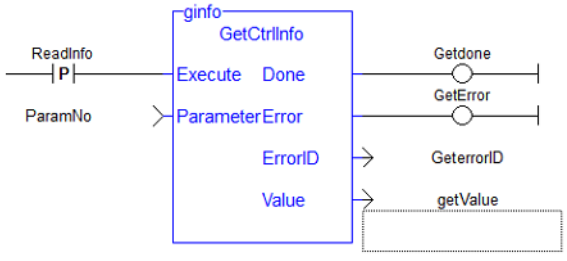
Value	**Description**	String containing data that was read.
	Data Type	STRING

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.

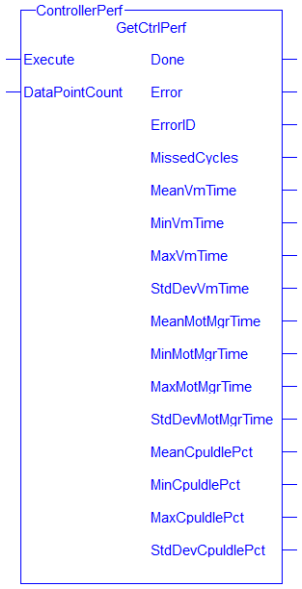


Figure 1-123: GetCtrlPerf

See also:

- Differences Between Functions and Function Blocks
- Calling a function block

4.5.2 Arguments

4.5.2.1 Input

Execute	**Description**	On the rising edge, request to collect the controller's performance data.
	Data type	BOOL
	Range	0,1
	Unit	n/a
	Default	—
DataPointCount	**Description**	The number of motion manager cycles over which performance statistics will be gathered.
	Data type	UDINT
	Range	2 - 240,000
	Unit	n/a
	Default	—

4.5.2.2 Output

Done	**Description**	If True, then the command completed successfully.
	Data type	BOOL
Error	**Description**	If True, an error has occurred.
	Data type	BOOL
ErrorID	**Description**	Indicates the error if Error output is set to TRUE.ErrorID = 0 indicates no error, ErrorID = 1 indicates an error
	Data type	INT
MissedCycles	**Description**	Indicates the number of missing VM cycles.

	Data type	UDINT
MeanVmTime	**Description**	The mean VM execution time measured in microseconds.
	Data type	LREAL
StdDevVmTime	**Description**	The standard deviation of the VM execution time measured in microseconds.
	Data type	LREAL
MinVmTime	**Description**	The minimum VM execution time measured in microseconds.
	Data type	LREAL
MaxVmTime	**Description**	The maximum VM execution time measured in microseconds.
	Data type	LREAL
MeanMotMgrTime	**Description**	The mean motion manager execution time measured in microseconds.
	Data type	LREAL
StdDevMotMgrTime	**Description**	The standard deviation of the motion manager execution time measured in microseconds.
	Data type	LREAL
MinMotMgrTime	**Description**	The minimum motion manager execution time measured in microseconds.
	Data type	LREAL
MaxMotMgrTime	**Description**	The maximum motion manager execution time measured in microseconds.
	Data type	LREAL
MeanCpuIdlePct	**Description**	The mean percentage of time the controller CPU is idle.
	Data type	LREAL
StdDevCpuIdlePct	**Description**	The standard deviation of the measurements of the time that the controller CPU is idle.
	Data type	LREAL
MinCpuIdlePct	**Description**	The minimum measurement of the time that the controller CPU is idle.
	Data type	LREAL
MaxCpuIdlePct	**Description**	The maximum measurement of the time that the controller CPU is idle.
	Data type	LREAL

5 Kollmorgen UDFBs

A Kollmorgen UDFB[1] 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
FB_AKDFltRpt	
FB_AxisPlsPosModulo	
FB_AxisPlsPosModulo	
FB_AxisPlsPosNoModulo	
FB_Cylinder	
FB_ElapseTime	
FB_FirstOrderDigitalFilter	
FB_PWDutyOutput	
FB_S700FltRpt	
FB_ScaleInput	
FB_ScaleOutput	
FB_TemperaturePID (➜ p. 556)	
MCFB_AKDFault (➜ p. 648)	
MCFB_AKDFaultLookup (➜ p. 650)	
"MCFB_GearedWebTension" (➜ p. 636)	
MCFB_Jog	
MCFB_StepAbsolute	
MCFB_StepAbsSwitch	
MCFB_StepAbsSwitchFastInput	
MCFB_StepBlock	
MCFB_StepLimitSwitch	
MCFB_StepLimitSwitchFastInput	
MCFB_StepRefPulse	
MLFB_HomeFindHomeFastInput	
MLFB_HomeFindHomeFastInputModulo	
MLFB_HomeFindHomeInput	
MLFB_HomeFindHomeInputThenZeroAngle	
MLFB_HomeFindLimitFastInput	
MLFB_HomeFindLimitFastInputModulo	
MLFB_HomeFindLimitInput	
MLFB_HomeFindLimitInputThenZeroAngle	
MLFB_HomeFindZeroAngle	
MLFB_HomeMoveUntilPosErrExceeded	
MLFB_HomeMoveUntilPosErrExceededThenZeroAngle	
MLFB_HomeUsingCurrentPosition	
MLFB_Jog	

[1]"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
MLFB_PlsPosFw	
MLFB_PlsPosFwBw	
MLFB_PlsTimeFw	
"PipeNetwork_FFLD" (➜ p. 553)	
"ProfilesCode_FFLD" (➜ p. 555)	

Table 1-8: 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**.

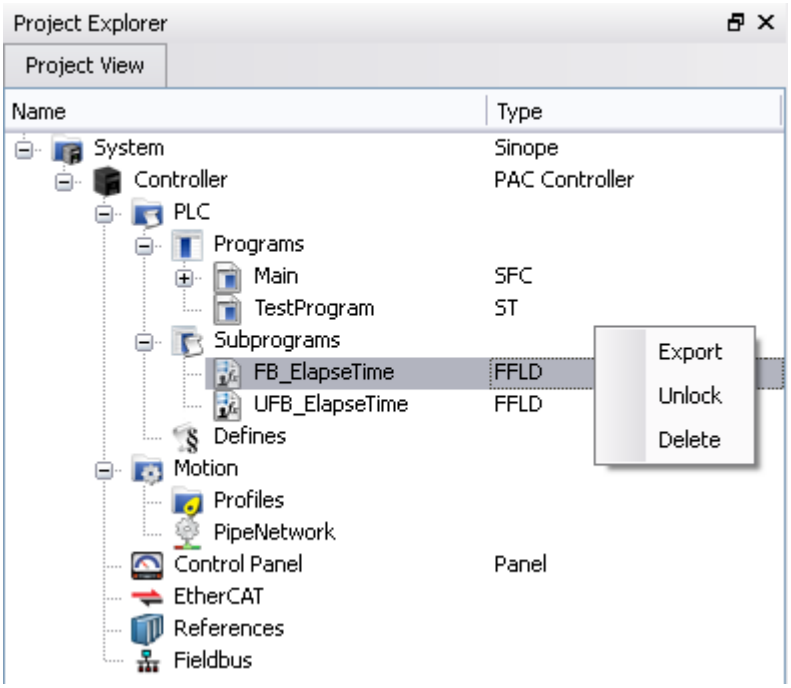
NOTE

You cannot create the instance of the UDFB directly from the dictionary or from the PLC Editor.

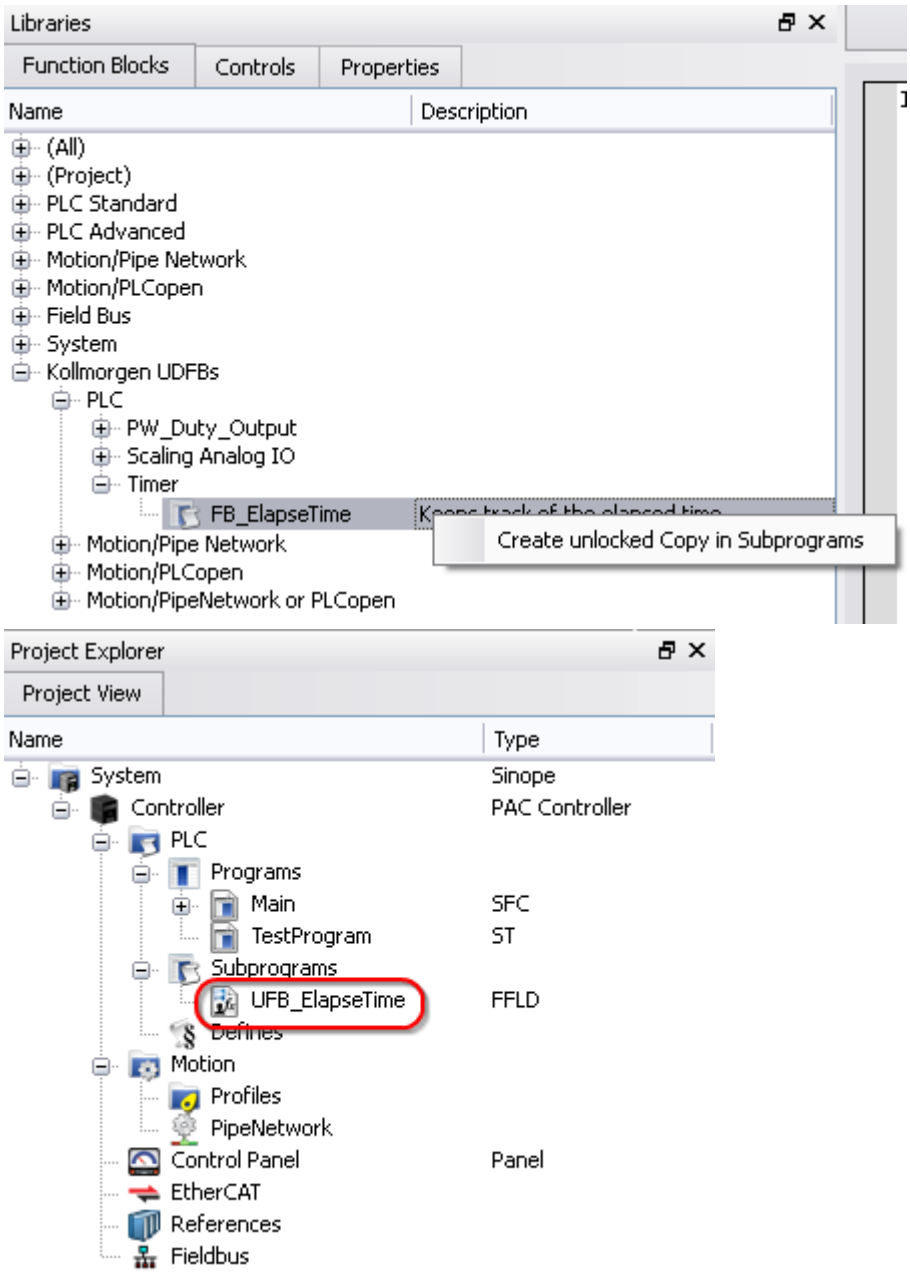
5.2 Working with Kollmorgen UDFBs

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

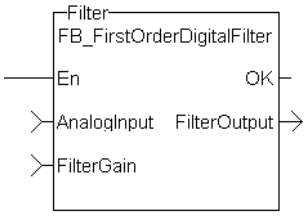


Figure 1-124: CBS First Order Digital Filter

5.2.1.2 Arguments

Inputs

EN	Description	Enables execution (FFLD only)
	Data type	BOOL
	Range	—
	Unit	n/a
	Default	
AnalogInput	Description	Analog Input from transducer
	Data type	INT
	Range	—
	Unit	n/a
	Default	
FilterGain	Description	Filter Gain
	Data type	REAL
	Range	[1 - 0.05]
	Unit	n/a
	Default	—

Outputs

OK	Description	Execution Complete
	Data type	BOOL
	Range	[0,1]
	Unit	

FilterOutput	Description	Filtered analog input value
	Data type	REAL
	Range	[0,1]
	Unit	

5.2.1.3 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.

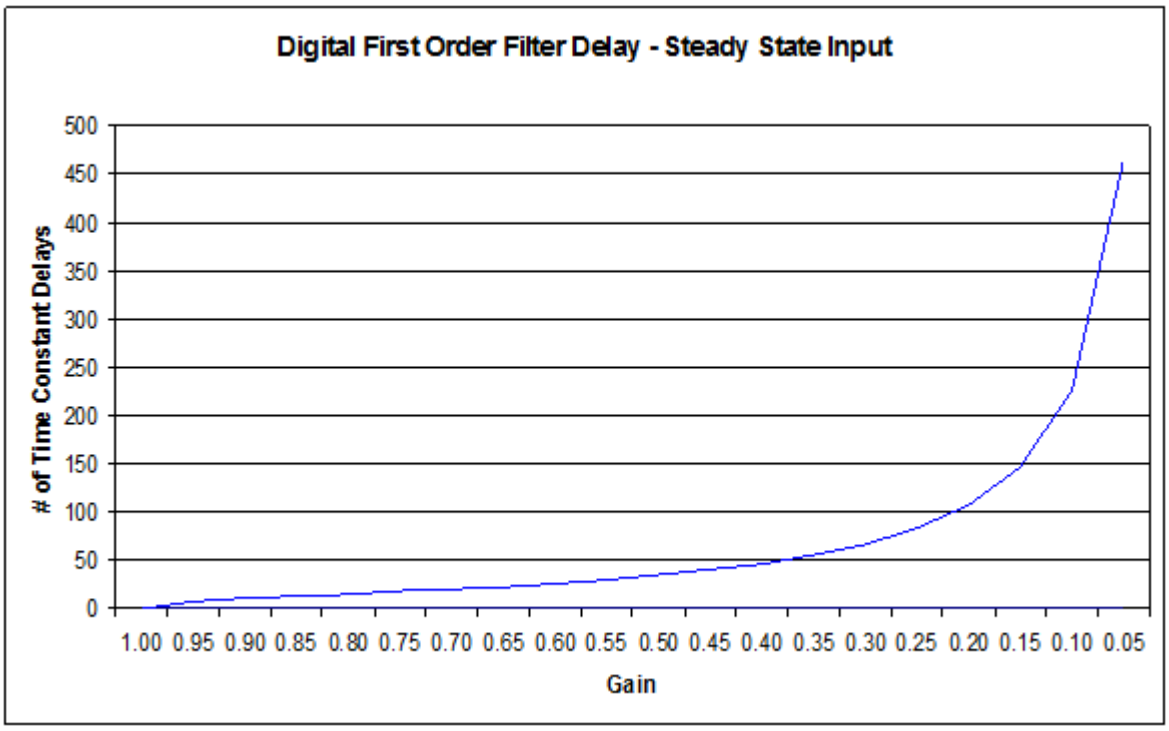
FilterGain	FilterInput	FilterOutput
0.8	0	0
	100	80
	100	96
	100	99.2
	100	99.84
	100	99.968
	100	99.9936
	100	99.99872
	100	99.999744
	100	99.9999488
	100	99.99998976
	100	99.99999795
	100	99.99999959
	100	99.99999992
	100	99.99999998
	100	100
	100	100
	100	100
	100	100
	100	100
	100	100
	100	100
	100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

Table 1-9: 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	
100.5303014	**100.0399168**	**-0.490384645**	**Averages**

Table 1-10: Filter Input Lag Example - Random Input

5.2.1.4 Related Functions

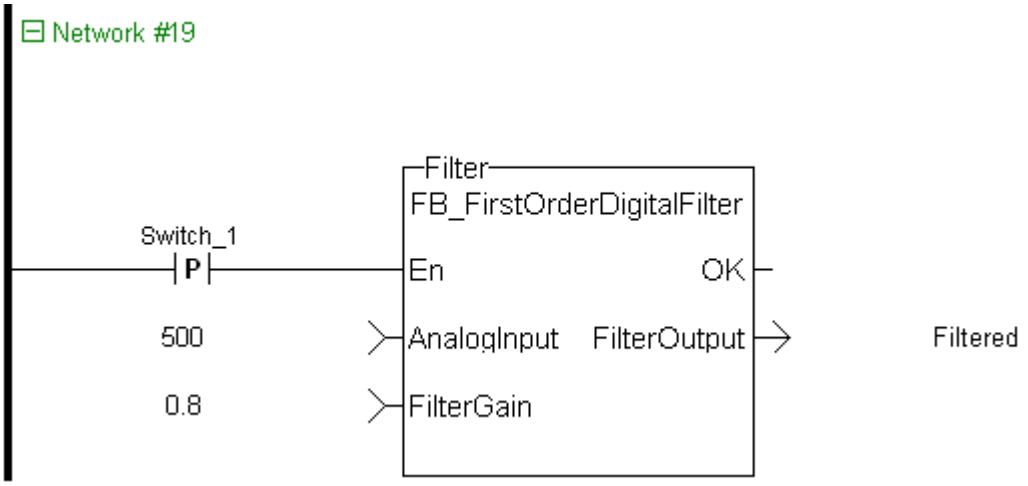
None.

5.2.1.5 Example

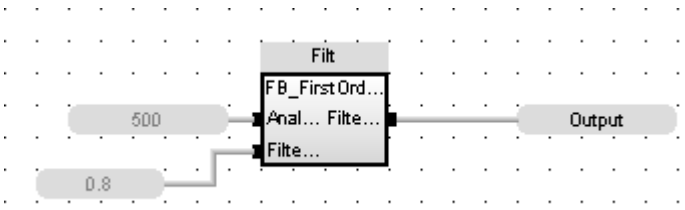
Structured Text

```
Inst_FB_FirstOrderDigitalFilter( AnalogInput:=500, FilterGain:=0.8 );
FilterOutput:= Inst_FB_FirstOrderDigitalFilter.FilterOutput
```

Ladder Diagram



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

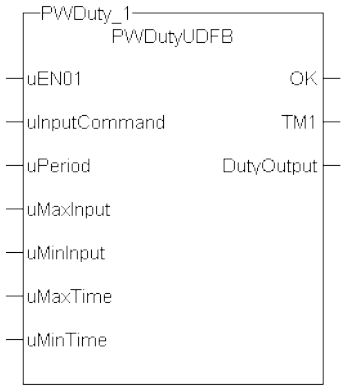


Figure 1-125: Pulse Width Duty Cycle

5.2.2.2 Arguments

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	TIME
	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	TIME
	Range	[0 , 1]
	Unit	n/a
	Default	—
uMinTime	Description	Minimum on time for the Output
	Data type	TIME
	Range	[0 , 1]
	Unit	n/a
	Default	—

Output

OK	Description	Function block is OK.
	Data type	BOOL
	Unit	n/a
TM1	Description	Duty cycle on time
	Data type	TIME
	Unit	n/a
DutyOutput	Description	Indicates if output is on or off

Data type	BOOL
Unit	n/a

5.2.2.3 Usage

Flash a warning light for operators.

5.2.2.4 Related Functions

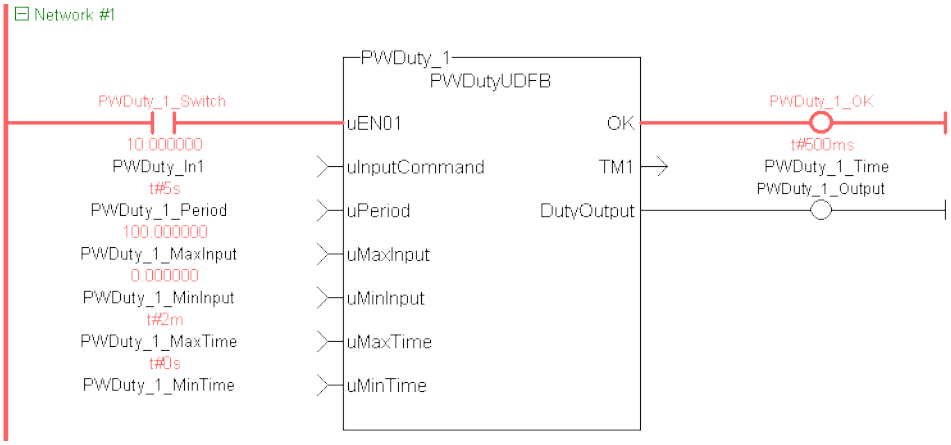
Timers

5.2.2.5 Example

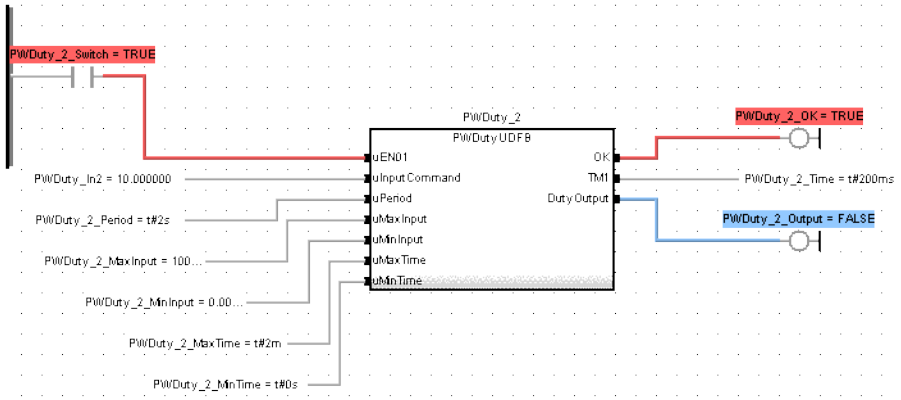
Structured Text

```
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;
```

Ladder Diagram



Function Block Diagram



5.2.2.6 FB_ScaleInput

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:

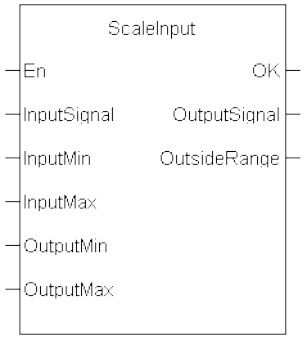


Figure 1-126: Scale Input

Arguments

Input

InputSignal	Description	Un-scaled input signal
	Data type	DINT
	Range	[0 , 4]
	Unit	n/a
	Default	—
InputMin	Description	Minimum value of accepted input signal range
	Data type	DINT

	Range	[0 , 4]
	Unit	n/a
	Default	—
InputMax	Description	Maximum value of accepted input signal range
	Data type	DINT
	Range	[0 , 4]
	Unit	n/a
	Default	—
OutputMin	Description	Output value mapped to the InputMin
	Data type	LREAL
	Range	[0 , 4]
	Unit	n/a
	Default	—
OutputMax	Description	Output value mapped to the InputMax
	Data type	LREAL
	Range	[0 , 4]
	Unit	n/a
	Default	—

Output

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

Usage

Scale an analog signal from a drive.

Related Functions

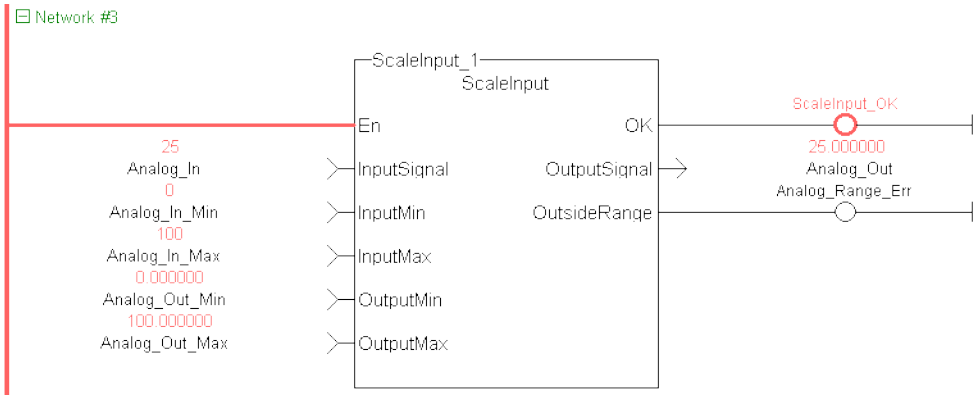
UDFB ScaleOutput

Example

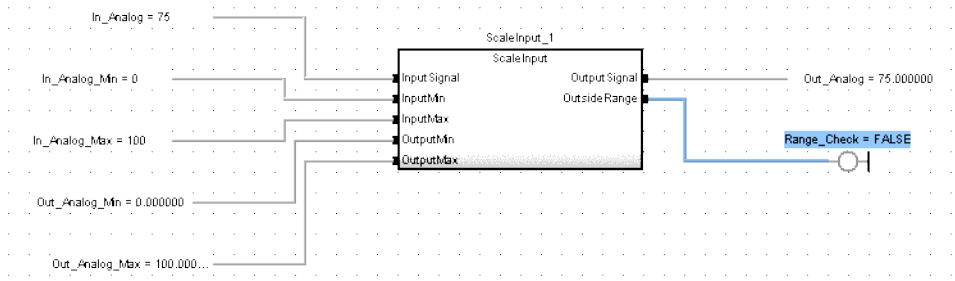
Structured Text

```
Inst_ScaleInput22( Analog_In    65 , 0, 100,0,100);
Analog_Out 65.000000 :=Inst_ScaleInput22.OutputSignal 65.000000 ;
Analog_Range_Err FALSE :=Inst_ScaleInput22.OutsideRange FALSE ;
```

Ladder Diagram



Function Block Diagram



5.2.2.7 FB_ ScaleOutput

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:

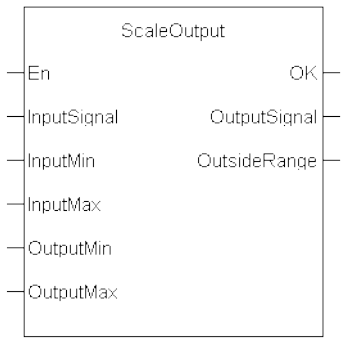


Figure 1-127: Scale Output

Arguments

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	—

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

Usage

Scale an analog signal to a drive.

Related Functions

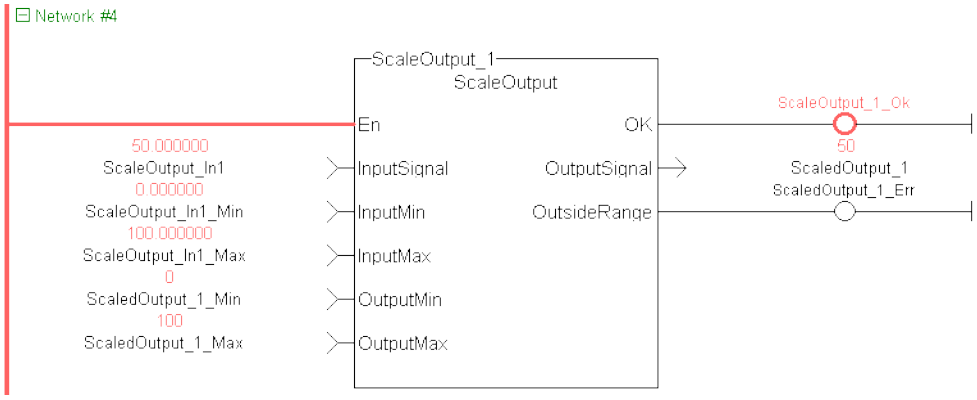
UDFB ScaleInput

Example

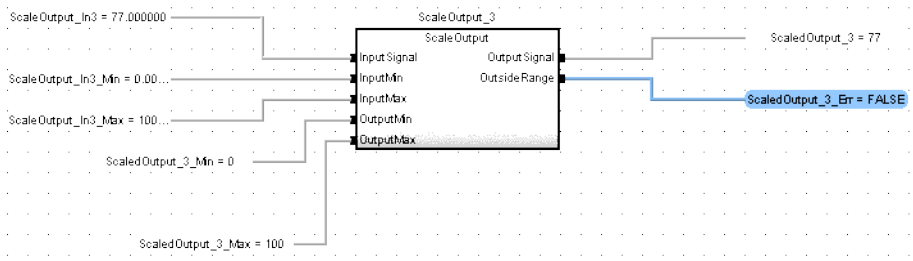
Structured Text

```
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;
```

Ladder Diagram



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.

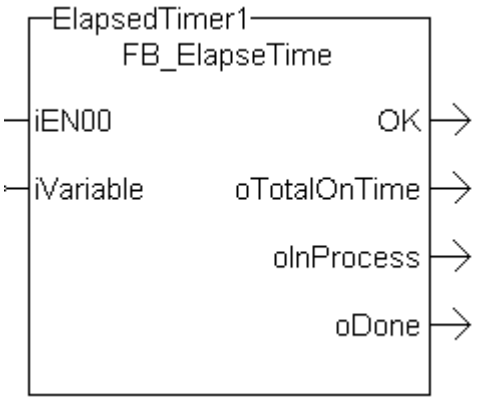


Figure 1-128: FB_ElapseTime

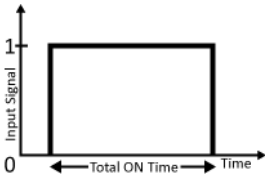


Figure 1-129: MFB_ElapseTime – Time Diagram

5.2.3.2 Arguments

Input

iEN00	Description	Enable for the block
	Data type	Boolean
	Range	FALSE or TRUE
	Unit	n/a
	Default	FALSE
iVariable	Description	The variable to be tracked
	Data type	Boolean
	Range	FALSE or TRUE
	Unit	n/a
	Default	FALSE

Output

OK	Description	Function Block OK. This output follows the state on iEN00 input
	Data type	Boolean
	Range	FALSE or TRUE
	Unit	n/a
oTotalOnTime	Description	The amount of time the iVariable is turned on.
	Data type	Time
	Range	0ms – 24h
	Unit	ms

oInProcess	Description	The state of block's execution whether or not it is still keeping track of time
	Data type	Boolean
	Range	FALSE or TRUE
	Unit	n/a
oDone	Description	The state of block's execution whether or not it is completed
	Data type	Boolean
	Range	FALSE or TRUE
	Unit	n/a

5.2.3.3 Usage

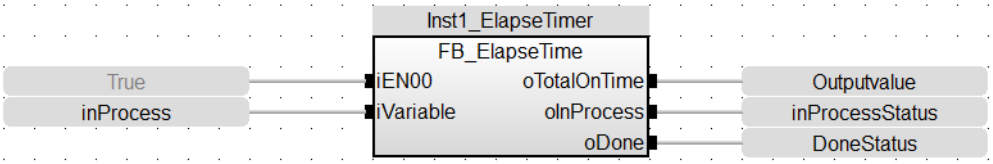
- Enable the block by setting iEN00 to TRUE
- Either manually set iVariable to TRUE or have the application set this variable to TRUE
- Once oDone returns TRUE, read the oTotalOnTime to find out how long iVariable was on.

5.2.3.4 Example

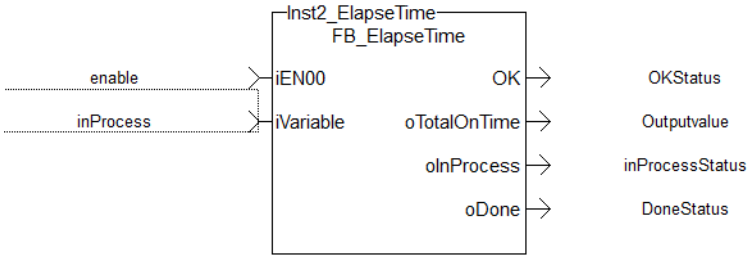
Structured text

```
Inst_FB_ElapseTime1( TRUE, InProcess );


IF Inst_FB_ElapseTime1.oDone THEN
Outputvalue := Inst_FB_ElapseTime1.oTotalOnTime;
END_IF;
```

Function Block Diagram



Free Form Ladder Diagram



5.2.4 PipeNetwork_FFLD

5.2.4.1 Description

This function is used to call the PNCode Function Block in FFLD POUs. 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 POUs 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.

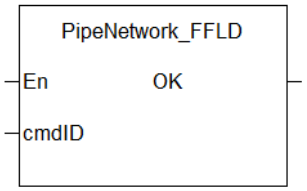


Figure 1-130: PipeNetwork_FFLD

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

Inputs

En	**Description**	Request to initialize the Pipe Network
	Data type	BOOL
	Range	0, 1
	Unit	n/a
	Default	—
cmdID	**Description**	Commands used to start and initialize the Pipe Network • MLPN_CREATE_OBJECTS – Create Pipe Network • MLPN_POWER_ON – Power on all axes • MLPN_POWER_OFF – Power off all axes • MLPN_ACTIVATE – Activate the pipes • MLPN_CONNECT – Connect the axes to the pipes • MLPN_DEACTIVATE – Deactivate the pipes
	Data type	DINT
	Range	n/a
	Unit	n/a
	Default	—

Outputs

OK	**Description**	Returns TRUE when the function has completed
	Data type	BOOL
	Unit	n/a

5.2.4.3 Usage

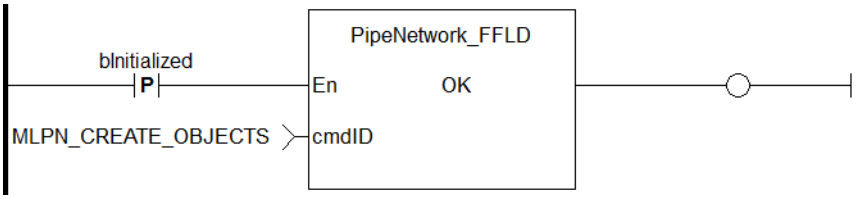
- This is a special function that should only be used in Pipe Network applications that contain FFLD POUs that call PNCode.
- To use this Function, PipeNetwork must be declared as a global variable in the dictionary.

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.

5.2.4.4 Example

FFLD



5.2.5 ProfilesCode_FFLD

5.2.5.1 Description

This function is used to call the Profiles Code Function Block in FFLD POUs. Internally this function calls the Function Block ProfilesCode.

This is a special function which should only be used in applications that contain FFLD POUs that call ProfilesCode. Calling this function instead of ProfilesCode in FFLD POUs 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.

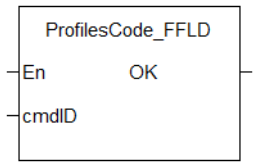


Figure 1-131: ProfilesCode_FFLD

5.2.5.2 Arguments

Inputs

En	**Description**	Request to initialize the Pipe Network
	Data type	BOOL
	Range	0,1
	Unit	n/a
	Default	—

cmdID	**Description**	Commands used to start and initialize the Pipe Network • MLPR_CREATE_PROFILES - Creation and initialization of profiles.
	Data type	DINT
	Range	n/a
	Unit	n/a
	Default	—

Outputs

OK	**Description**	Returns TRUE when the function has completed
	Data type	BOOL
	Unit	n/a

5.2.5.3 Usage

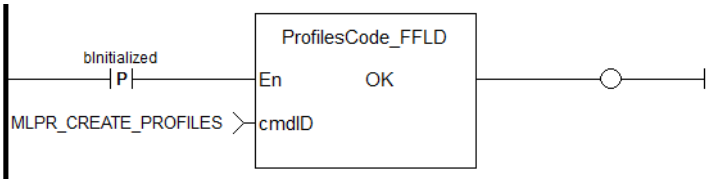
- This is a special function that should only be used in applications that contain FFLD POUs that call ProfilesCode.
- To use this function, Profiles must be declared as a global variable in the dictionary.

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.

5.2.5.4 Example

FFLD



5.2.6 FB_TemperaturePID

5.2.6.1 Description

This function block provides PID temperature control with auto tuning.

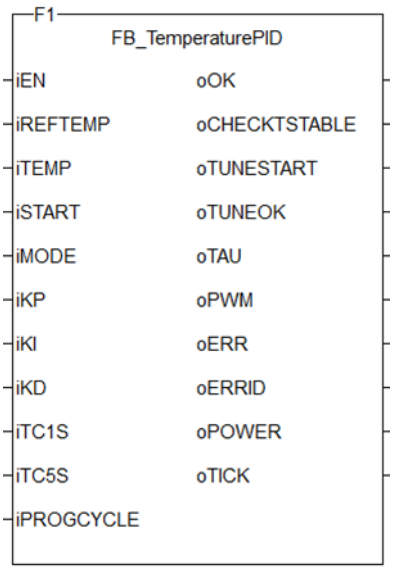


Figure 1-132: The TemperaturePID user-defined function block

5.2.6.2 Arguments

Inputs

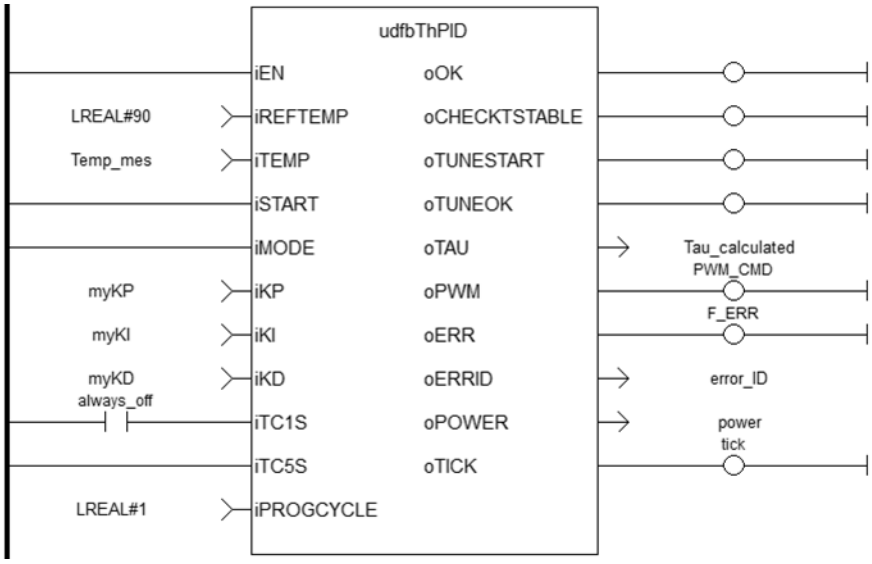
iEN	**BOOL**	Enable function
iREFTEMP	**LREAL**	Reference temperature [°C]
iTEMP	**LREAL**	Actual temperature [°C]
iSTART	**BOOL**	Start PID or auto tuning
iMODE	**BOOL**	FALSE-automatic, TRUE-tuning
iKP	**LREAL**	PID Proportional Gain
iKI	**LREAL**	PID Integral Gain
iKD	**LREAL**	PID Derivative Gain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

Outputs

oOK	**BOOL**	Function enabled
oCHECKSTABLE	**BOOL**	TRUE when checking if ambient temperature is stable
oTUNESTART	**BOOL**	Tuning is started
oTUNEOK	**BOOL**	Tuning is completed
oTAU	**LREAL**	System Time Constant[s]
oPWM	**BOOL**	PWM command for heater
oERR	**BOOL**	Function error
oERRID	**INT**	Function ID error (in case of oERR=TRUE)
oPOWER	**LREAL**	% of power requested from heater (100%=full power)
oTICK	**BOOL**	Pulse every sampling time

5.2.6.3 Usage

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

After, PID gains are calculated as:

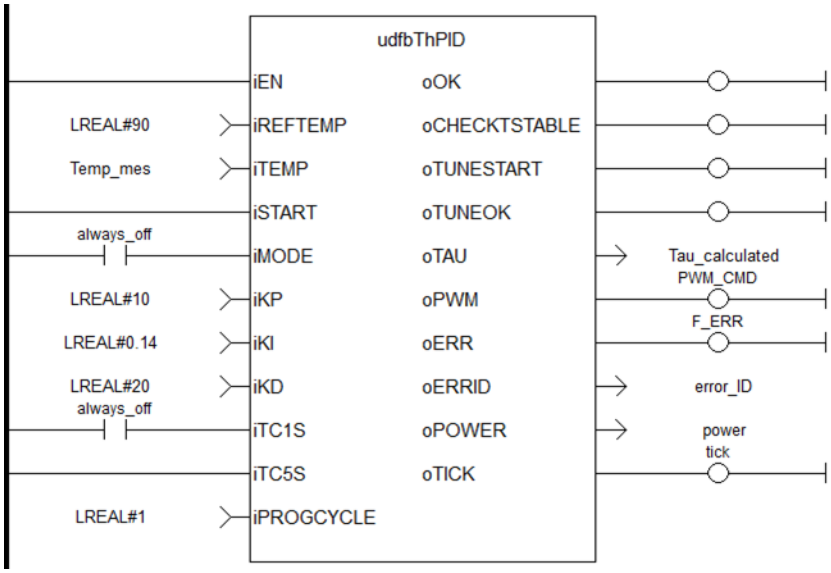
```
Kp=10
Ki=0.14
delta_time = time to complete Phase2

Kd=SQRT(delta_time)*7
```

The tuning is completed.

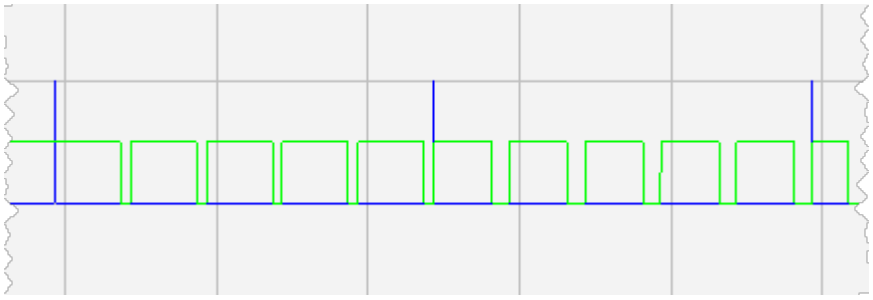
TIP

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

Start PID Controller

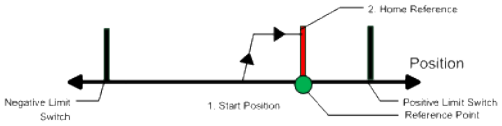


Upon starting the PID controller, the output **oPWM** is modulated 5 times within the sampling time (blue line is **oTICK**, green line is **oPWM**) and each pulse length depends on output **oPOWER** (100%=full length).



5.2.6.4 MLFB_HomeFindHomeInput

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.

Arguments

Input

ibExecute	Description	Start homing, edge-triggered
	Data type	BOOL
iAxisID	Description	ID of Axis block of Pipe Network

	Data type	DINT
iPosition	Description	Reference position
	Data type	LREAL
ibDirection	Description	0=positive, 1=negative
	Data type	BOOL
iVelocity	Description	Reference speed
	Data type	LREAL
iAcceleration	Description	Reference acceleration
	Data type	LREAL
iDeceleration	Description	Reference deceleration
	Data type	LREAL
ibHomeInput	Description	Home input, high-active
	Data type	BOOL
ibPosLimitSwitch	Description	Positive limit switch, high-active
	Data type	BOOL
ibNegLimitSwitch	Description	Negative limit switch, high-active
	Data type	BOOL
iTimeout	Description	Time monitoring (T#0ms: off)
	Data type	TIME

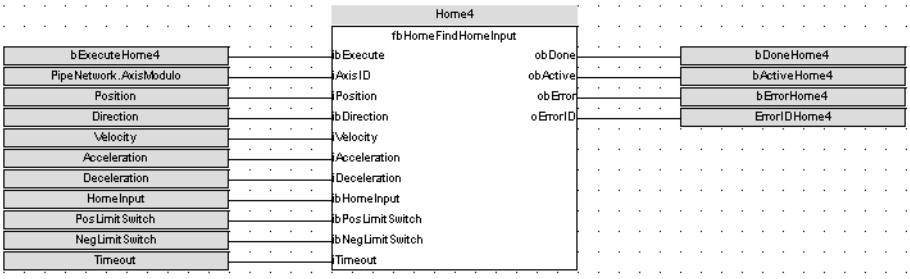
Output

obDone	Description	Done bit
	Data type	BOOL
obActive	Description	Active bit
	Data type	BOOL
obError	Description	Error bit
	Data type	BOOL
oErrorID	Description	Error identifier, see list here
	Data type	DINT

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

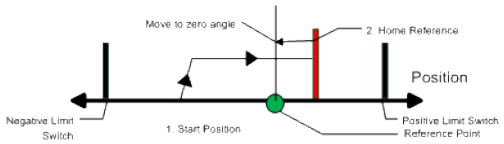
Example

Function Block Diagram



5.2.6.5 MLFB_HomeFindHomeInputThenZeroAngle

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.

Arguments

Input

ibExecute	Description	Start homing, edge-triggered
	Data type	BOOL
iAxisID	Description	ID of Axis block of Pipe Network
	Data type	DINT
iPosition	Description	Reference position
	Data type	LREAL
ibDirection	Description	0=positive, 1=negative
	Data type	BOOL
iVelocity	Description	Reference speed
	Data type	LREAL
iAcceleration	Description	Reference acceleration
	Data type	LREAL
iDeceleration	Description	Reference deceleration
	Data type	LREAL
ibHomeInput	Description	Home input, high-active
	Data type	BOOL
ibPosLimitSwitch	Description	Positive limit switch, high-active
	Data type	BOOL
ibNegLimitSwitch	Description	Negative limit switch, high-active
	Data type	BOOL
iTimeout	Description	Time monitoring (T#0ms: off)
	Data type	TIME

Output

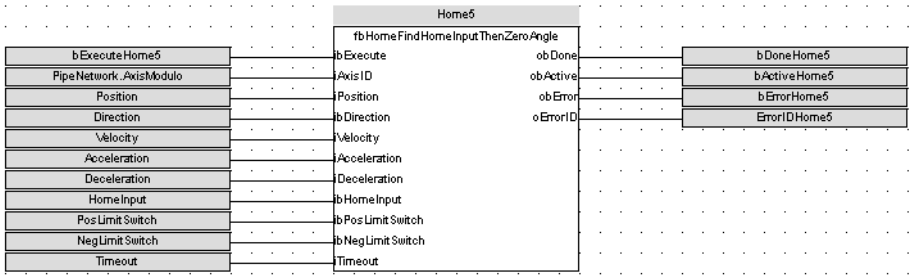
obDone	Description	Done bit
	Data type	BOOL
obActive	Description	Active bit
	Data type	BOOL
obError	Description	Error bit
	Data type	BOOL
oErrorID	Description	Error identifier, see list here

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

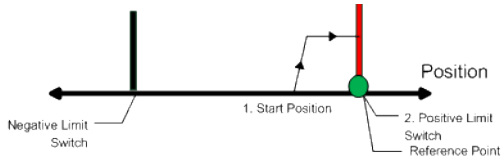
Example

Function Block Diagram



5.2.6.6 MLFB_HomeFindLimitInput

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.

Arguments

Input

ibExecute	Description	Start homing, edge-triggered
	Data type	BOOL
iAxisID	Description	ID of Axis block of Pipe Network

	Data type	DINT
iPosition	Description	Reference position
	Data type	LREAL
ibDirection	Description	0=positive, 1=negative
	Data type	BOOL
iVelocity	Description	Reference speed
	Data type	LREAL
iAcceleration	Description	Reference acceleration
	Data type	LREAL
iDeceleration	Description	Reference deceleration
	Data type	LREAL
ibLimitSwitch	Description	Pos. or neg. limit switch, high-active (depends on ibDirection)
	Data type	BOOL
iTimeout	Description	Time monitoring (T#0ms: off)
	Data type	TIME

Output

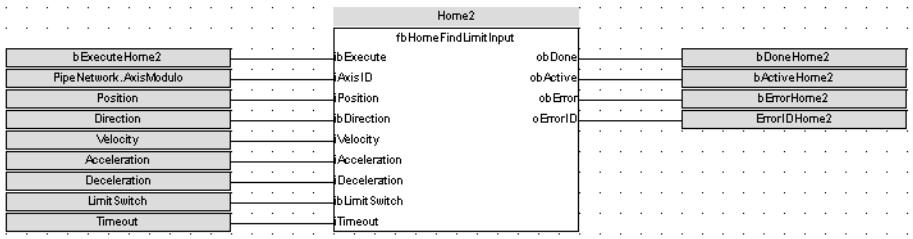
obDone	Description	Done bit
	Data type	BOOL
obActive	Description	Active bit
	Data type	BOOL
obError	Description	Error bit
	Data type	BOOL
oErrorID	Description	Error identifier, see list here

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

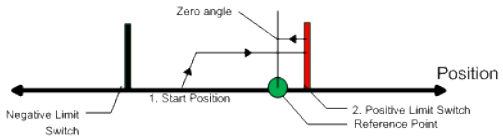
Example

Function Block Diagram



5.2.6.7 MLFB_HomeFindLimitInputThenZeroAngle

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.

Arguments

Input

ibExecute	Description	Start homing, edge-triggered
	Data type	BOOL
iAxisID	Description	ID of Axis block of Pipe Network
	Data type	BOOL
iPosition	Description	Reference position
	Data type	BOOL
ibDirection	Description	0=positive, 1=negative
	Data type	BOOL
iVelocity	Description	Reference speed
	Data type	BOOL
iAcceleration	Description	Reference acceleration
	Data type	BOOL
iDeceleration	Description	Reference deceleration
	Data type	BOOL
ibLimitSwitch	Description	Pos. or neg. limit switch, high-active (depends on ibDirection)
	Data type	BOOL
iTimeout	Description	Time monitoring (T#0ms: off)
	Data type	BOOL

Output

obDone	Description	Done bit
	Data type	BOOL
obActive	Description	Active bit
	Data type	BOOL
obError	Description	Error bit
	Data type	BOOL

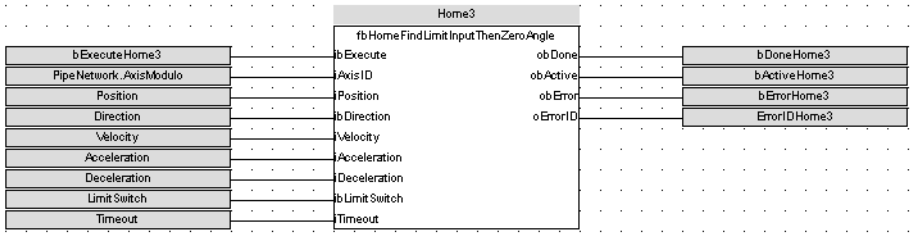
oErrorID	Description	Error identifier, see list here

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

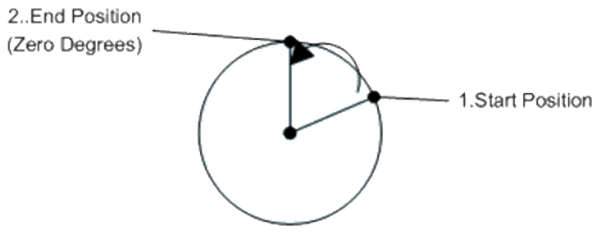
Example

Function Block Diagram



5.2.6.8 MLFB_HomeFindZeroAngle

Description



Mode to find the zero angle reference of the motor.

Arguments

Input

ibExecute	Description	Start homing, edge-triggered
	Data type	BOOL
iAxisID	Description	ID of Axis block of Pipe Network
	Data type	DINT
iPosition	Description	Reference position
	Data type	LREAL
iDirectionType	Description	0=positive, 1=negative, 2=shortest
	Data type	DINT
iVelocity	Description	Reference speed
	Data type	LREAL

iAcceleration	Description	Reference acceleration
	Data type	LREAL
iDeceleration	Description	Reference deceleration
	Data type	LREAL
iTimeout	Description	Time monitoring (T#0ms: off)
	Data type	TIME

Output

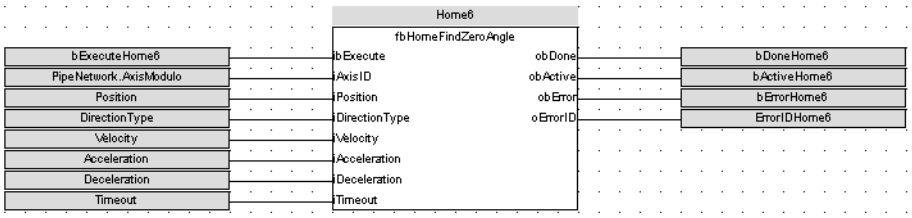
obDone	Description	Done bit
	Data type	BOOL
obActive	Description	Active bit
	Data type	BOOL
obError	Description	Error bit
	Data type	BOOL
oErrorID	Description	Error identifier, see list here

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

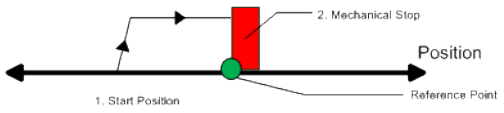
Example

Function Block Diagram



5.2.6.9 MLFB_HomeMoveUntilPosErrExceeded

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.

Arguments

Input

ibExecute	Description	Start homing, edge-triggered
	Data type	BOOL
iAxisID	Description	ID of Axis block of Pipe Network
	Data type	DINT
iPosition	Description	Reference position
	Data type	LREAL
ibDirection	Description	0=positive, 1=negative
	Data type	BOOL
iVelocity	Description	Reference speed
	Data type	LREAL
iAcceleration	Description	Reference acceleration
	Data type	LREAL
iDeceleration	Description	Reference deceleration
	Data type	LREAL
iMaxPositionError	Description	Maximum position error
	Data type	LREAL
iPeakCurrent	Description	Peak current in mA
	Data type	DINT
iTimeout	Description	Time monitoring (T#0ms: off)
	Data type	TIME

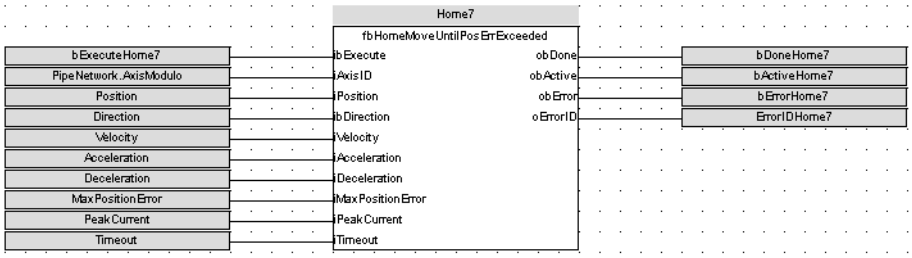
Output

obDone	Description	Done bit
	Data type	BOOL
obActive	Description	Active bit
	Data type	BOOL
obError	Description	Error bit
	Data type	BOOL
oErrorID	Description	Error identifier, see list here
	Data type	DINT

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

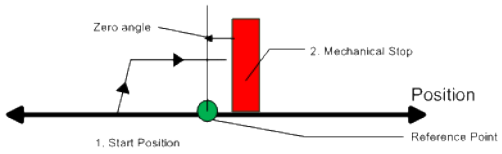
Example

Function Block Diagram



5.2.6.10 MLFB_HomeMoveUntilPosErrExceededThenZeroAngle

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.

Arguments

Input

ibExecute	Description	Start homing, edge-triggered
	Data type	BOOL
iAxisID	Description	ID of Axis block of Pipe Network
	Data type	DINT
iPosition	Description	Reference position
	Data type	LREAL
ibDirection	Description	0=positive, 1=negative
	Data type	BOOL
iVelocity	Description	Reference speed
	Data type	LREAL
iAcceleration	Description	Reference acceleration
	Data type	LREAL
iDeceleration	Description	Reference deceleration
	Data type	LREAL
iMaxPositionError	Description	Maximum position error
	Data type	LREAL
iPeakCurrent	Description	Peak current in mA
	Data type	DINT
iTimeout	Description	Time monitoring (T#0ms: off)
	Data type	TIME

Output

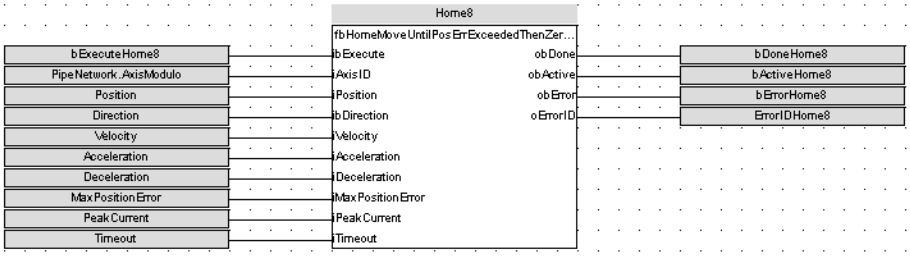
obDone	Description	Done bit
	Data type	BOOL
obActive	Description	Active bit
	Data type	BOOL
obError	Description	Error bit
	Data type	BOOL
oErrorID	Description	Error identifier, see list here

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

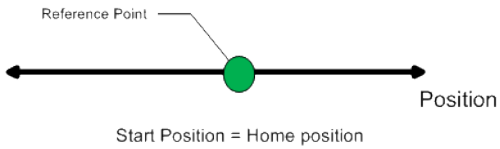
Example

Function Block Diagram



5.2.6.11 MLFB_HomeUsingCurrentPosition

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 then zero. This allows you to offset your home reference away from zero.

Arguments

Input

ibExecute	Description	Start homing, edge-triggered
	Data type	BOOL

iAxisID	Description	ID of Axis block of Pipe Network
	Data type	DINT
iPosition	Description	Reference position
	Data type	LREAL

Output

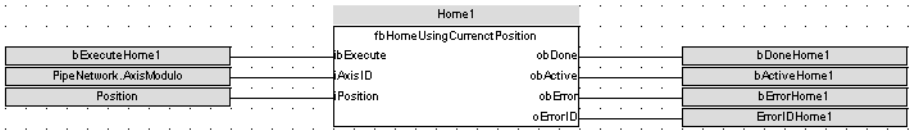
obDone	Description	Done bit
	Data type	BOOL
obActive	Description	Active bit
	Data type	BOOL
obError	Description	Error bit
	Data type	BOOL
oErrorID	Description	Error identifier, see list here

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

Example

Function Block Diagram



5.2.6.12 MLFB_HomeFindHomeFastInput

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:

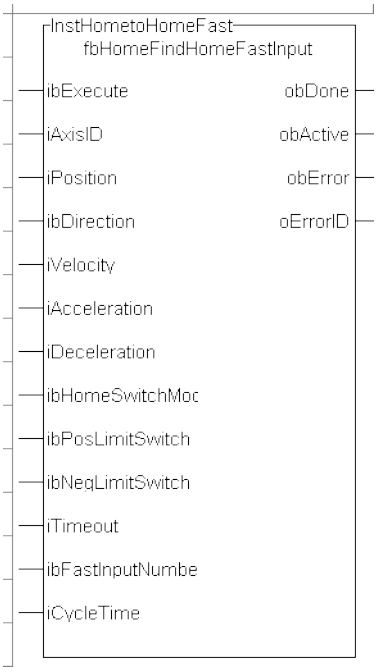


Figure 1-133: MLFB HomeFindHomeFastInput

Arguments

Input

ibExecute	Description	Request the homing step procedure at rising edge
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
iAxisID	Description	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	[1 , 256]
	Unit	n/a
	Default	—
iPosition	Description	Offset Position Applied After Home Switch is found
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
ibDirection	Description	Define the axis homing direction
	Data type	BOOL
	Range	[0 , 1]

Value	Description
0	clockwise rotation
1	counterclockwise rotation

	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^2
	Default	—
iDeceleration	Description	Commanded deceleration for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
ibHomeSwitchMode	Description	Limit switch state to complete homing
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
ibPosLimitSwitch	Description	The positive direction limit switch input I/O point
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
ibNegLimitSwitch	Description	The negative direction limit switch input I/O point
	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	—

Value	Description
0	Rising edge of switch
1	Falling edge of switch

		Limit switch state to complete homing.
ibFastInputNumber	Description	Value: 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	—

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
oErrorID	Description	Indicates the error if Error output is set to TRUE Value: 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

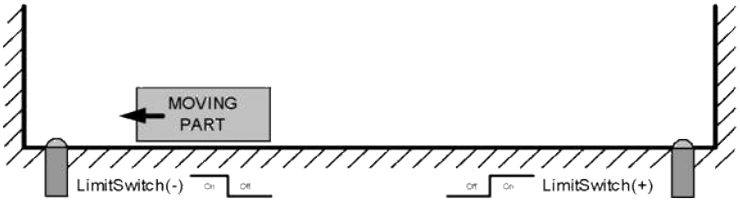
Usage

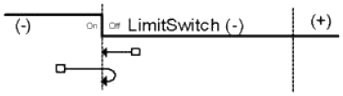
This homing procedure performs a homing function searching for sensor using only High Speed Input Switches. (A High Speed Limit Switch has 1 “Off” (or “On”) area).

- 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





Related Functions

MLFB_HomeFindHomeFastInputModulo

MLFB_HomeFindLimitFastInput

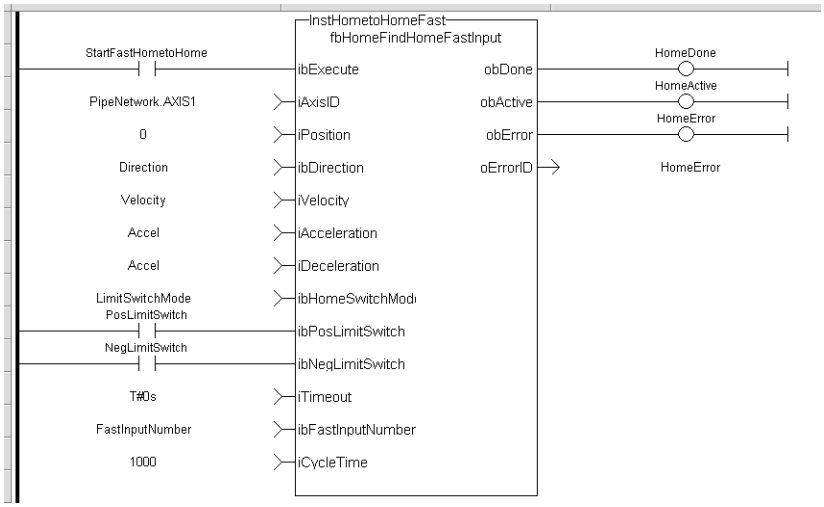
MLFB_HomeFindLimitFastInputModulo

Example

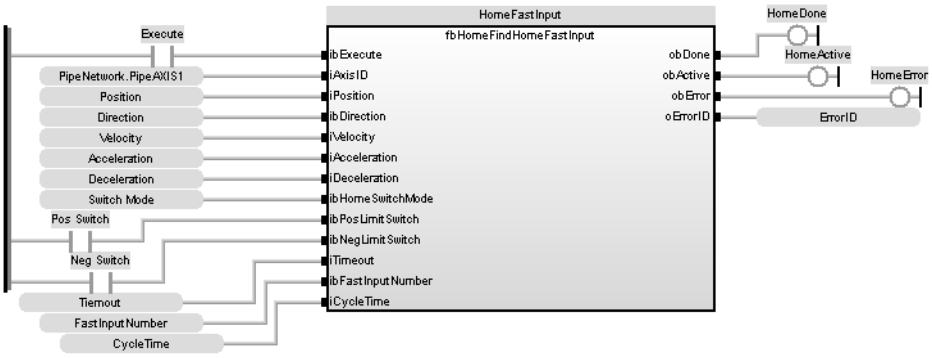
Structured Text

```
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, Velo-
city, 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 *)
```

Ladder Diagram



Function Block Diagram



5.2.6.13 MLFB_HomeFindHomeFastInputModulo

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:

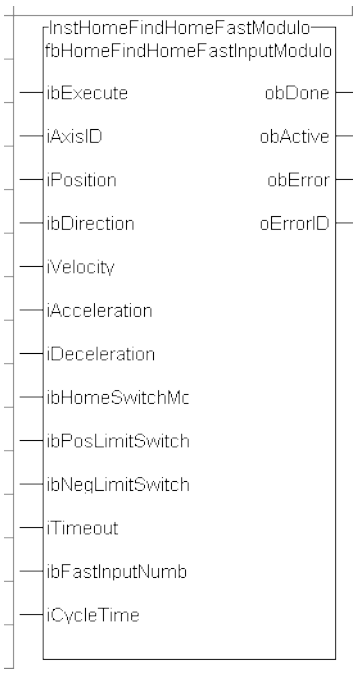


Figure 1-134: MLFB HomeFindHomeFastInputModulo

Arguments

Input

ibExecute	Description	Request the homing step procedure at rising edge
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
iAxisID	Description	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	[1 , 256]
	Unit	n/a
	Default	—
iPosition	Description	Offset Position Applied After Home Switch is found
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
ibDirection	Description	Define the axis homing direction Value: 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^2
	Default	—
iDeceleration	Description	Commanded deceleration for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—

ibLimitSwitchMode	Description	Limit switch state to complete homing Value: 0 — Rising edge of switch; 1 — Falling edge of switch
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
ibPosLimitSwitch	Description	The positive direction limit switch input I/O point
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
ibNegLimitSwitch	Description	The negative direction limit switch input I/O point
	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	—
ibFastInputNumber	Description	Limit switch state to complete homing. Value: 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	—

Output

obDone	Description	Indicates the move completed successfully. The Command Position has reached the endpoint
	Data type	BOOL

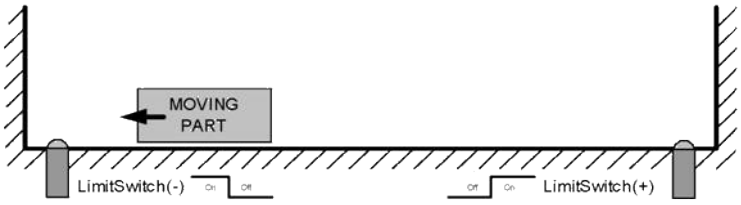
	Unit	n/a
obActive	Description	Indicates this move is the active move
	Data type	BOOL
	Unit	n/a
obError	Description	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
	Unit	n/a
oErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	DINT
	Unit	n/a

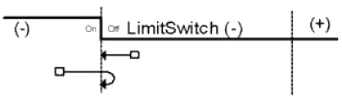
Value	Description
1	Axis in Error State
2	Axis is Not Enabled
3	Timeout Exceeded
4	SDO Read/Write Error
5	Input Parameter out of Range

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





Related Functions

MLFB_HomeFindHomeFastInput

MLFB_HomeFindLimitFastInput

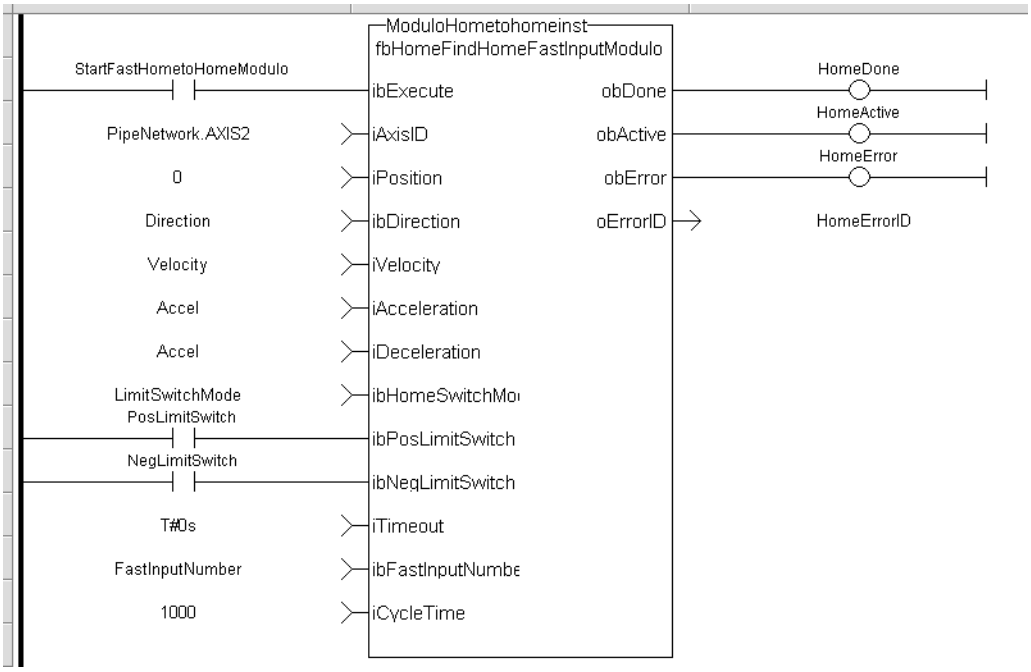
MLFB_HomeFindLimitFastInputModulo

Example

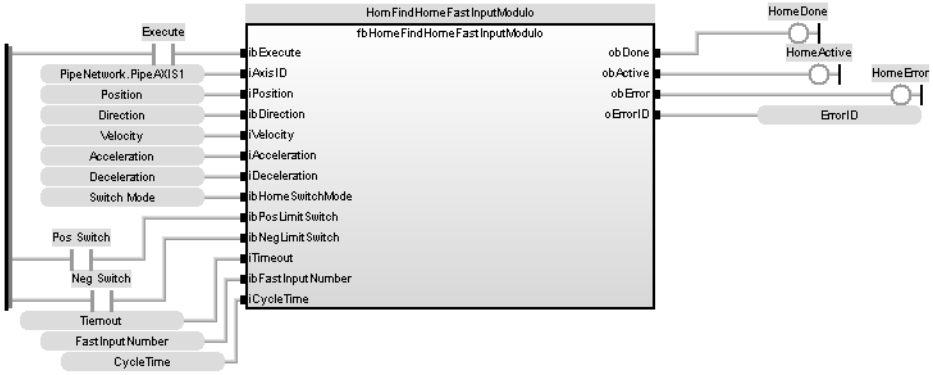
Structured Text

```
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 *)
```

Ladder Diagram



Function Block Diagram



5.2.6.14 MLFB_HomeFindLimitFastInput

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:

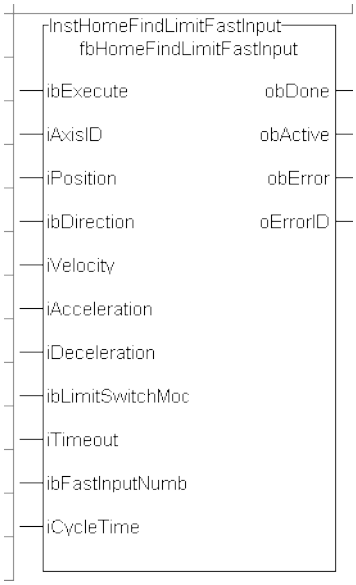


Figure 1-135: MLFB HomeFindLimitFastInput

Arguments

Input

ibExecute	Description	Request the homing step procedure at rising edge
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
iAxisID	Description	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	[1 , 256]
	Unit	n/a
	Default	—
iPosition	Description	Offset Position Applied After Home Switch is found
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
ibDirection	Description	Define the axis homing direction
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
iVelocity	Description	Commanded velocity for the homing move

Value	Description
0	clockwise rotation
1	counterclockwise rotation

	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^2
	Default	—
iDeceleration	Description	Commanded deceleration for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
ibLimitSwitchMode	Description	Limit switch state to complete homing

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	—
ibFastInputNumber	Description	Limit switch state to complete homing.

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	—

Output

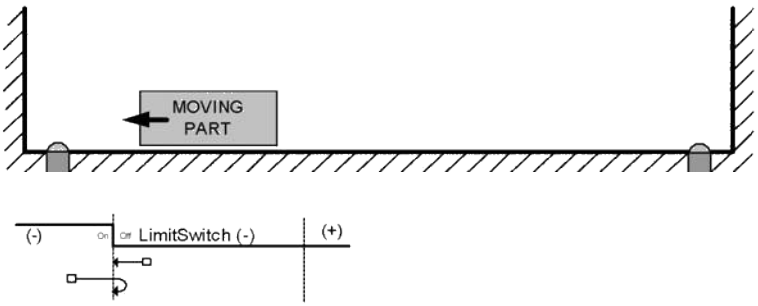
obDone	Description	Indicates the move completed successfully. The Command Position has reached the endpoint
	Data type	BOOL
	Unit	n/a
obActive	Description	Indicates this move is the active move
	Data type	BOOL
	Unit	n/a
obError	Description	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
	Unit	n/a
oErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	DINT
	Unit	n/a

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

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



Related Functions

MLFB_HomeFindHomeFastInput

MLFB_HomeFindHomeFastInputModulo

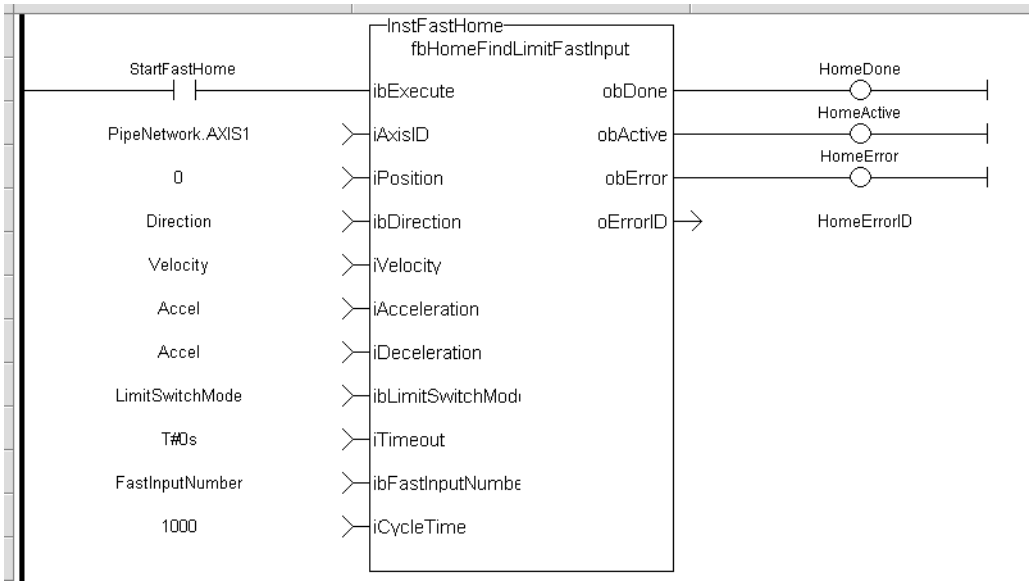
MLFB_HomeFindLimitFastInputModulo

Example

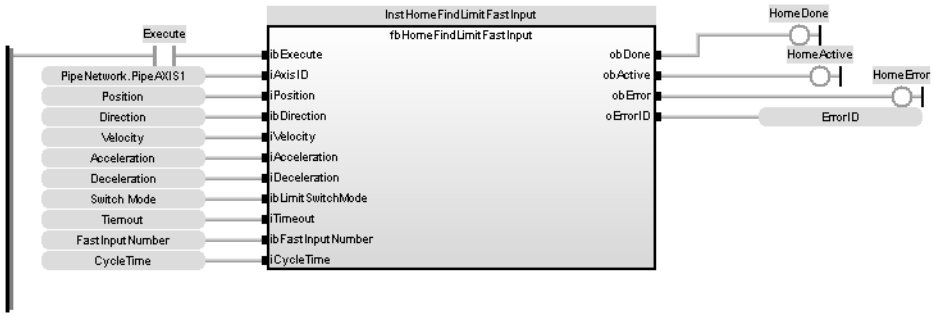
Structured Text

```
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, Velo-
city, Acceleration, Deceleration, LimitSwitchMode, Timeout, FastIn-
putNumber, CycleTime);

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

Ladder Diagram



Function Block Diagram



5.2.6.15 MLFB_HomeFindLimitFastInputModulo

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:

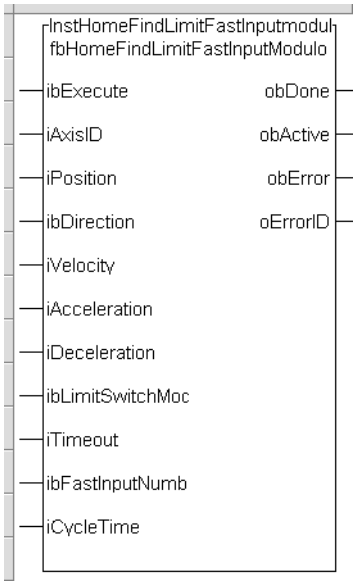


Figure 1-136: MLFB HomeFindLimitFastInputModulo

Arguments

Input

ibExecute	Description	Request the homing step procedure at rising edge
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—

iAxisID	Description	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	[1 , 256]
	Unit	n/a
	Default	—
iPosition	Description	Offset Position Applied After Home Switch is found
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
ibDirection	Description	Define the axis homing direction

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^2
	Default	—
iDeceleration	Description	Commanded deceleration for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
ibLimitSwitchMode	Description	Limit switch state to complete homing

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	—
ibFastInputNumber	Description	Limit switch state to complete homing.

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	—

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
oErrorID	Description	Indicates the error if Error output is set to TRUE

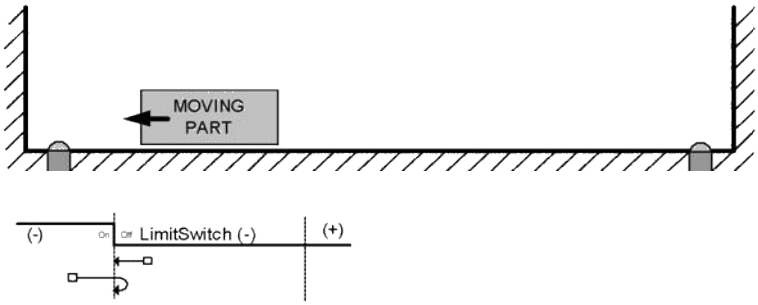
Value	Description
1	Axis in Error State
2	Axis is Not Enabled
3	Timeout Exceeded
4	SDO Read/Write Error
5	Input Parameter out of Range

	Data type	DINT
	Unit	n/a

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



Related Functions

MLFB_HomeFindHomeFastInput

MLFB_HomeFindHomeFastInputModulo

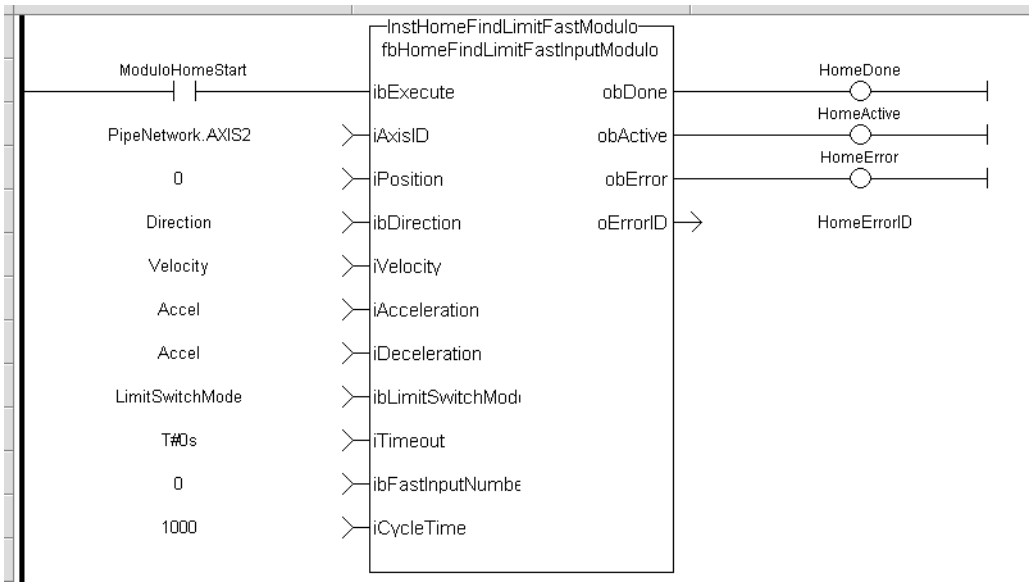
MLFB_HomeFindLimitFastInput

Example

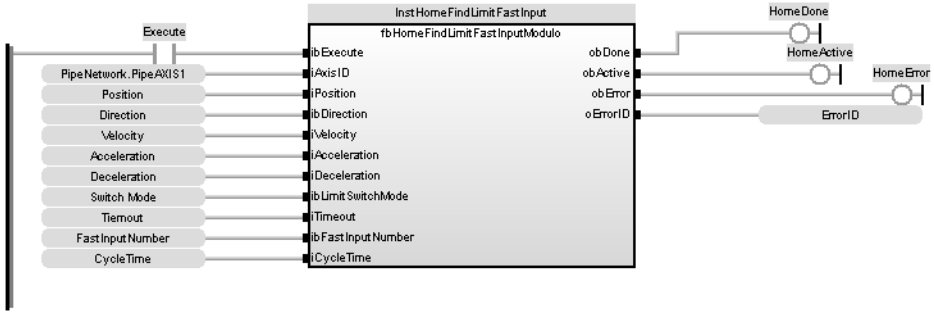
Structured Text

```
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, FastIn-
putNumber, CycleTime);

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

Ladder Diagram



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

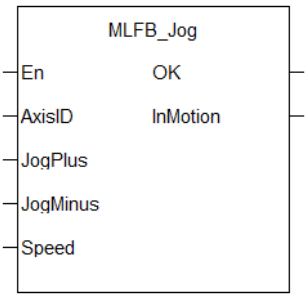


Figure 1-137: Kollmorgen UDFB Jog for PipeNetwork

5.2.7.2 Arguments

Input

En	**Description**	Enables execution (FFLD only)
	Data type	BOOL
	Range	—
	Unit	n/a
	Default	—
AxisID	**Description**	ID Name of the Axis
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—
JogPlus	**Description**	Enables a Jog in the plus direction
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
JogMinus	**Description**	Enables a Jog in the Minus direction
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
Speed	**Description**	Rate at which the axis will move
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—

Output

InMotion	**Description**	Jogging is active when TRUE
	Data type	BOOL
	Range	n/a

5.2.7.3 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.

5.2.7.4 Related Functions

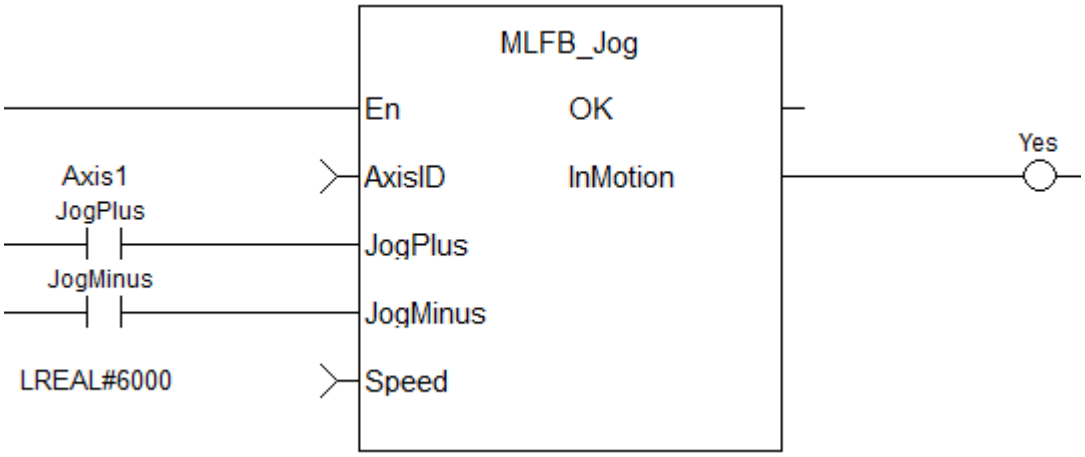
MLAxisMoveVel

5.2.7.5 Example

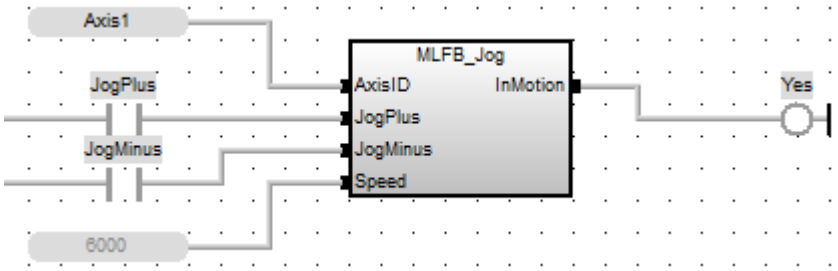
Structured Text

```
Jog_PipeNetwork(Axis1(DINT), JogPlus(BOOL), JogMinus(BOOL), 6000(LREAL));
```

Ladder Diagram



Function Block Diagram



5.2.7.6 MLFB_PlsPosFw

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.

Arguments

Input

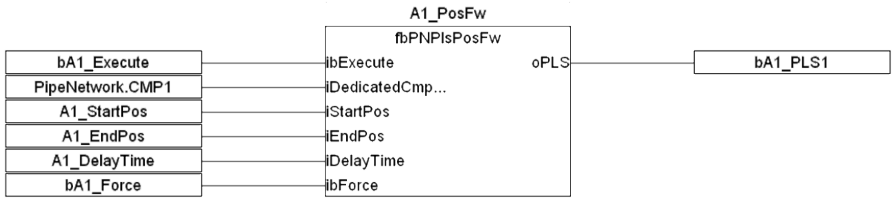
ibExecute	Description	Enable PLS
	Data type	BOOL
iDedicatedCmpID	Description	ID of dedicated comparator
	Data type	DINT
iStartPos	Description	Start position of PLS
	Data type	LREAL
iEndPos	Description	End position of PLS
	Data type	LREAL
iDelayTime	Description	Delay time for compensation
	Data type	TIME
ibForce	Description	Force PLS
	Data type	BOOL

Output

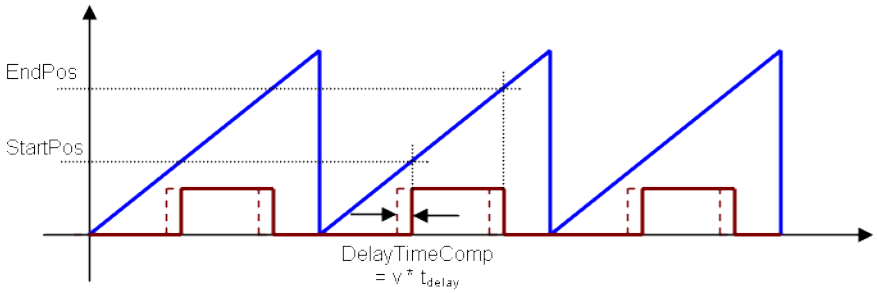
oPLS	Description	Position limit switch
	Data type	BOOL

Example

Function Block Diagram



Timing



5.2.7.7 MLFB_PlsPosFwBw

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.

Arguments

Input

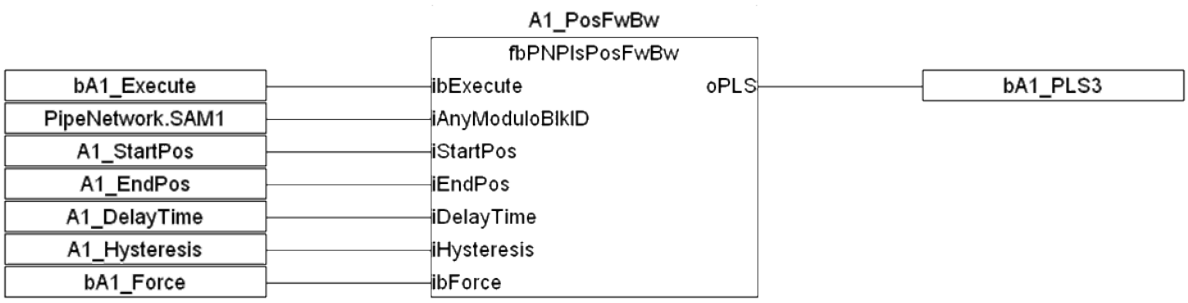
ibExecute	Description	Enable PLS
	Data type	BOOL
iAnyModuloBlkID	Description	Any modulo pipe network block ID
	Data type	DINT
iStartPos	Description	Start position of PLS
	Data type	LREAL
iEndPos	Description	End position of PLS
	Data type	LREAL
iDelayTime	Description	Delay time for compensation
	Data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

Output

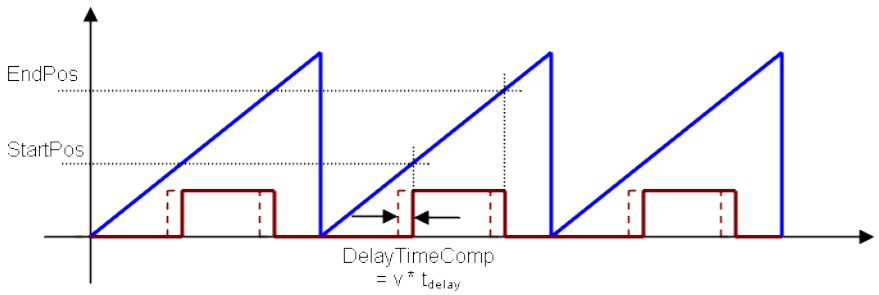
oPLS	Description	Position limit switch
	Data type	BOOL

Example

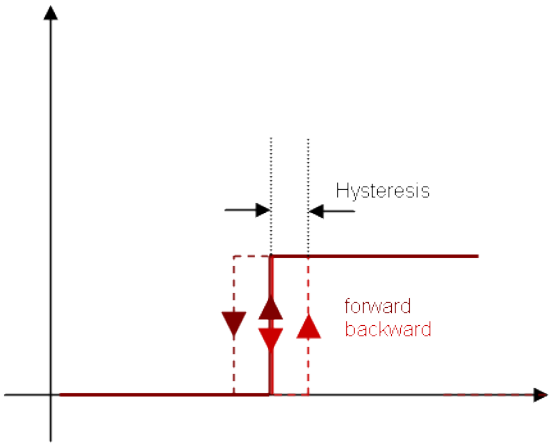
Function Block Diagram



Timing



Hysteresis



5.2.7.8 MLFB_PlsTimeFw

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.

Arguments

Input

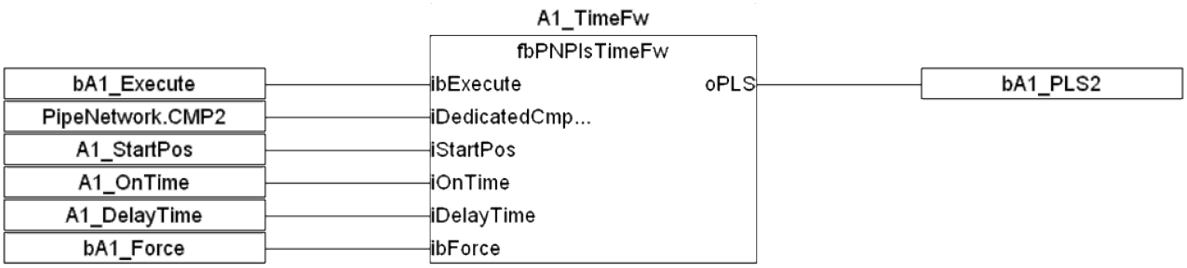
ibExecute	Description	Enable PLS
	Data type	BOOL
iDedicatedCmpID	Description	ID of dedicated comparator
	Data type	DINT
iStartPos	Description	Start position of PLS
	Data type	LREAL
iOnTime	Description	Time PLS is on
	Data type	TIME
iDelayTime	Description	Delay time for compensation
	Data type	TIME
ibForce	Description	Force PLS
	Data type	BOOL

Output

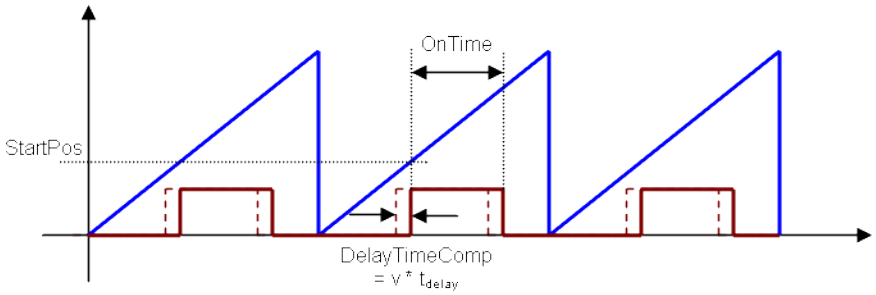
oPLS	Description	Position limit switch
	Data type	BOOL

Example

Function Block Diagram



Timing



5.2.7.9 MCFB_StepAbsolute

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:

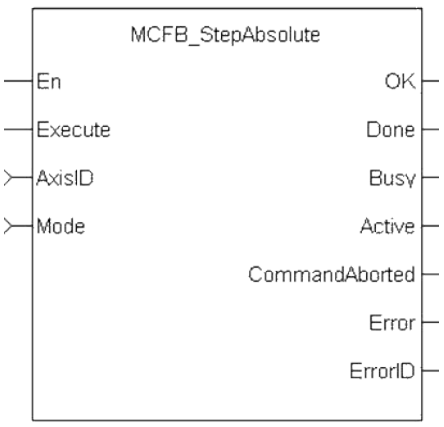


Figure 1-138: MCFB StepAbsolute

Arguments

Input

En	**Description**	Enables execution (FFLD only)
	Data type	BOOL

	Range	—
	Unit	n/a
	Default	—
Execute	**Description**	Request the homing step procedure at rising edge
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
AxisID	**Description**	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	[1 , 256]
	Unit	n/a
	Default	—
Mode	**Description**	Define the actual position assignment source

Value	Description
0	use drive feedback position for actual position
1	use once per rev feedback position

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

Output

Done	**Description**	Indicates the move completed successfully. The Command Position has reached the endpoint
	Data type	BOOL
	Unit	n/a
Busy	**Description**	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
	Unit	n/a
Active	**Description**	Indicates this move is the active move
	Data type	BOOL
	Unit	n/a
CommandAborted	**Description**	Indicates the move was aborted
	Data type	BOOL
	Unit	n/a
Error	**Description**	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
	Unit	n/a
Error	**Description**	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL

	Unit	n/a
ErrorID	**Description**	Indicates the error if Error output is set to TRUE

Value	Description
1	Desired SetPosition is outside of Rollover period

	Data type	INT
	Unit	n/a

Related Functions

MCFB_StepAbsSwitch

MCFB_StepRefPulse

MCFB_StepBlock

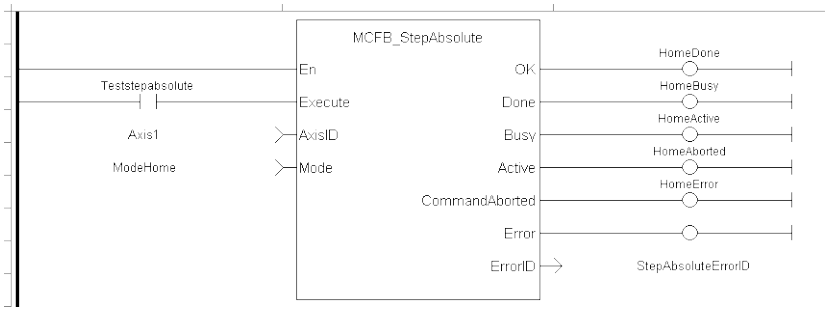
MCFB_StepLimitSwitch

Example

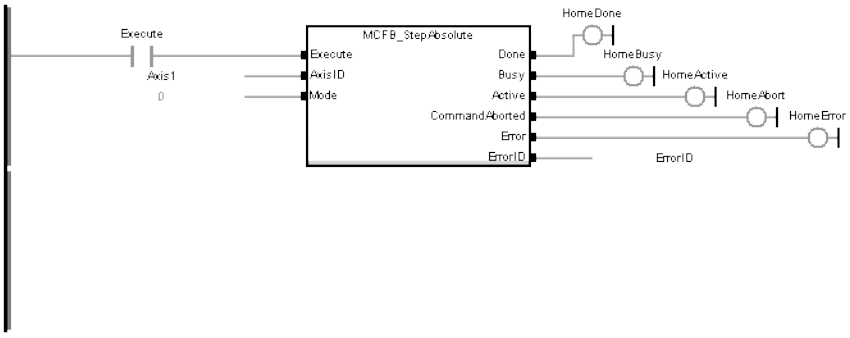
Structured Text

```
MCFB_StepAbsolute( True, Axis1, 0 );
```

Ladder Diagram



Function Block Diagram



5.2.7.10 MCFB_StepAbsSwitch

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:

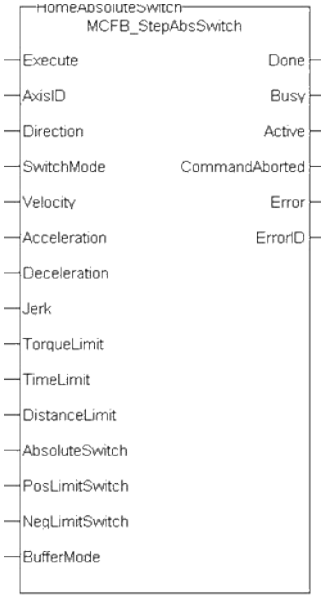


Figure 1-139: MCFB StepAbsSwitch

Arguments

Input

Execute

Description	Request the homing step procedure at rising edge. Outputs are reset when execute input is false.
Data type	BOOL
Range	[0 , 1]
Unit	n/a
Default	—

AxisID

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

Direction

Description	Define the axis homing direction

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

Data type	DINT
Range	[0 , 3]
Unit	n/a
Default	—

SwitchMode

Description Switch state to complete homing

Value	Description
0	switch is on
1	switch if off
2	rising edge of switch
3	falling edge of switch

Data type DINT

Range [0 , 3]

Unit n/a

Default —

Velocity

Description Commanded velocity for the homing move

Data type LREAL

Range —

Unit User unit/sec

Default —

Acceleration

Description Commanded acceleration for the homing move

Data type LREAL

Range —

Unit User unit/sec^2

Default —

Deceleration

Description Commanded deceleration for the homing move

Data type LREAL

Range —

Unit User unit/sec^2

Default —

Jerk

Description Commanded jerk for the homing move (if zero, then trapezoidal acc/dec is used)

Data type LREAL

Range —

Unit User unit/sec^3

Default —

TorqueLimit

Description Maximum torque applied for the homing move entered in thousandths of maximum torque, e.g. "250" is 250/1000, or 25%.

Data type LREAL

Range —

Unit User unit

Default —

TimeLimit

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 —

DistanceLimit	**Description**	Maximum distance for homing move to complete. If exceeded the homing procedure will error out. 0= no distance limit
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
AbsoluteSwitch	**Description**	The absolute switch input I/O point
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
PosLimitSwitch	**Description**	The positive direction limit switch input I/O point
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
NegLimitSwitch	**Description**	The negative direction limit switch input I/O point
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
SwitchMode	**Description**	Switch state to complete homing

Value	Description
0	abort
1	buffer
2	Blend to active
3	blend to next
4	blend to low velocity
5	blend to high velocity

	Data type	SINT
	Range	[0 , 5]
	Unit	n/a
	Default	—

Output

Done	**Description**	Indicates the move completed successfully. The Command Position has reached the endpoint
	Data type	BOOL
	Unit	n/a
Busy	**Description**	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
	Unit	n/a

Active	**Description**	Indicates this move is the active move
	Data type	BOOL
	Unit	n/a
CommandAborted	**Description**	Indicates the move was aborted
	Data type	BOOL
	Unit	n/a
Error	**Description**	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
	Unit	n/a
ErrorID	**Description**	Indicates the error if Error output is set to TRUE

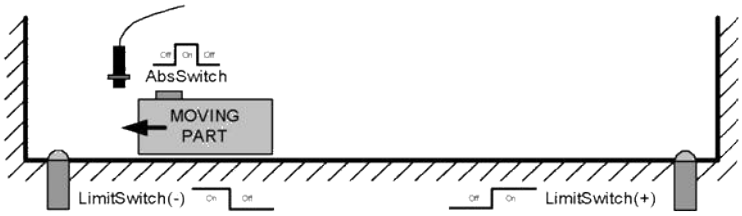
Value	Description
1	TimeLimit exceeded
2	DistanceLimit exceeded
3	TorqueLimit exceeded
4	axis error stop state
5	axis not enabled
6	invalid inputs for Velocity-Acceleration-Deceleration

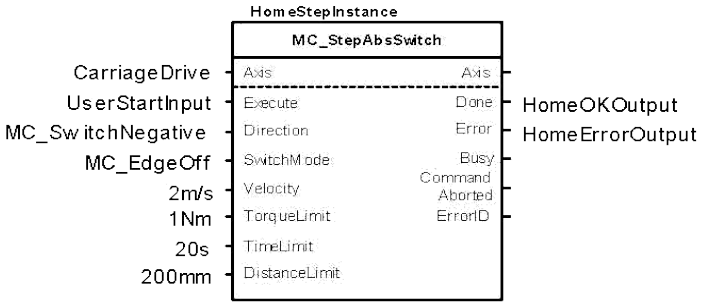
Data type	INT
Unit	n/a

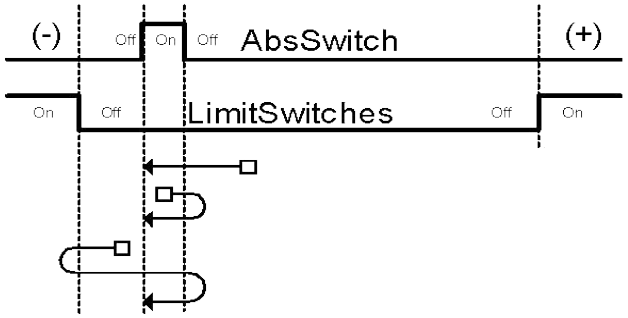
Usage

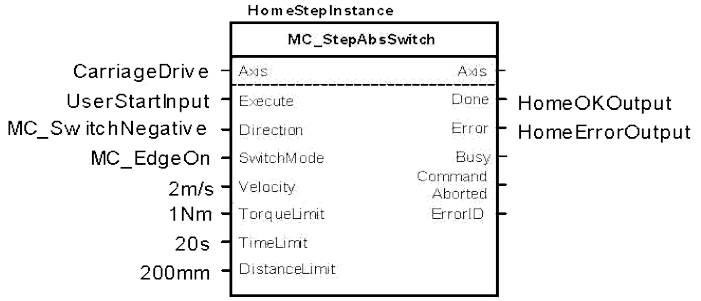
This physical layout has the risk that homing is started in the wrong direction (escaping the switch). To support such case, it implements a special behavior when Limit Switches are found (or the AbsSwitch itself is “On” at Execute):

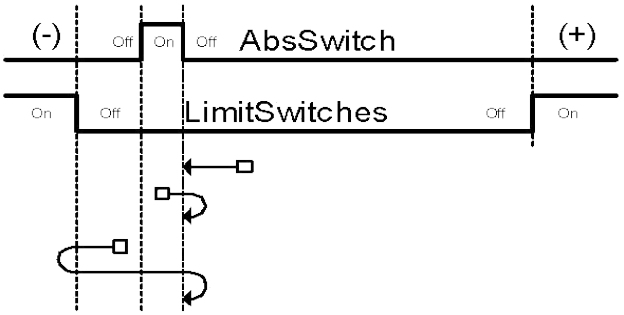
- The homing is commanded in the most likely direction were the sensor can be found. In this example (-).
- The velocity is defined by the input.
- The torque is limited.
- Both Time and Distance Limits can cause an error if exceeded
- If any LimitSwitch is found during Homing (any of them), then a special 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



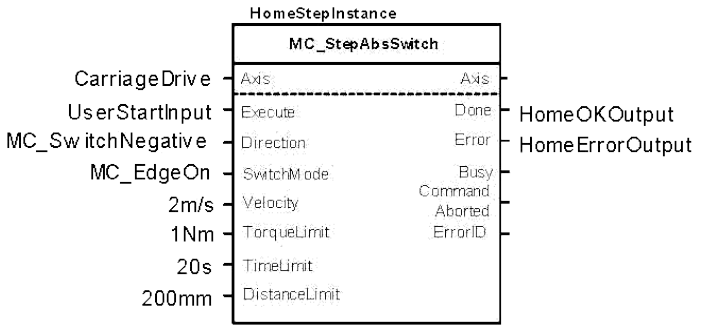


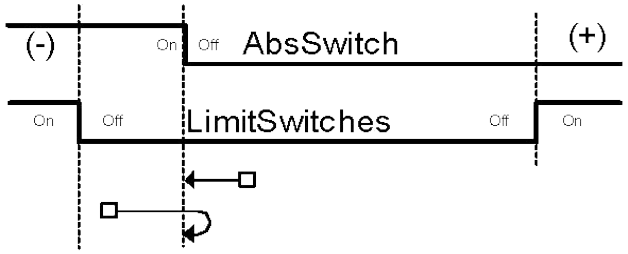




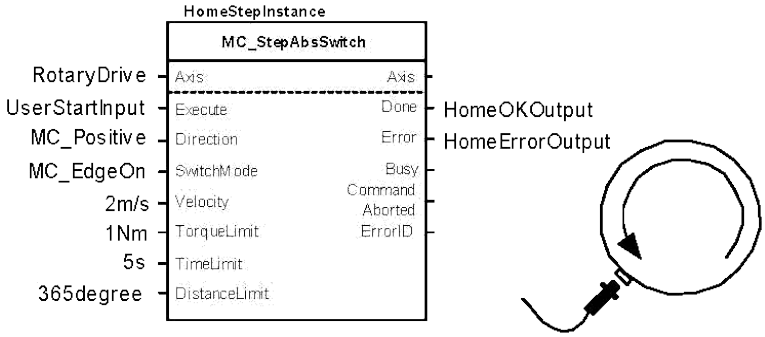


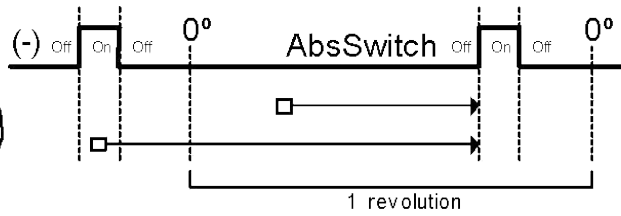
An overlapping switch configuration is also possible. This has same the behavior as working on the limit switches:





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):





Related Functions

MCFB_StepAbsolute

MCFB_StepRefPulse

MCFB_StepBlock

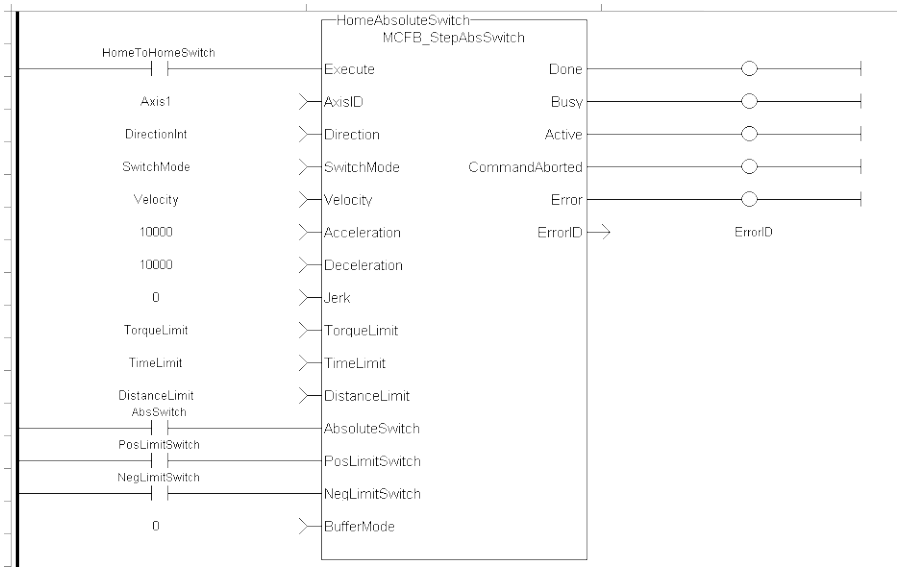
MCFB_StepLimitSwitch

Example

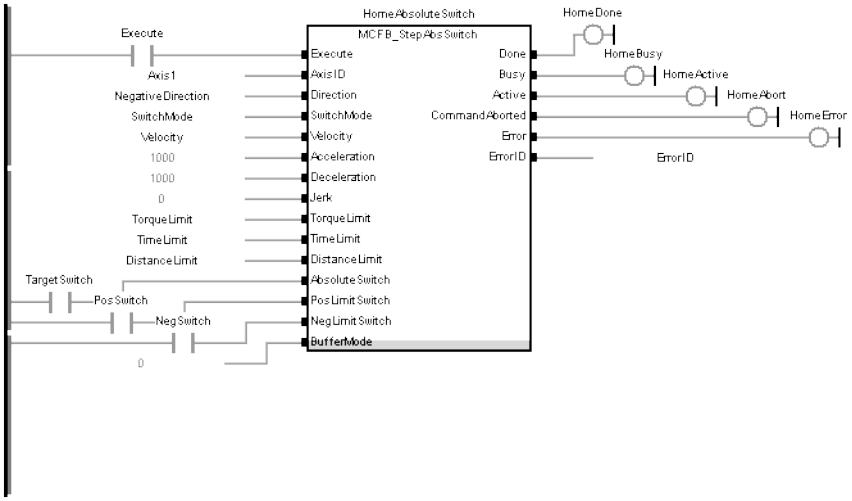
Structured Text

```
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, Abso-
luteSwitch, 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 *)
```

Ladder Diagram



Function Block Diagram



5.2.7.11 MCFB_StepBlock

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:

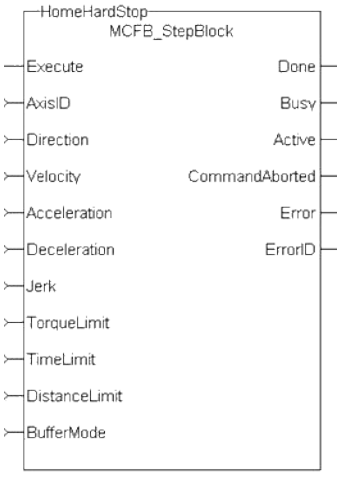


Figure 1-140: MCFB StepBlock

Arguments

Input

Execute	Description	Request the homing step procedure at rising edge. Outputs are reset when execute input is false.
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—

AxisID	Description	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	[1 , 256]
	Unit	n/a
	Default	—
Direction	Description	Define the axis homing direction Value: 0 — clockwise rotation; 1 — counterclockwise rotation
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
Velocity	Description	Commanded velocity for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	Description	Commanded acceleration for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Deceleration	Description	Commanded deceleration for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Jerk	Description	Commanded jerk for the homing move (if zero, then trapezoidal acc/dec is used)
	Data type	LREAL
	Range	—
	Unit	User unit/sec^3
	Default	—
TorqueLimit	Description	Maximum torque applied for the homing move entered in thousandths of maximum torque, e.g. "250" is 250/1000, or 25%.
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
TimeLimit	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	—
DistanceLimit	Description	Maximum distance for homing move to complete. If exceeded the homing procedure will error out. 0= no distance limit
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
BufferMode	Description	Define the homing move start action 0 abort 1 buffer 2 Blend to active 3 blend to next 4 blend to low velocity 5 blend to high velocity
	Data type	SINT
	Range	[0 , 5]
	Unit	n/a
	Default	—

Output

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

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

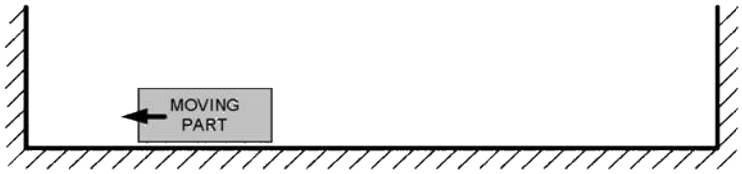
Value	Description
1	TimeLimit exceeded
2	DistanceLimit exceeded
3	
4	axis error stop state
5	axis not enabled
6	invalid inputs for Velocity-Acceleration-Deceleration

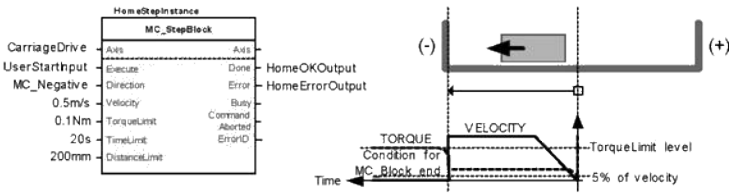
Data type	INT
Unit	n/a

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





Related Functions

MCFB_StepAbsolute

MCFB_StepRefPulse

MCFB_StepAbsSwitch

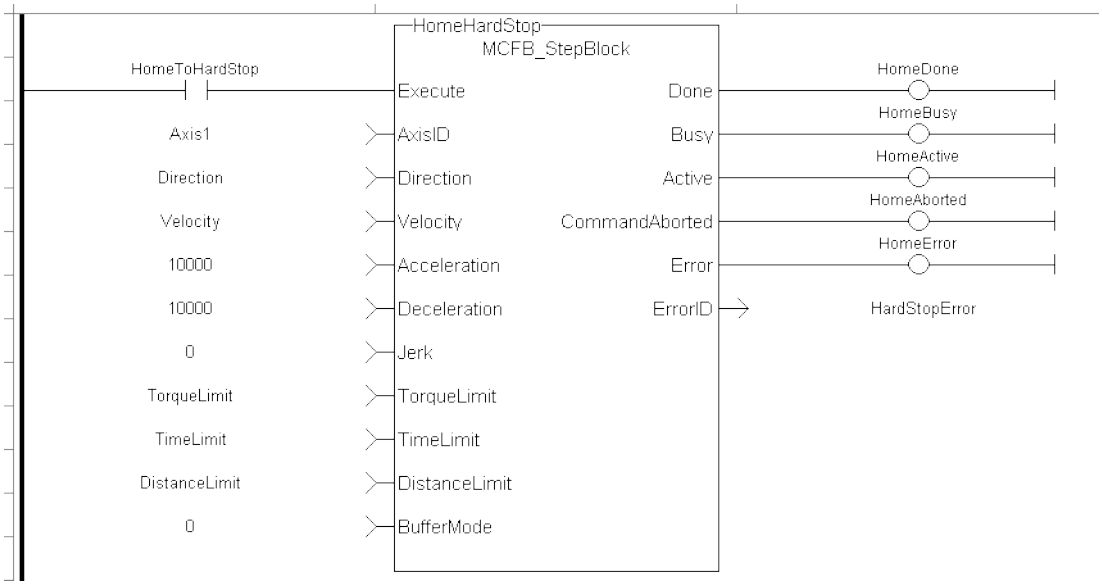
MCFB_StepLimitSwitch

Example

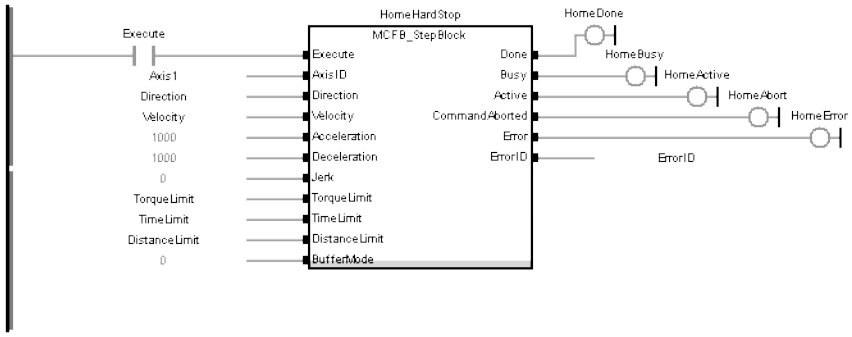
Structured Text

```
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;
```

Ladder Diagram



Function Block Diagram



5.2.7.12 MCFB_StepLimitSwitch

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:

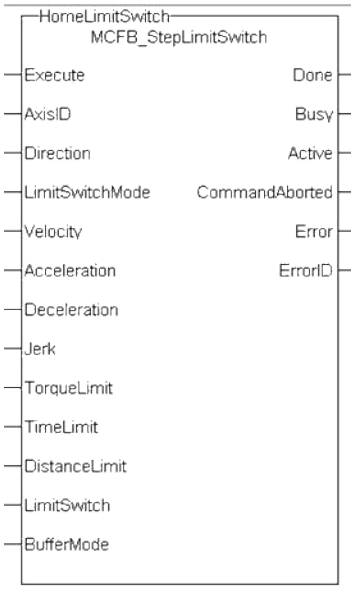


Figure 1-141: MCFB StepLimitSwitch

Arguments

Input

Execute	Description	Request the homing step procedure at rising edge. Outputs are reset when execute input is false.
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
AxisID	Description	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	[1 , 256]
	Unit	n/a

	Default	—
Direction	Description	Define the axis homing direction Value: 0 — clockwise rotation; 1 — counterclockwise rotation
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
LimitSwitchMode	Description	Limit switch state to complete homing Value: 0 — switch is on; 1 — switch if off; 2 — rising edge of switch; 3 — falling edge of switch
	Data type	DINT
	Range	[0 , 3]
	Unit	n/a
	Default	—
Velocity	Description	Commanded velocity for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	Description	Commanded acceleration for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Deceleration	Description	Commanded deceleration for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Jerk	Description	Commanded jerk for the homing move (if zero, then trapezoidal acc/dec is used)
	Data type	LREAL
	Range	—
	Unit	User unit/sec^3
	Default	—
TorqueLimit	Description	Maximum torque applied for the homing move entered in thousandths of maximum torque, e.g. "250" is 250/1000, or 25%.
	Data type	LREAL

	Range	—
	Unit	User unit
	Default	—
TimeLimit	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	—
DistanceLimit	Description	Maximum distance for homing move to complete. If exceeded the homing procedure will error out. 0= no distance limit
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
LimitSwitch	Description	The limit switch input I/O point
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
BufferMode	Description	Define the homing move start action Value: 0 abort; 1 buffer; 2 Blend to active; 3 blend to next; 4 blend to low velocity; 5 blend to high velocity
	Data type	SINT
	Range	[0 , 5]
	Unit	n/a
	Default	—

Output

Done	Description	Indicates the move completed successfully. The Command Position has reached the endpoint
	Data type	BOOL
	Unit	n/a
Busy	Description	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
	Unit	n/a
Active	Description	Indicates this move is the active move
	Data type	BOOL

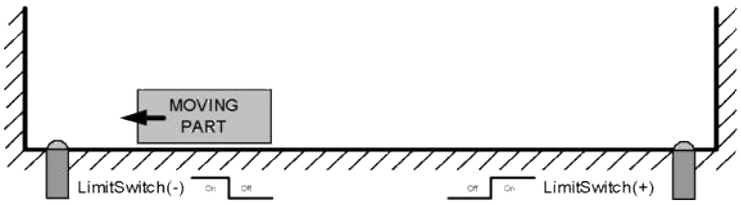
	Unit	n/a
CommandAborted	Description	Indicates the move was aborted
	Data type	BOOL
	Unit	n/a
Error	Description	Indicates an invalid input was specified or the move was terminated due to an error
	Data type	BOOL
	Unit	n/a
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type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

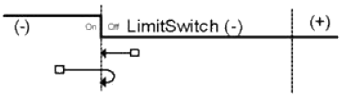
Value	Description
1	TimeLimit exceeded
2	DistanceLimit exceeded
3	TorqueLimit exceeded
4	axis error stop state
5	axis not enabled
6	invalid inputs for Velocity-Acceleration-Deceleration

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





Related Functions

MCFB_StepAbsolute

MCFB_StepRefPulse

MCFB_StepBlock

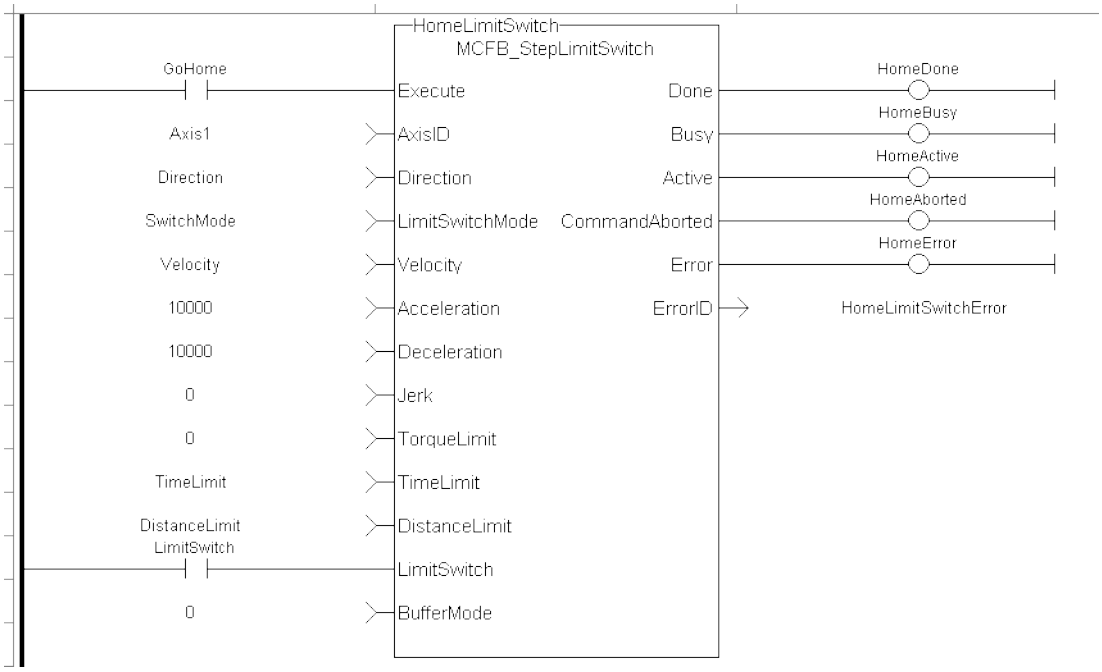
MCFB_StepAbsSwitch

Example

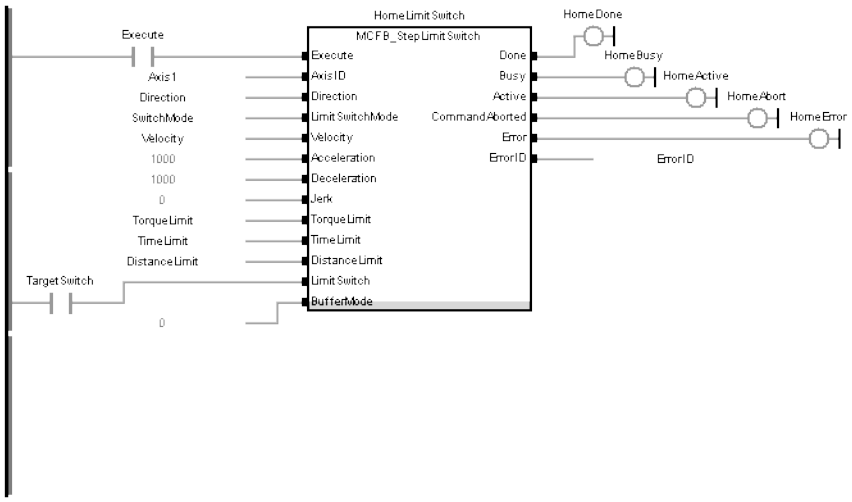
Structured Text

```
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, Lim-
itSwitch, 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 *)
```

Ladder Diagram



Function Block Diagram



5.2.7.13 MCFB_StepRefPulse

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:

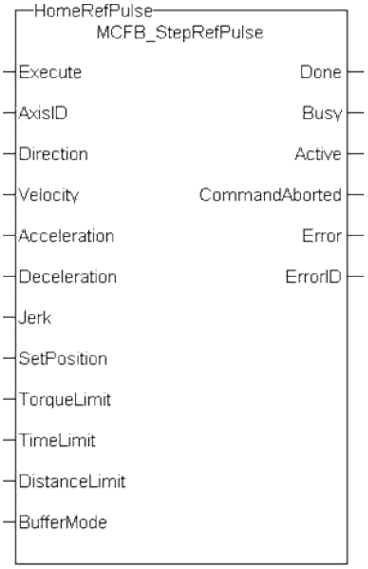


Figure 1-142: MCFB StepRefPulse

Arguments

Input

Execute	Description	Request the homing step procedure at rising edge
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
AxisID	Description	Name of a declared instance of the AXIS_REF library function
	Data type	AXIS_REF
	Range	[1 , 256]
	Unit	n/a
	Default	—
Direction	Description	Define the axis homing direction

Value	Description
0	clockwise rotation
1	counterclockwise rotation

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

SwitchMode	Description	Switch state to complete homing Value: 0 — switch is on; 1 — switch if off; 2 — rising edge of switch; 3 — falling edge of switch
	Data type	DINT
	Range	[0 , 3]
	Unit	n/a
	Default	—
Velocity	Description	Commanded velocity for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	Description	Commanded acceleration for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Deceleration	Description	Commanded deceleration for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Jerk	Description	Commanded jerk for the homing move (if zero, then trapezoidal acc/dec is used)
	Data type	LREAL
	Range	—
	Unit	User unit/sec^3
	Default	—
SetPosition	Description	Value of the absolute position to be set when the homing move is done
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
TorqueLimit	Description	Maximum torque applied for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—

TimeLimit	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	—
DistanceLimit	Description	Maximum distance for homing move to complete. If exceeded the homing procedure will error out. 0= no distance limit
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
BufferMode	Description	Define the homing move start action

Value	Description
0	abort
1	buffer
2	Blend to active
3	blend to next
4	blend to low velocity
5	blend to high velocity

	Data type	SINT
	Range	[0 , 5]
	Unit	n/a
	Default	—

Output

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

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

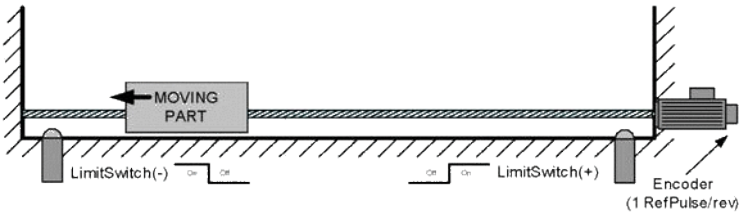
Value	Description
1	TimeLimit exceeded
2	DistanceLimit exceeded
3	TorqueLimit exceeded
4	axis error stop state
5	axis not enabled
6	invalid inputs for Velocity-Acceleration-Deceleration

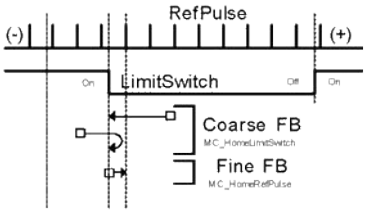
Data type	INT
Unit	n/a

Usage

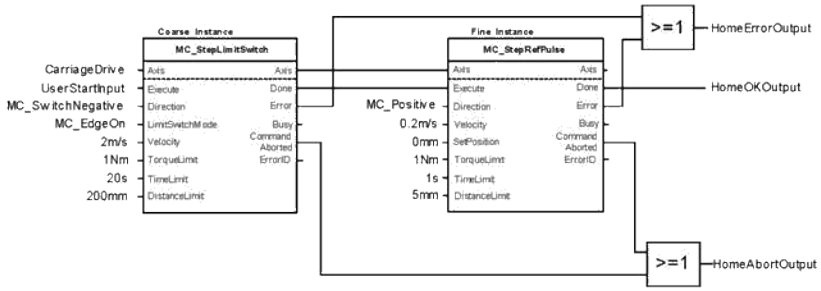
This function Block performs homing by searching for Zero pulse (also called Marker or reference pulse) in encoder. The reference pulse appears once per encoder revolution.

- 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).



Related Functions

MCFB_StepAbsolute

MCFB_StepAbsSwitch

MCFB_StepBlock

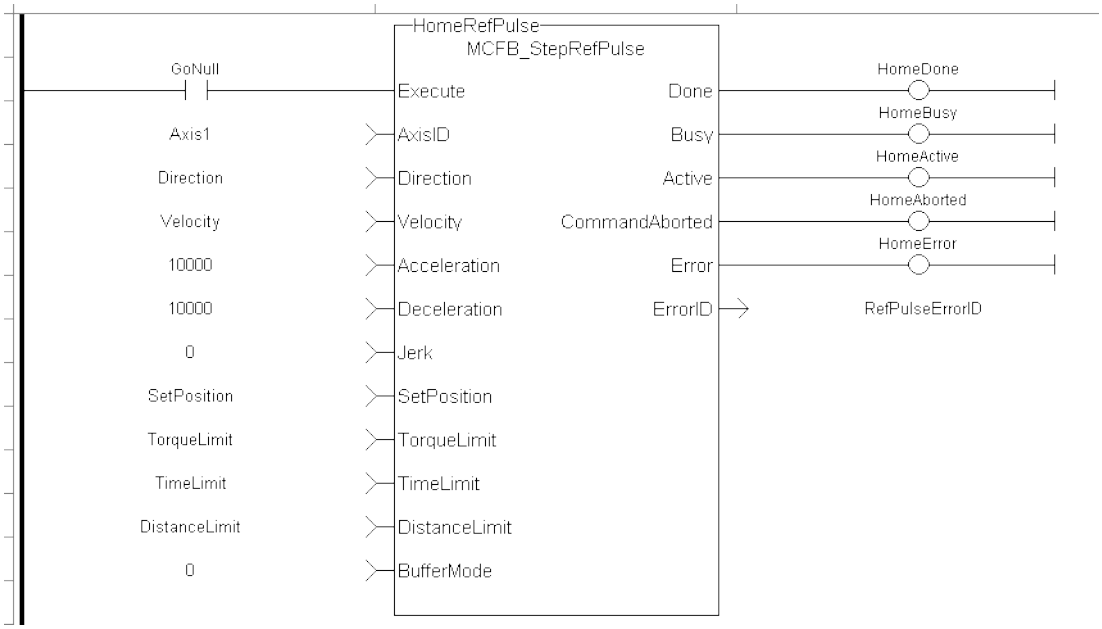
MCFB_StepLimitSwitch

Example

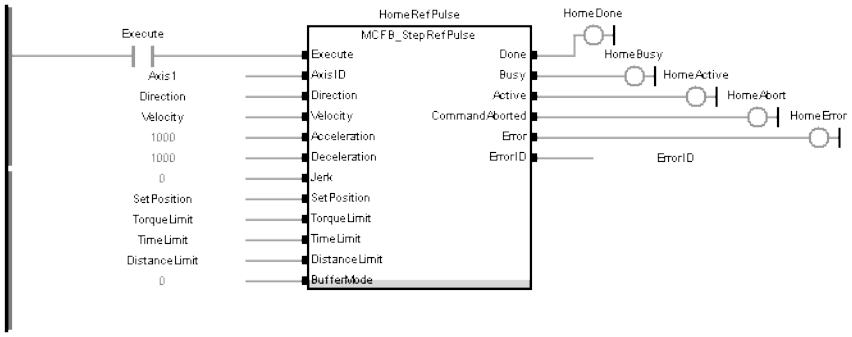
Structured Text

```
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;
```

Ladder Diagram



Function Block Diagram



5.2.7.14 MCFB_StepAbsSwitchFastInput

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:

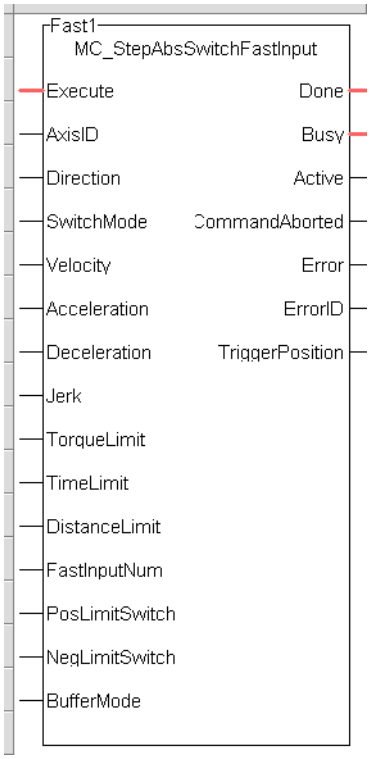


Figure 1-143: MCFB StepAbsSwitchFastInput

Input

Execute	Description	Request the homing step procedure at rising edge. Outputs are reset when execute input is false.
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
AxisID	Description	Structure for specified Axis desired to home
	Data type	AXIS_REF
	Range	[1 , 256]
	Unit	n/a
	Default	—
Direction	Description	Define the axis homing direction

Value	Description
0	clockwise rotation
1	counterclockwise rotation

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

SwitchMode	Description	Switch state to complete homing Value: 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
	Data type	DINT
	Range	[0 , 3]
	Unit	n/a
	Default	—
Velocity	Description	Commanded velocity for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	Description	Commanded acceleration for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Deceleration	Description	Commanded deceleration for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Jerk	Description	Commanded jerk for the homing move (if zero, then trapezoidal acc/dec is used)
	Data type	LREAL
	Range	—
	Unit	User unit/sec^3
	Default	—
TorqueLimit	Description	Maximum torque applied for the homing move entered in thousandths of maximum torque, e.g. "250" is 250/1000, or 25%.
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
TimeLimit	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	—
DistanceLimit	Description	Maximum distance for homing move to complete. If exceeded the homing procedure will error out. 0= no distance limit
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
FastInputNum	Description	0 for first fast input (X7 Pin 10), 1 for second fast input (X7 pin 9)
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
PosLimitSwitch	Description	The positive direction limit switch input I/O point
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
NegLimitSwitch	Description	The negative direction limit switch input I/O point
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
BufferMode	Description	Define the homing move start action

Value	Description
0	abort
1	buffer
2	Blend to active
3	blend to next
4	blend to low velocity
5	blend to high velocity

	Data type	SINT
	Range	[0 , 5]
	Unit	n/a
	Default	—

Output

Done	Description	Indicates the move completed successfully. The Command Position has reached the endpoint
	Data type	BOOL
	Unit	n/a
Busy	Description	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
	Unit	n/a

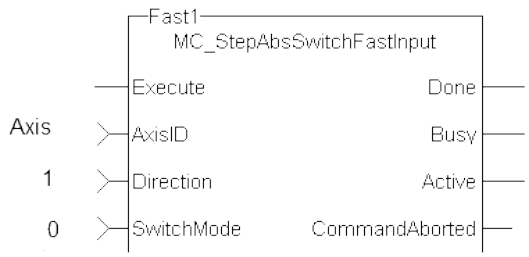
Active	Description	Set when the function block is active
	Data type	BOOL
	Unit	n/a
CommandAborted	Description	Indicates the move was aborted
	Data type	BOOL
	Unit	n/a
Error	Description	Signals that an error has occurred within the function block
	Data type	BOOL
	Unit	n/a
ErrorID	Description	Indicates the error if Error output is set to TRUE
	Data type	INT
	Unit	n/a
TriggerPosition	Data type	LREAL
	Range	-
	Unit	User units
	Default	-

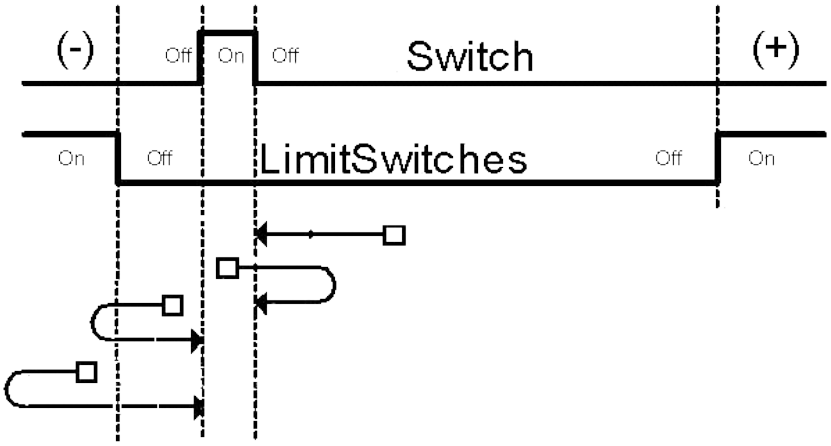
Value	Description
1	TimeLimit exceeded
2	DistanceLimit exceeded
3	TorqueLimit exceeded
4	axis error stop state
5	axis not enabled
6	invalid inputs for Velocity-Accel-Decel

Usage

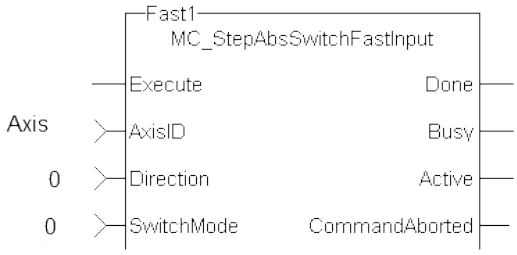
- The homing is commanded in the most likely direction were the sensor can be found. In this example (-).
- If any LimitSwitch is found during Homing (any of them), then a special process is started in the opposite direction, the AbsSwitch is searched to switch off (or On, depending on SwitchMode setting). 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.

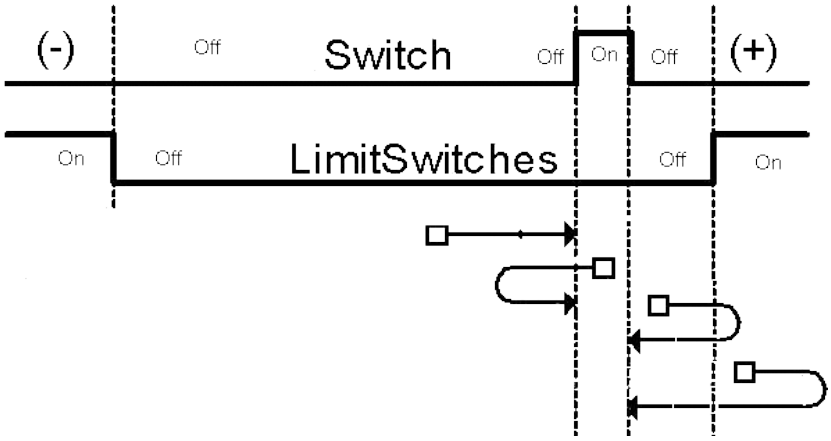
.



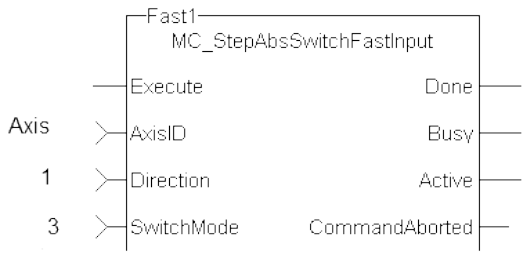


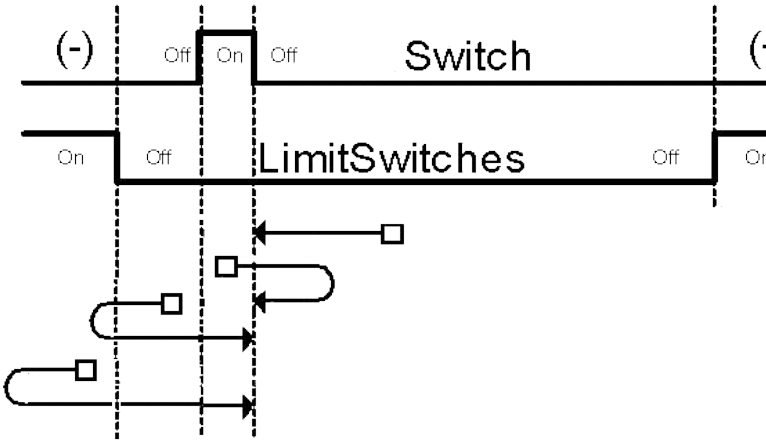
If the switch does not change states the motion will continue back and forth between the LimitSwitches



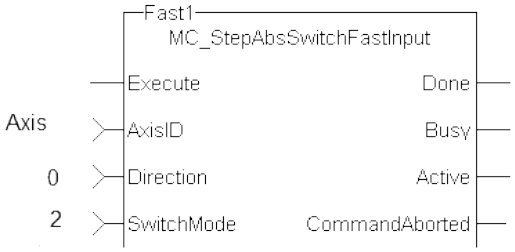


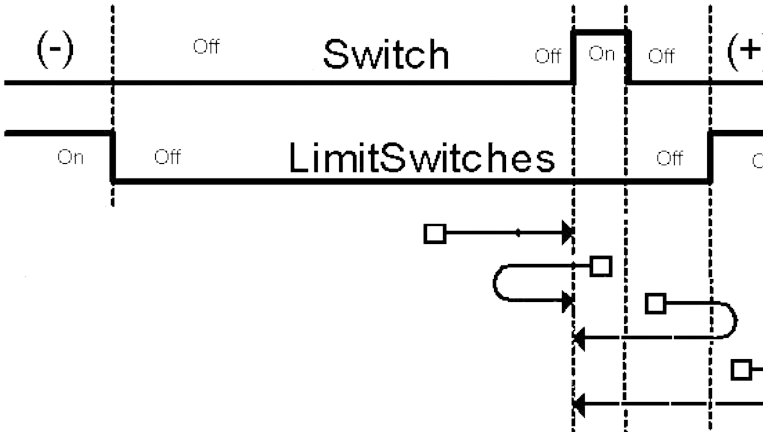
If the switch does not change states the motion will continue back and forth between the LimitSwitches





If the switch does not change states the motion will continue back and forth between the LimitSwitches





If the switch does not change states the motion will continue back and forth between the LimitSwitches

Related Functions

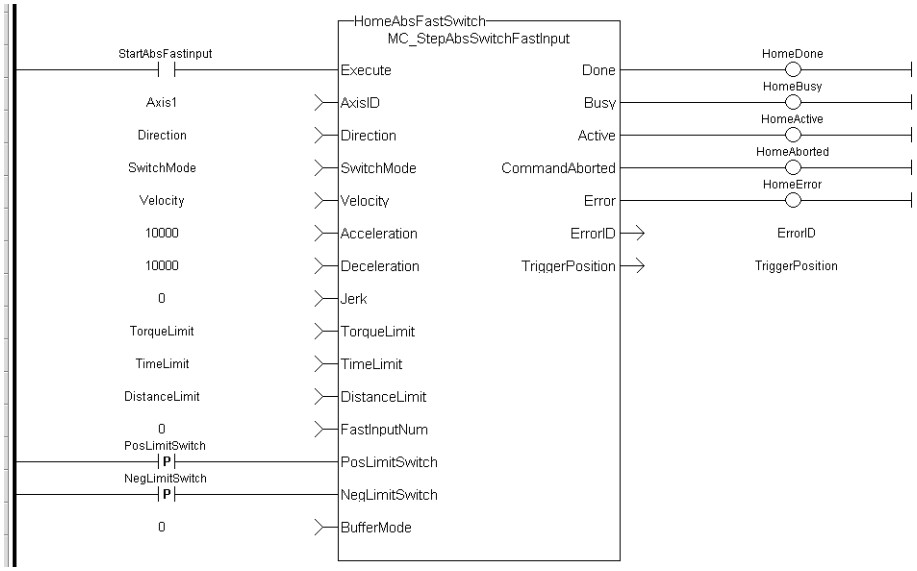
MCFB_StepLimitSwitchFastInput

Example

Structured Text

```
Execute_1 :=1;

(*Positive_Switch and Negitive_Switch are physical hardware in Dic-
tionary. *)


Inst_MC_StepAbsSwitchFastInput( Execute_1, Axis1, 0, 0, 10000.0,Ac-
celeration:=10000.0, 10000.0, 0, 0, 0, 0, 0, Positive_Switch , Negit-
ive_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;
```

Ladder Diagram



(* PosLimitSwitch, NegLimitSwitch are declared I/O points *)

5.2.7.15 MCFB_StepLimitSwitchFastInput

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:

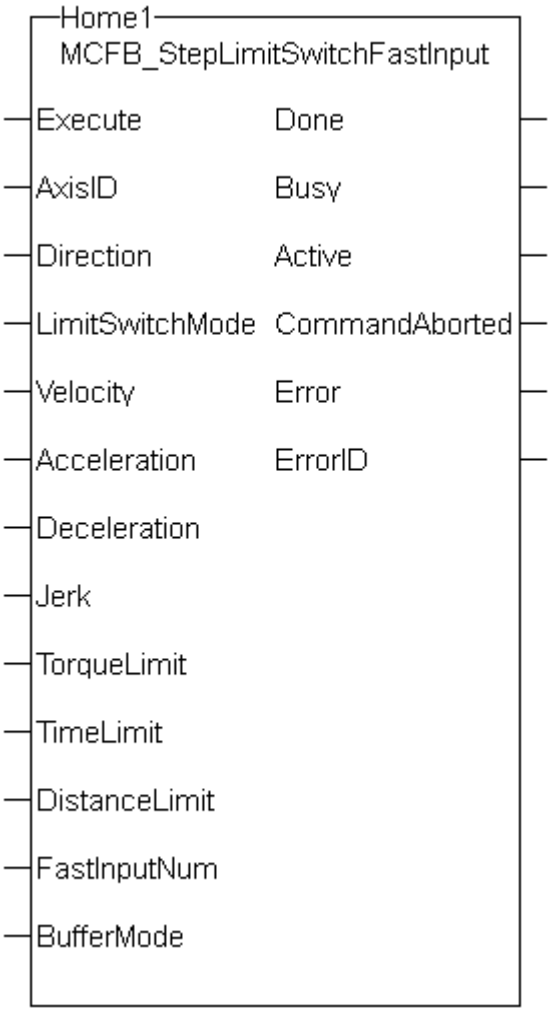


Figure 1-144: MCFB StepLimitSwitchFastInput

Input

Execute

Description	Request the homing step procedure at rising edge. Outputs are reset when execute input is false.
Data type	BOOL
Range	[0 , 1]
Unit	n/a
Default	—

AxisID

Description	Structure for specified Axis desired to home
Data type	AXIS_REF
Range	[1 , 256]
Unit	n/a
Default	—

Direction

Description	Define the axis homing direction

Value	Description
0	clockwise rotation
1	counterclockwise rotation

	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
LimitSwitchMode	**Description**	Limit switch state to complete homing

Value	Description
0	when rising edge of sensor
1	when falling edge

	Data type	DINT
	Range	[0 , 1]
	Unit	n/a
	Default	—
Velocity	**Description**	Commanded velocity for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Acceleration	**Description**	Commanded acceleration for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Deceleration	**Description**	Commanded deceleration for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit/sec^2
	Default	—
Jerk	**Description**	Commanded jerk for the homing move (if zero, then trapezoidal acc/dec is used)
	Data type	LREAL
	Range	—
	Unit	User unit/sec^3
	Default	—
TorqueLimit	**Description**	Maximum torque applied for the homing move
	Data type	LREAL
	Range	—
	Unit	User unit entered in thousandths of maximum torque, e.g. "250" is 250/1000, or 25%.
	Default	—
TimeLimit	**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	—
DistanceLimit	**Description**	Maximum distance for homing move to complete. If exceeded the homing procedure will error out. 0= no distance limit
	Data type	LREAL
	Range	—
	Unit	User unit
	Default	—
FastInputNum	**Description**	0 for first fast input (X7 Pin 10), 1 for second fast input (X7 pin 9)
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
BufferMode	**Description**	Define the homing move start action

Value	Description
0	abort
1	buffer
2	Blend to active
3	blend to next
4	blend to low velocity
5	blend to high velocity

	Data type	SINT
	Range	[0 , 5]
	Unit	n/a
	Default	—

Output

Done	**Description**	Indicates the move completed successfully. The Command Position has reached the endpoint
	Data type	BOOL
	Unit	n/a
Busy	**Description**	High from the moment the Execute input is one-shot to the time the move is ended
	Data type	BOOL
	Unit	n/a
Active	**Description**	Set when the function block is active
	Data type	BOOL
	Unit	n/a
CommandAborted	**Description**	Indicates the move was aborted
	Data type	BOOL
	Unit	n/a
Error	**Description**	Signals that an error has occurred within the function block
	Data type	BOOL

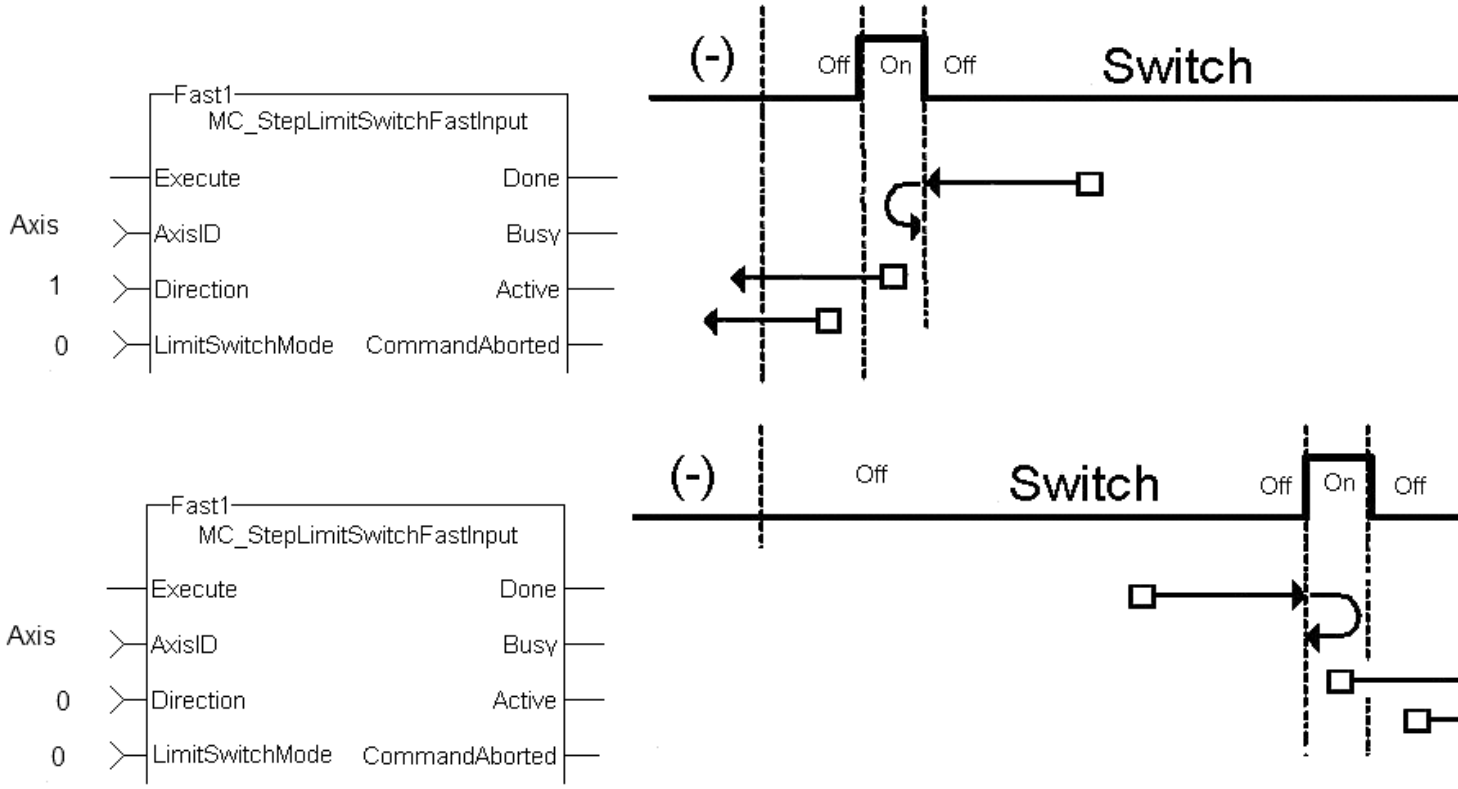
	Unit	n/a
ErrorID	**Description**	Indicates the error if Error output is set to TRUE

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-Accel-Decel

	Data type	INT
	Unit	n/a

Usage

The homing is commanded in the most likely direction were the sensor can be found. In this example (-).



Related Functions

MCFB_StepAbsSwitchFastInput

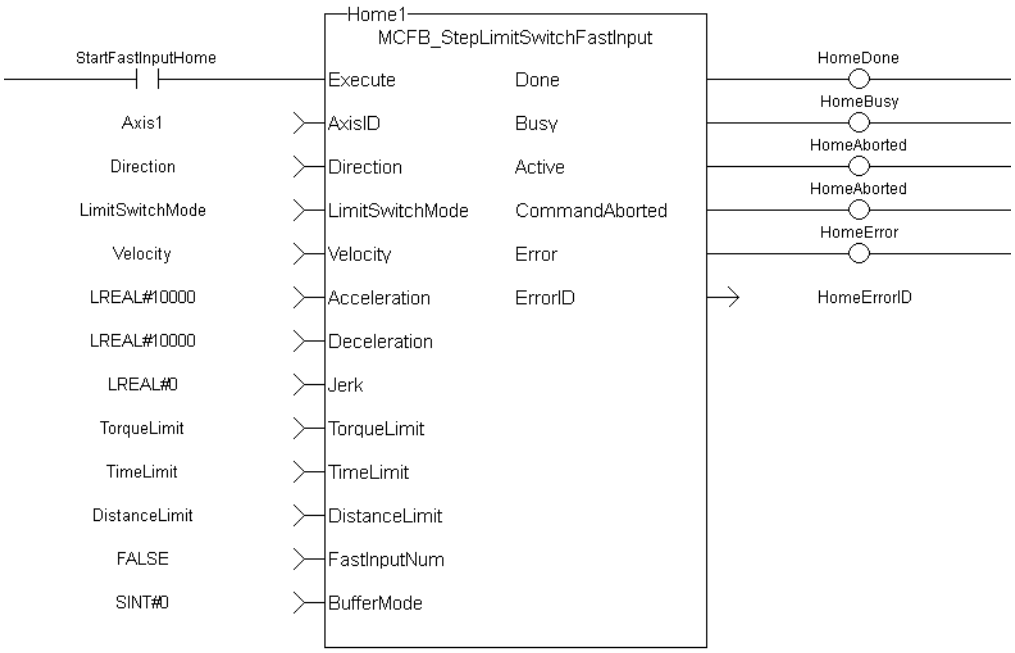
Example

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;
```

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

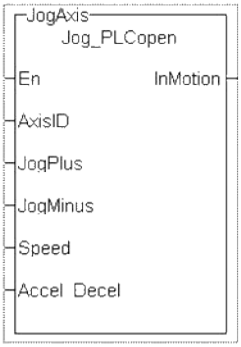


Figure 1-145: Jog for PLCopen

5.2.8.2 Arguments

Input

En	Description	Enables execution (FFLD only)
	Data type	BOOL
	Range	—
	Unit	n/a
	Default	—
AxisID	Description	ID Name of the Axis
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—
JogPlus	Description	Enables a Jog in the plus direction
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
JogMinus	Description	Enables a Jog in the Minus direction
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
Speed	Description	Rate at which the axis will move
	Data type	LREAL
	Range	—
	Unit	User unit/sec
	Default	—
Accel Decel	Description	Linear Acc/Dec rate
	Data type	LREAL

	Range	—
	Unit	User unit/sec^2
	Default	—

Output

InMotion	Description	Jogging is active when TRUE
	Data type	BOOL
	Unit	n/a

5.2.8.3 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.

5.2.8.4 Related Functions

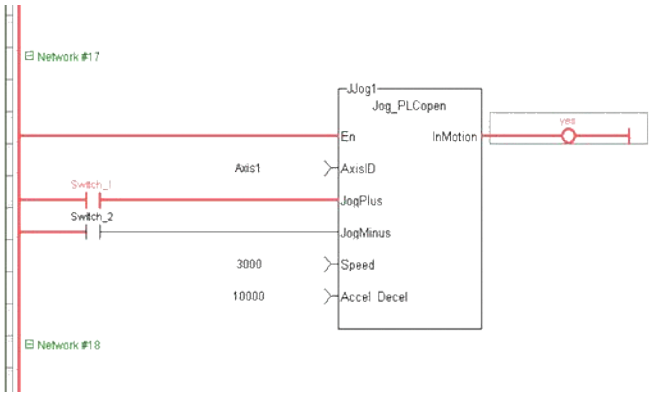
MC_MoveVelocity

5.2.8.5 Example

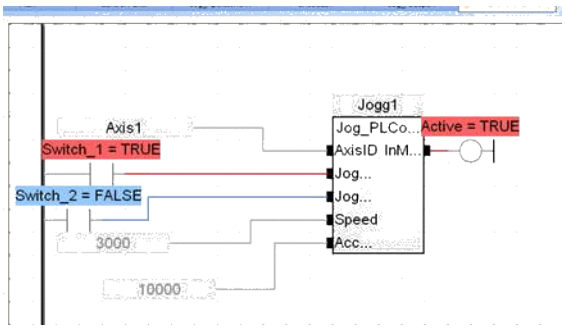
Structured Text

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

Ladder Diagram



Function Block Diagram



5.2.9 MCFB_GearedWebTension

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.

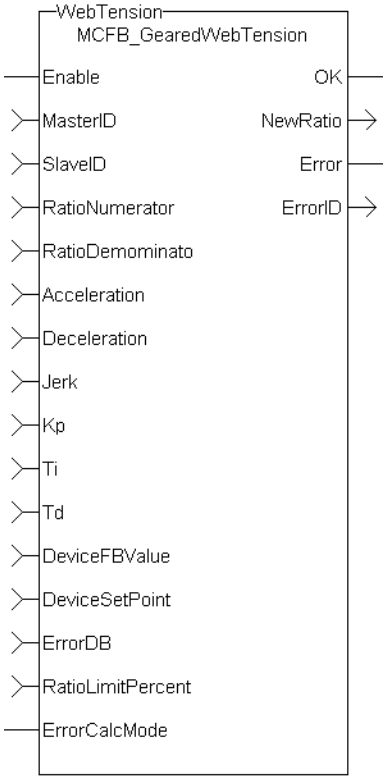


Figure 1-146: MCFB_GearedWebTension Function Block I/O

5.2.9.1 Arguments

Inputs

Enable	**Description**	Enables execution
	Data Type	BOOL
	Range	[0,1]
	Unit	n/a
	Default	-
MasterID	**Description**	Identifies the master axis
	Data Type	AXIS_REF
	Range	
	Unit	n/a
	Default	-
SlaveID	**Description**	Identifies the slave axis

	Data Type	AXIS_REF
	Range	
	Unit	n/a
	Default	-
RatioNumerator	**Description**	Numerator of the master/slave ratio
	Data Type	DINT
	Range	[-2147483648 to +2147483647]
	Unit	n/a
	Default	-
RatioDenominator	**Description**	Denominator of the master/slave ratio
	Data Type	DINT
	Range	[-2147483648 to +2147483647]
	Unit	n/a
	Default	-
Acceleration	**Description**	Trapezoidal: acceleration rate, S-Curve: maximum acceleration
	Data Type	LREAL
	Range	[-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)]
	Unit	n/a
	Default	-
Deceleration	**Description**	Trapezoidal: deceleration rate, S-Curve: not used
	Data Type	LREAL
	Range	[-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)]
	Unit	n/a
	Default	-
Jerk	**Description**	Trapezoidal: 0, S-Curve: constant jerk
	Data Type	LREAL
	Range	[-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)]
	Unit	n/a
	Default	-
Kp	**Description**	Proportional gain
	Data Type	LREAL
	Range	[-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)]
	Unit	n/a
	Default	-
Ti	**Description**	Integral gain
	Data Type	LREAL
	Range	[-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)]
	Unit	n/a
	Default	-

Td	**Description**	Derivative gain
	Data Type	LREAL
	Range	[-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)]
	Unit	n/a
	Default	-
DeviceFBValue	**Description**	Analog input
	Data Type	DINT
	Range	[-2147483648 to +2147483647]
	Unit	n/a
	Default	-
DeviceSetPoint	**Description**	Analog set point
	Data Type	DINT
	Range	[-2147483648 to +2147483647]
	Unit	n/a
	Default	-
ErrorDB	**Description**	Maximum or minimum error between DeviceFBValue and DeviceSetPoint before a change will take place.
	Data Type	LREAL
	Range	[-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)]
	Unit	n/a
	Default	-
RatioLimitPercent	**Description**	Maximum and minimum master/slave ratio window
	Data Type	LREAL
	Range	[-1.7E308 to 1.7E308 and -1.7E-308 to 1.7E-308 (14 to 15 significant digits of accuracy)]
	Unit	n/a
	Default	-
ErrorCalcMode	**Description**	Not set: DeviceFBValue-DeviceSetPoint, Set: DeviceSetPoint-DeviceFBValue
	Data Type	BOOL
	Range	[0,1]
	Unit	n/a
	Default	-

Output

OK	**Description**	The output will have power flow after the enable input has been energized.
	Data Type	BOOL
	Range	[0,1]
	Unit	n/a
NewRatio	**Description**	New master/slave ratio
	Data Type	REAL

	Range	[-3.4E38 to 3.4E38 and -3.4E-38 to 3.4E-38 (6 to 7 significant digits of accuracy)]
	Unit	n/a
Error	**Description**	Function block error
	Data Type	BOOL
	Range	[0,1]
	Unit	n/a
ErrorID	**Description**	Function block error value
	Data Type	INT
	Range	[-32768 to +32767]
	Unit	n/a

5.2.9.2 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.

Example 1

NOTE

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

RatioNumerator = 1
RatioDemominator = 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 RatioDemominator = (RatioDemoninator - 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 that the previous ration the slave axis is going faster and the tension is reduced.

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))

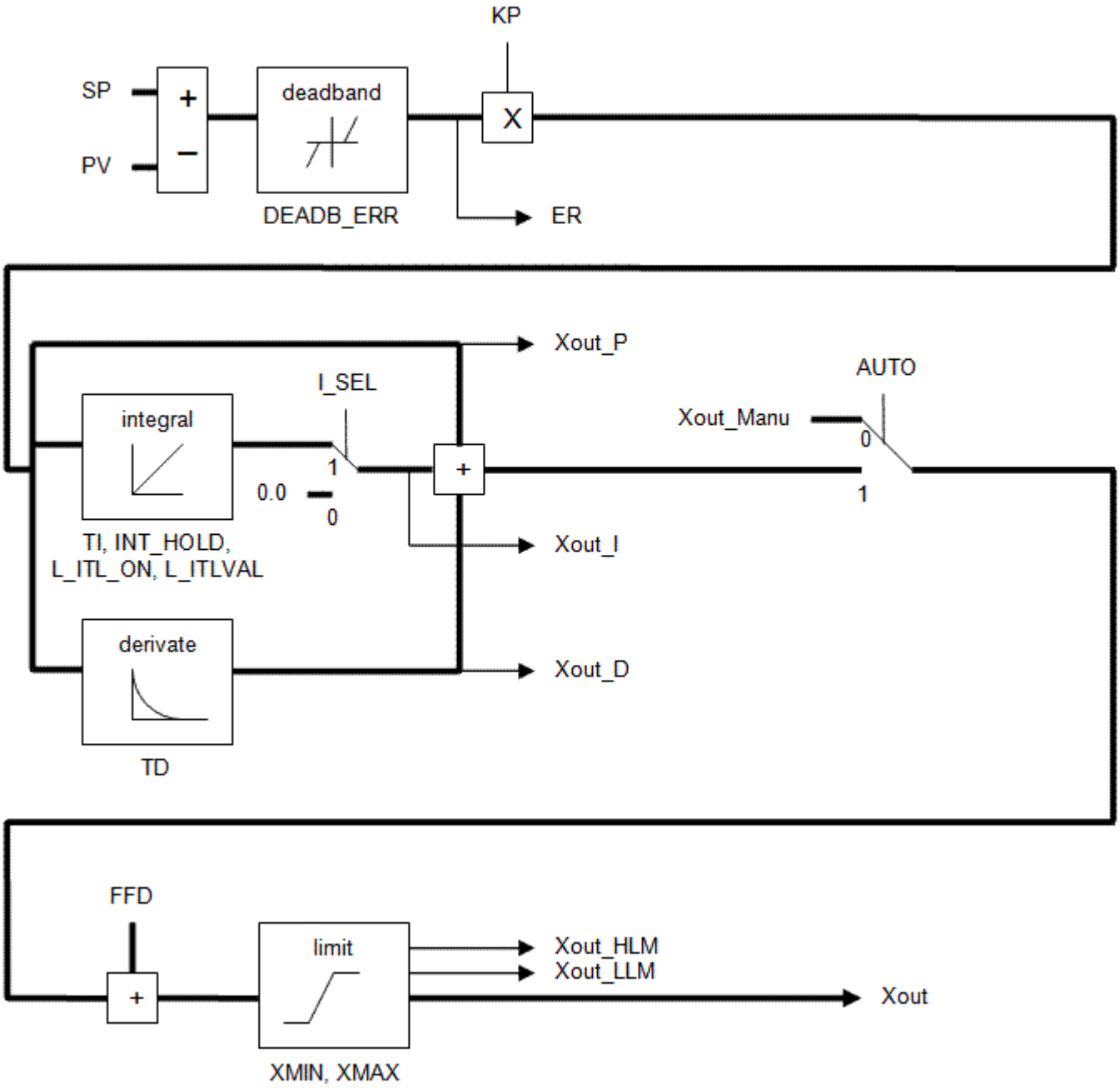
Therefore the new RatioDenominator = (2 + 0.005*2) = 2.01

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.

PID Function in KAS:

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



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.

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 KP = 1, Ti and Td =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 is a significant change to the slave velocity that will probably cause excess slack or web breakage.

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 KP = 1, Ti and Td =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

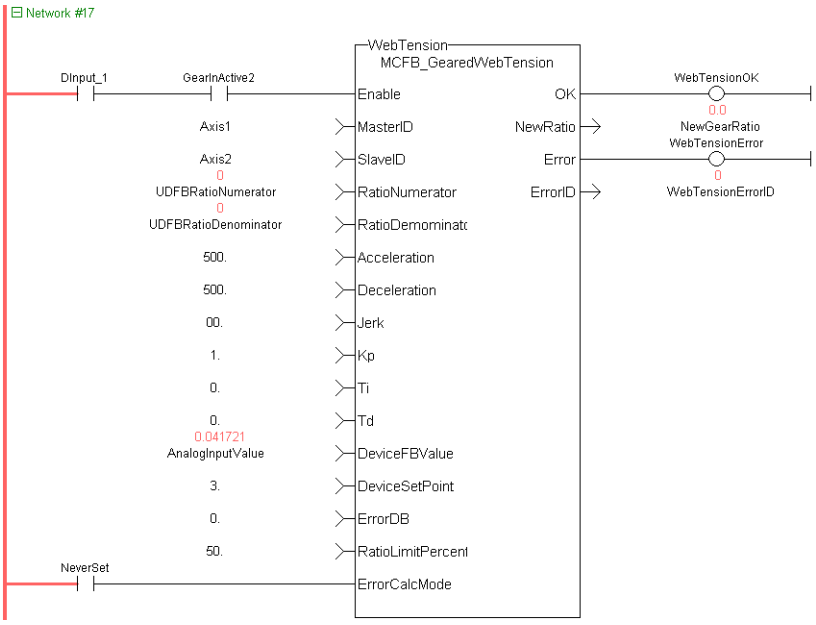
This resolution is much finer than example 1 so for a change of 1 for the error output of the PID this creates a Master/Slave ratio change of 1999. This results is a slower rate of change to the slave velocity that the more suited to good tension in a machine process.

5.2.9.3 Related Functions

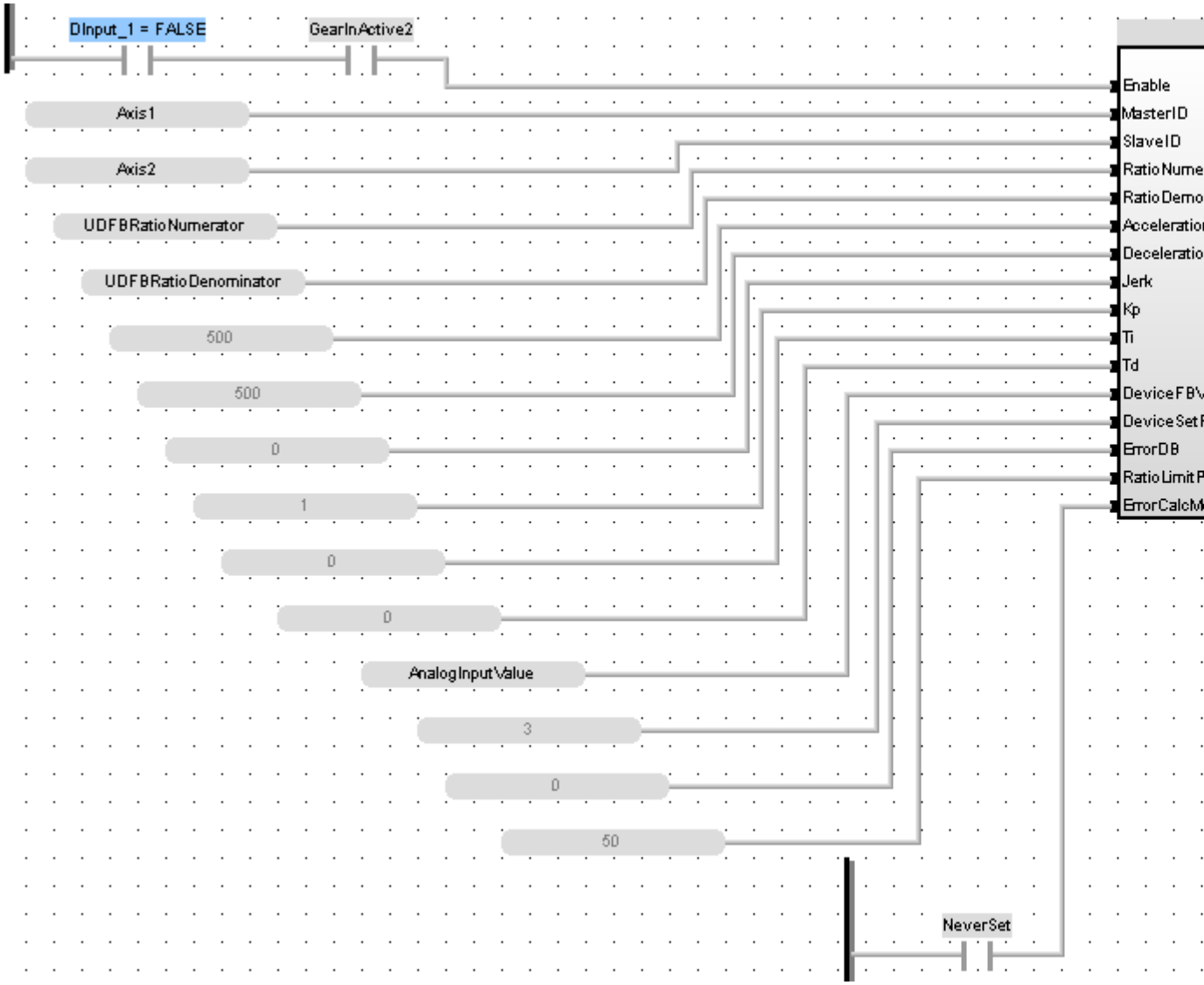
" FB_FirstOrderDigitalFilter" (➜ p. 643)

5.2.9.4 Example

Ladder Example



Function Block Diagram Example



Structured Text Example

```
Inst_MCFB_GearedWebTension
( DInput_1 FALSE , Axis1, Axis2, UDFBRatioNumerator     0   , UDFBRati
500.0, 500.0, 0.0,1.0, 0.0,0.0, AnalogInputValue    0.0 , 3.0, 0.0, 50
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 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

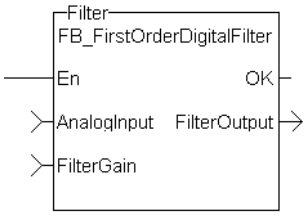


Figure 1-147: CBS First Order Digital Filter

5.2.10.2 Arguments

Inputs

EN	Description	Enables execution (FFLD only)
	Data type	BOOL
	Range	—
	Unit	n/a
	Default	
AnalogInput	Description	Analog Input from transducer
	Data type	INT
	Range	—
	Unit	n/a
	Default	
FilterGain	Description	Filter Gain
	Data type	REAL
	Range	[1 - 0.05]
	Unit	n/a
	Default	—

Outputs

OK	Description	Execution Complete
	Data type	BOOL
	Range	[0,1]
	Unit	
FilterOutput	Description	Filtered analog input value
	Data type	REAL
	Range	[0,1]
	Unit	

5.2.10.3 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.

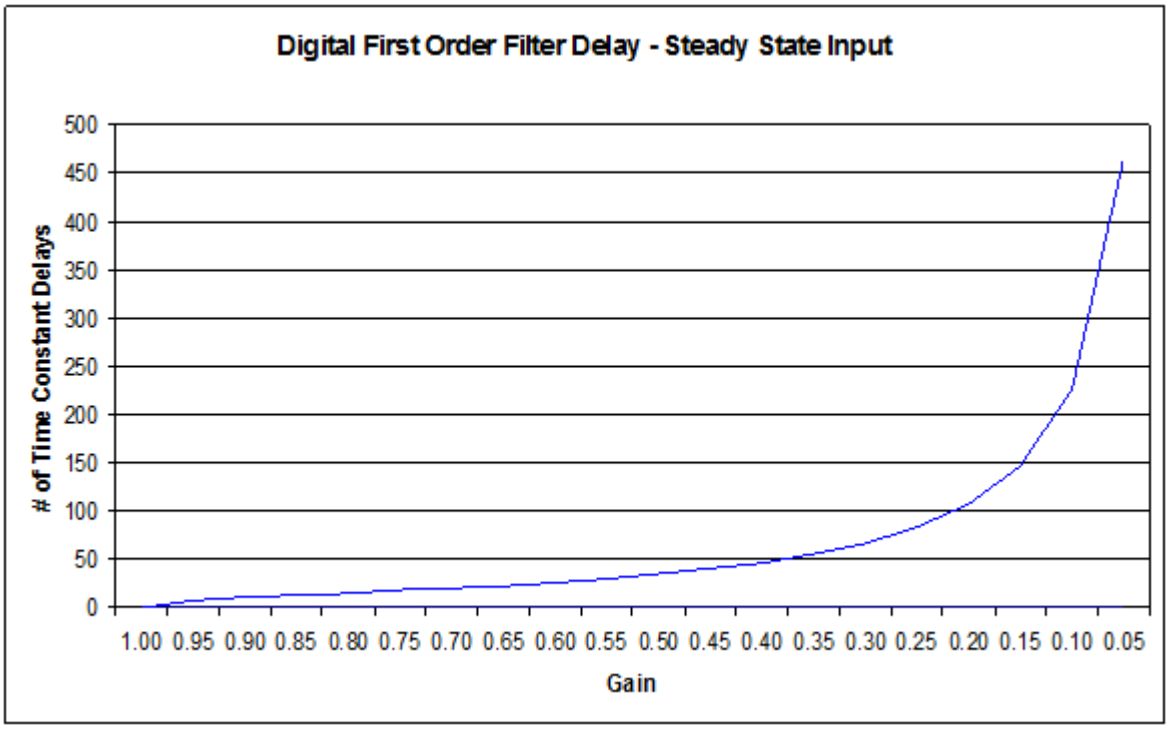
FilterGain	FilterInput	FilterOutput
0.8	0	0
	100	80
	100	96
	100	99.2
	100	99.84
	100	99.968
	100	99.9936
	100	99.99872
	100	99.999744
	100	99.9999488
	100	99.99998976
	100	99.99999795
	100	99.99999959
	100	99.99999992
	100	99.99999998
	100	100
	100	100
	100	100
	100	100
	100	100
	100	100
	100	100
	100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

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%

Filter Input	Filter Current Output	Amount of Input Filtering	Random Filter % Variation
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	
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	

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

Table 1-12: Filter Input Lag Example - Random Input

5.2.10.4 Related Functions

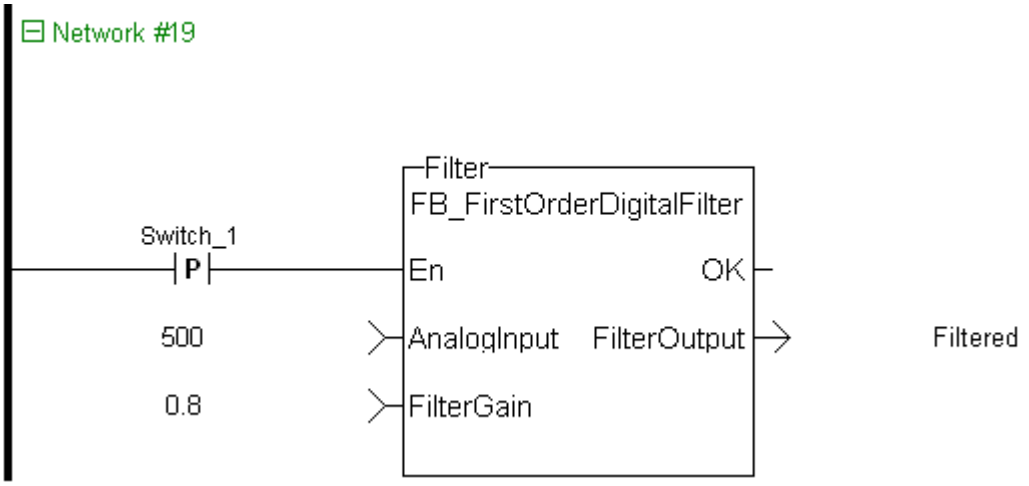
None.

5.2.10.5 Example

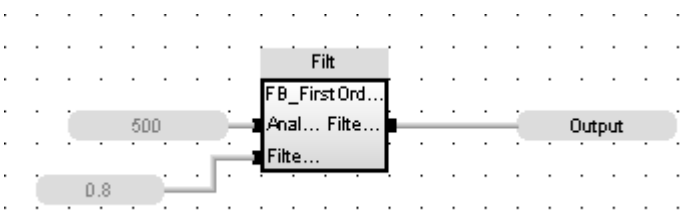
Structured Text

```
Inst_FB_FirstOrderDigitalFilter( AnalogInput:=500, FilterGain:=0.8 );
FilterOutput:= Inst_FB_FirstOrderDigitalFilter.FilterOutput
```

Ladder Diagram



Function Block Diagram



5.2.11 MCFB_AKDFault

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:

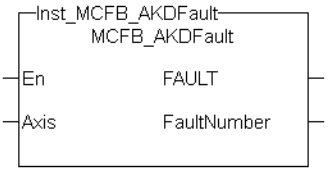


Figure 1-148: MCFB_AKDFault

5.2.11.2 Arguments

Input

EN	**Description**	ENABLES the Kollmorgen UDFB (used in FFLD editor only)
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
Axis	**Description**	Name of a declared instance of the AXIS_REF library function. For more details, About Axis Name and Number.
	Data type	AXIS_REF
	Range	[1, 256]
	Unit	n/a
	Default	—

Output

FAULT	**Description**	TRUE if selected drive currently has a Fault
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
FaultNumber	**Description**	Three digit AKD Fault identifier
	Data type	DINT
	Range	[100 , 999]
	Unit	n/a

5.2.11.3 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. 650)

"FB_AKDFltRpt" (➜ p. 653)

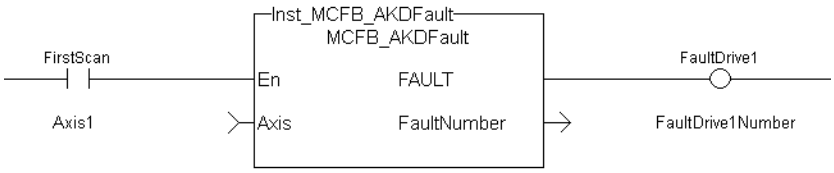
"MC_ReadStatus" (➜ p. 331) (PLCopen Motion Engine)

5.2.11.4 Example

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*)
);
```

Ladder Diagram



Function Block Diagram

5.2.12 MCFB_AKDFaultLookup

5.2.12.1 Description

String message of the corresponding AKD drive fault number. The OK output turns TRUE when the 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:

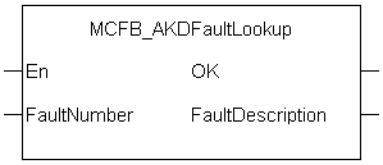


Figure 1-149: MCFB_AKDFaultLookup

5.2.12.2 Arguments

Input

EN	**Description**	ENABLES the Kollmorgen UDFB (used in FFLD editor only)
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
	Default	—
FaultNumber	**Description**	The AKD drive fault number
	Data type	DINT
	Range	—
	Unit	n/a
	Default	—

Output

OK	**Description**	TRUE if there is a match for the FaultNumber
	Data type	BOOL
	Range	[0 , 1]
	Unit	n/a
FaultDescription	**Description**	Description of the Fault
	Data type	STRING
	Range	n/a
	Unit	n/a

5.2.12.3 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.

5.2.12.4 Related Functions

"MCFB_AKDFault" (➜ p. 648)

"FB_AKDFltRpt" (➜ p. 653)

"MC_ReadStatus" (➜ p. 331) (PLCopen Motion Engine)

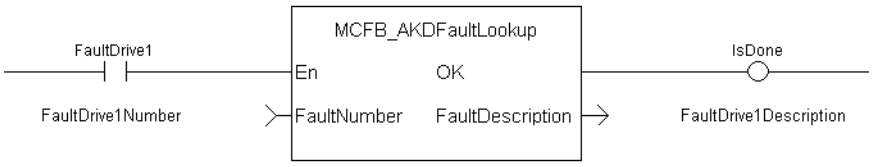
5.2.12.5 Example

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;
```

```
ultDrive1Description := MCFB_AKDFaultLookup( FaultDrive1Number(*DINT*)
```

Ladder Diagram



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

Input

iInA	Description	Set direction A
	Data type	BOOL
iInB	Description	Set direction B
	Data type	BOOL
iLsA	Description	Limit Switch at End of direction A
	Data type	BOOL
iLsB	Description	Limit Switch at End of direction B
	Data type	BOOL
iCtrlTime	Description	Max Time till Lim.Sw. has to be reached
	Data type	TIME
iResetFault	Description	Reset Fault (Is set to FALSE by UDFB!)
	Data type	BOOL

Output

oDirA	Description	Direction A
	Data type	BOOL
oDirB	Description	Direction B

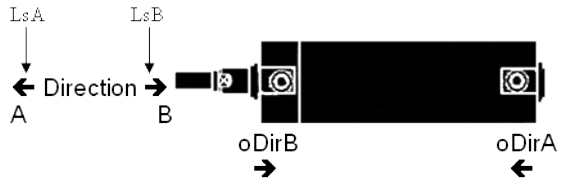
	Data type	BOOL
oFaultLsA	Description	Fault of Lim.Sw. at End direction A
	Data type	BOOL
oFaultLsB	Description	Fault of Lim.Sw. at End direction B
	Data type	BOOL

5.2.13.3 Usage

The signal flow is valid for both directions (A and B)

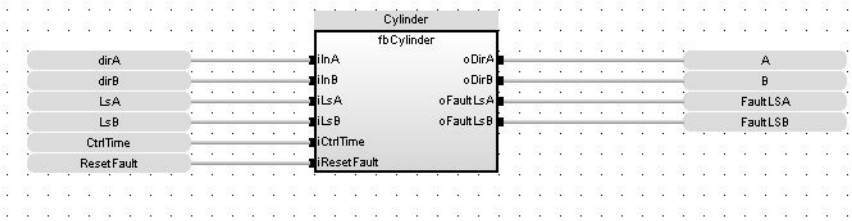
If oDirA AND oDirB are active there is no Fault Control.

The Fault can be reset by iRestFault = True.



5.2.13.4 Example

Function Block Diagram



5.2.13.5 FB_AKDFltRpt

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:

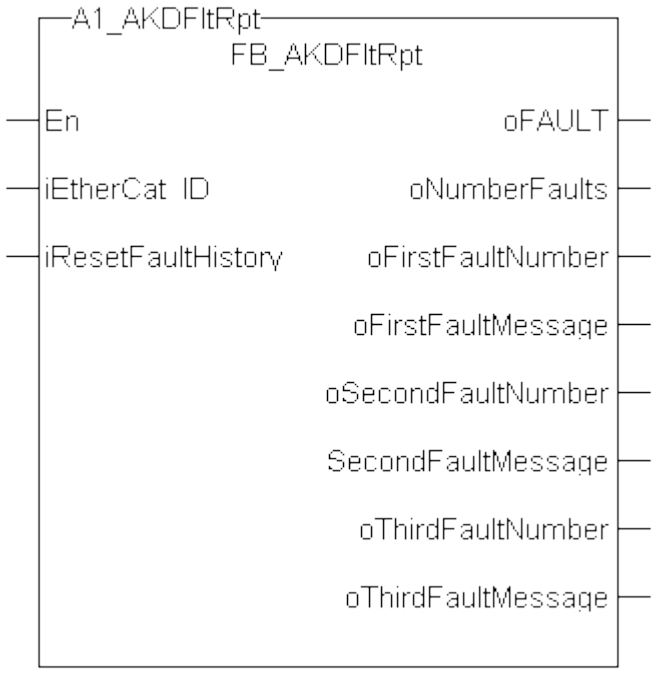


Figure 1-150: AKDFltRpt

Arguments

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	INT
	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	—

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	Three digit AKD Fault identifier
	Data type	DINT
	Range	[100 , 999]
	Unit	n/a
oFirstFaultMessage	Description	Description of the Fault
	Data type	STRING
	Unit	n/a
oSecondFaultNumber	Description	Three digit AKD Fault identifier.
	Data type	DINT
	Range	[100 , 999]
	Unit	n/a
oSecondFaultMessage	Description	Description of the Fault
	Data type	STRING
	Unit	n/a
oThirdFaultNumber	Description	Three digit AKD Fault identifier
	Data type	DINT
	Range	[100 , 999]
	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

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

MC_ReadStatus (PLCopen Motion Engine)

MLAxisStatus (Pipe Network Motion Engine)

Example

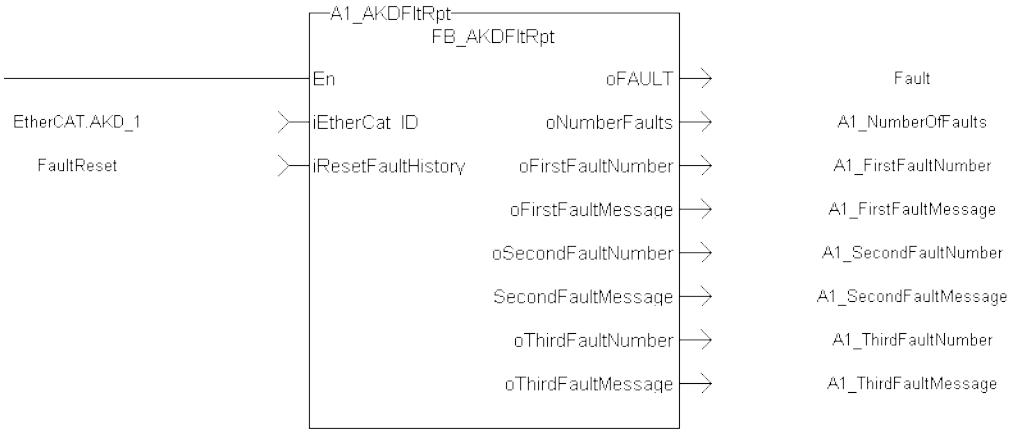
Structured Text

```
//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;
;
```

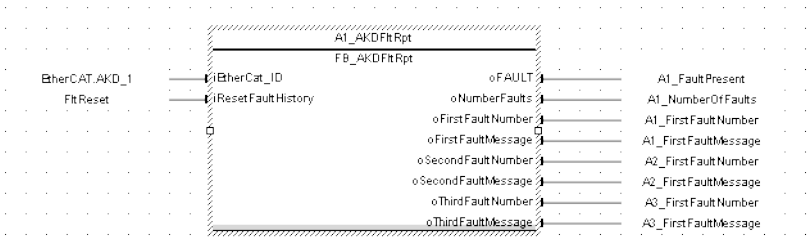
NOTE

A1_FaultReporting is an instance of the FB_S700FltRpt function block.

Ladder Diagram



Function Block Diagram



5.2.13.6 FB_S700FltRpt

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:

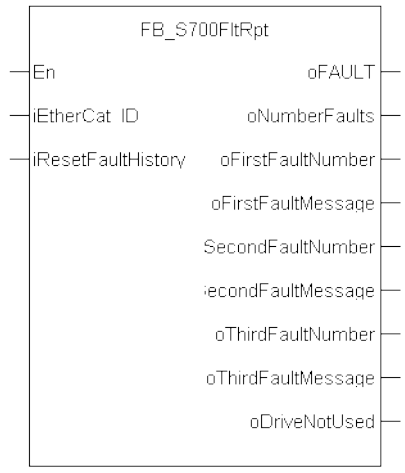


Figure 1-151: S700FltRpt

Arguments

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	—

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

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

MC_ReadStatus (PLCopen Motion Engine)

MLAxisStatus (Pipe Network Motion Engine)

Example

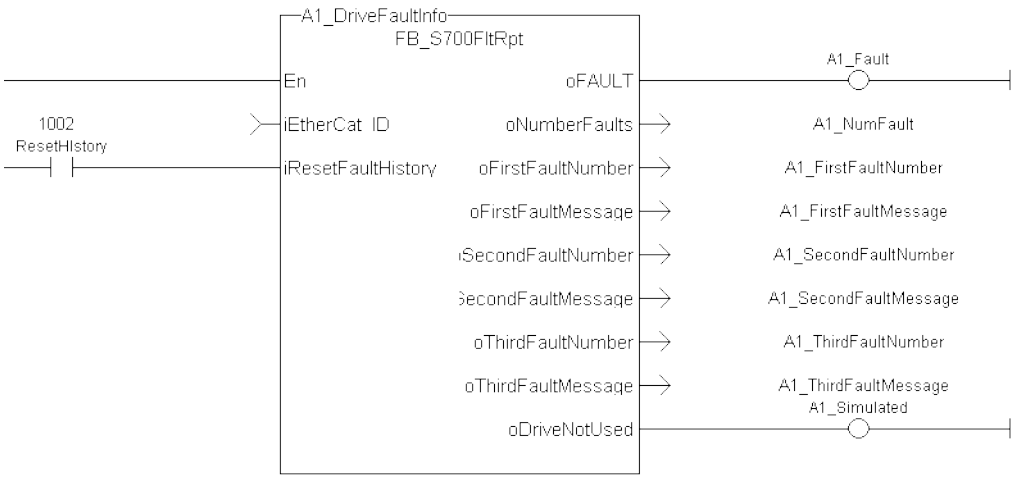
Structured Text

```
//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;
```

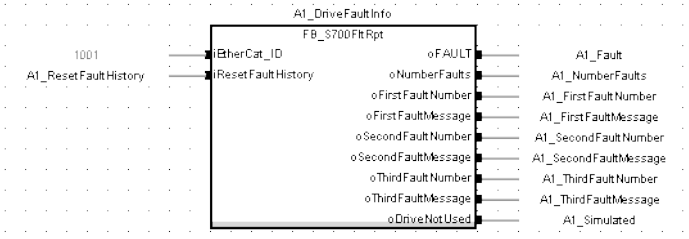
NOTE

A1_FaultReporting is an instance of the FB_S700FltRpt function block.

Ladder Diagram



Function Block Diagram



5.2.13.7 FB_AxisPlsPosModulo

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.

Arguments

Input

ibExecute	Description	Enable PLS
	Data type	BOOL
iPosition	Description	Any position of a modulo axis
	Data type	LREAL
iModuloPosition	Description	Modulo position of axis
	Data type	LREAL
iStartPos	Description	Start position of PLS
	Data type	LREAL
iEndPos	Description	End position of PLS
	Data type	LREAL

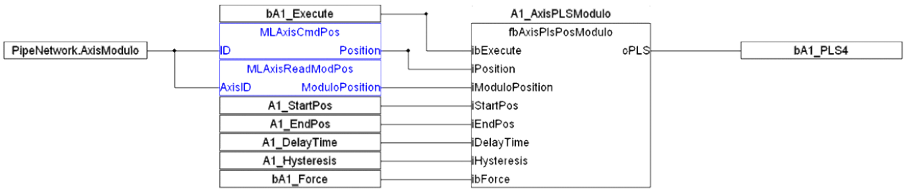
iDelayTime	Description	Delay time for compensation
	Data type	TIME
iHysteresis	Description	Hysteresis
	Data type	LREAL
ibForce	Description	Force PLS
	Data type	BOOL

Output

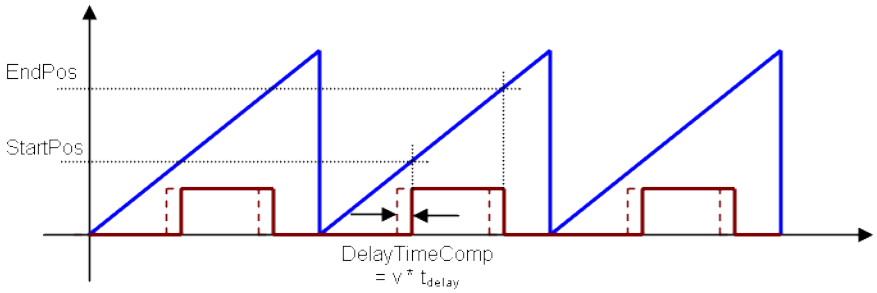
oPLS	Description	Position limit switch
	Data type	BOOL

Example

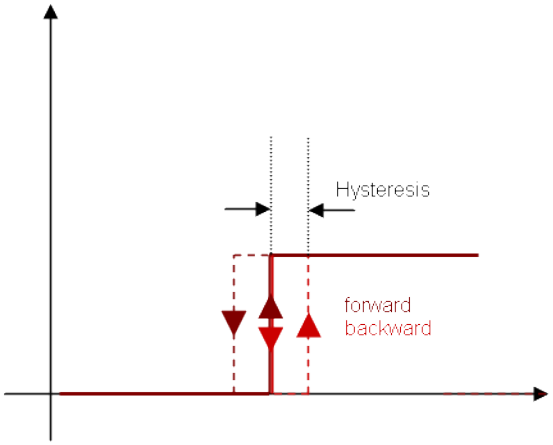
Function Block Diagram



Timing



Hysteresis



5.2.13.8 FB_AxisPlsPosNoModulo

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.

Arguments

Input

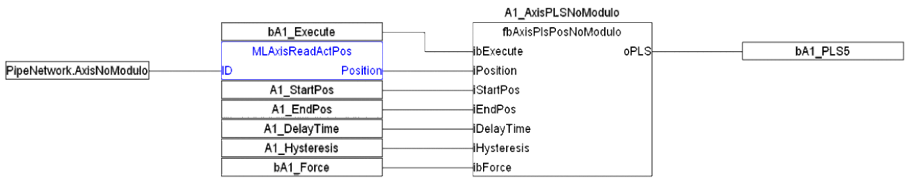
ibExecute	Description	Enable PLS
	Data type	BOOL
iPosition	Description	Any position of a none-modulo axis
	Data type	LREAL
iStartPos	Description	Start position of PLS
	Data type	LREAL
iEndPos	Description	End position of PLS
	Data type	LREAL
iDelayTime	Description	Delay time for compensation
	Data type	TIME
iHysteresis	Description	Hysteresis
	Data type	LREAL
ibForce	Description	Force PLS
	Data type	BOOL

Output

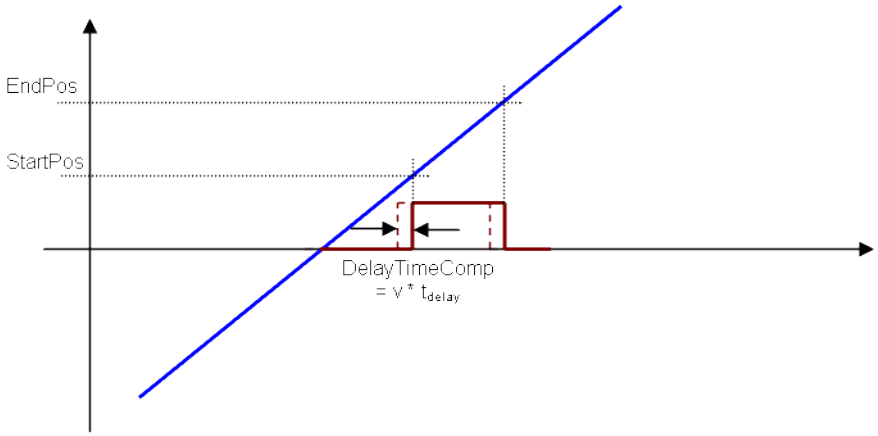
oPLS	Description	Position limit switch
	Data type	BOOL

Example

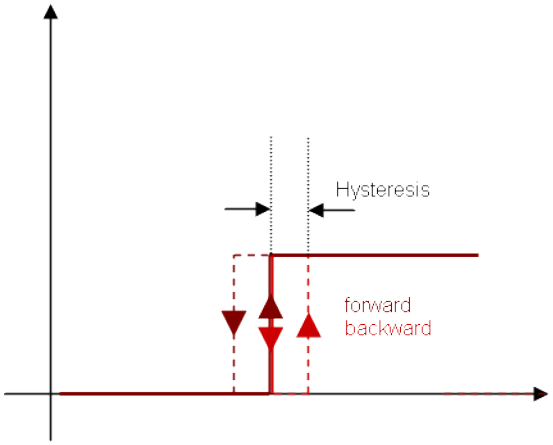
Function Block Diagram



Timing


EndPos
StartPos
DelayTimeComp
= v * t_{delay}

Hysteresis


Hysteresis
forward
backward

6 Index

A

C

D

E

F

G

H

I

J

K

M

motion library function

P

R

S

T

U

Z

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
Mail: support@kollmorgen.com
Tel.: +1 - 540 - 633 - 3545
Fax: +1 - 540 - 639 - 4162

Europe
KOLLMORGEN Europe GmbH
Pempelfurtstraße 1
40880 Ratingen
Germany

Web: www.kollmorgen.com
Mail: technik@kollmorgen.com
Tel.: +49 - 2102 - 9394 - 0
Fax: +49 - 2102 - 9394 - 3155

China and SEA
KOLLMORGEN
Room 202, Building 3, Lane 168,
Lin Hong Road, Changning District
Shanghai

Web: www.kollmorgen.cn
Mail: sales.china@kollmorgen.com
Tel.: +86 - 400 661 2802
Fax: +86 - 21 6128 9877